Apolo Scientific Computing Center's Documentation

Release 0.1

The Apolo Team

Aug 24, 2023

Contents

Apolo is the scientific computing center at Universidad EAFIT, Medellín, Colombia.

CHAPTER 1

Getting Started

To compute in Apolo it is necessary to take into account several considerations. In this section you will find the minimum conditions you must meet to be able to compute on the clusters of the center and what is the procedure to do it.

1.1 How to apply for computing time on Apolo

The first contact to evaluate the viability of computing in Apolo is by means of an email to apolo@eafit.edu.co or a call +57 (4) 2619592. An appointment will be scheduled via physical or virtual to assess the needs of the user. At Apolo, researchers from the Universidad EAFIT can compute for free. Other universities, academic institutions or industrial companies can compute, applying costs on the **core-hour**, **gpu-hour** and **TiB-month** basis.

1.1.1 How to estimate time to use on Apolo

Sometimes, it can be difficult to estimate the time it will take to compute, depending on many factors, including the order of the algorithm, processor clock speed, memory speed and the buses speed that interconnects them. Sometimes it has to be estimated from experience.

To have a basis on which to estimate the cost of using Apolo, we have a unit known as the **core-hour**. The definition of this unit is: having **one hour** of computing in a **core at 100% usage**. You should estimate how long it would take if your work were serial and multiply the **core-hour** price which could be got from a quote writing to apolo@eafit.edu.co.

1.2 Requirements on the software

The following items describes the ideal technical conditions of the software to compute on Apolo's supercomputers.

1. Scientific Software
2. Linux compatible
3. Already parallel or parallelizable

4. Distributed (Highly recommended)

If you answer affirmatively the previous questions then you are a good candidate to compute in Apolo. The next step is to schedule an appointment to make the following verifications:

1. Your software is installed and ready to use it or we have to install it.
2. Your software is optimized and configured in a proper way or we have to reconfigure it.
3. You are already trained or need to be guided to use Apolo in a proper way.

Once all this things were analyzed, we can proceed to create the account on the clusters.

1.3 Get an account

To use Apolo you will need to have the following items:

- A VPN account, created by the system administrators. In the following section you will find a tutorial to configure the VPN account in different operating systems.
- An account on Apolo's clusters, created by system administrators, this account is personal and non-transferable.
- Test if your accounts are working and launch a job!

In order to get an account, you need to send an email to apolo@eafit.edu.co with the following information:

- User
- Mail
- Name
- Last Name
- Cell Phone Number
- Working or Investigation Group
- Type of user (undergrad, master, phd, research)
- Main usage of Apolo (CPU, memory, GPU, disk)
- Operating System of the PC you will use to access to Apolo
- Applications that you will use in Apolo.
- Possible times for the introductory meeting

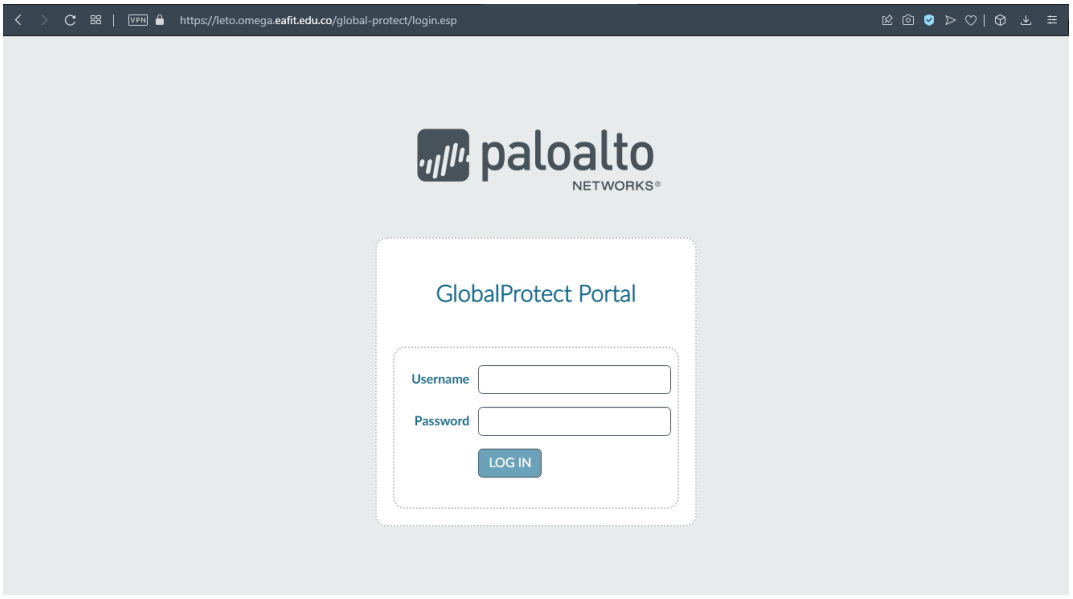
1.4 Configure VPN

A Virtual Private Network or VPN is a mechanism that allows to make a secure channel across insecure public networks like Internet. It creates a cyphered tunnel between your computer and the network segment of Apolo's supercomputers. The use of VPN is important because it prevents intruders from seeing our users' network traffic and even attacking our servers. Given these conditions, the use of the VPN is mandatory even though all traffic to and from Apolo is encrypted by means of the ssh and sftp protocol. The following subsections explains how to configure the Apolo's VPN in your particular operating system.

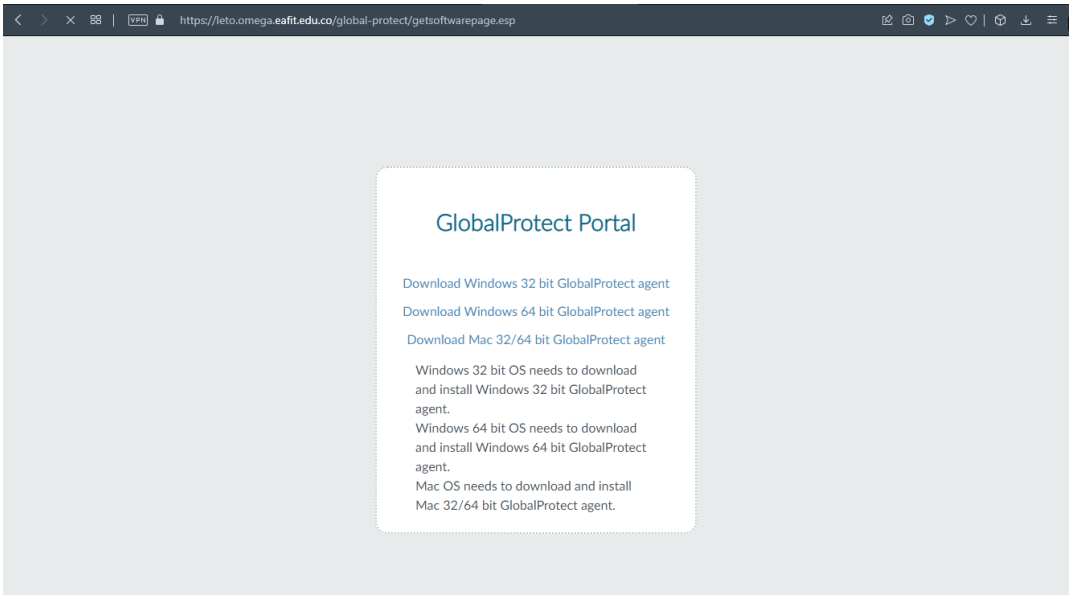
1.4.1 Windows and Mac OSX

To configure the VPN on Windows and Mac OS X systems, you must follow exactly the same procedure. Here are the steps to download and configure the VPN client:

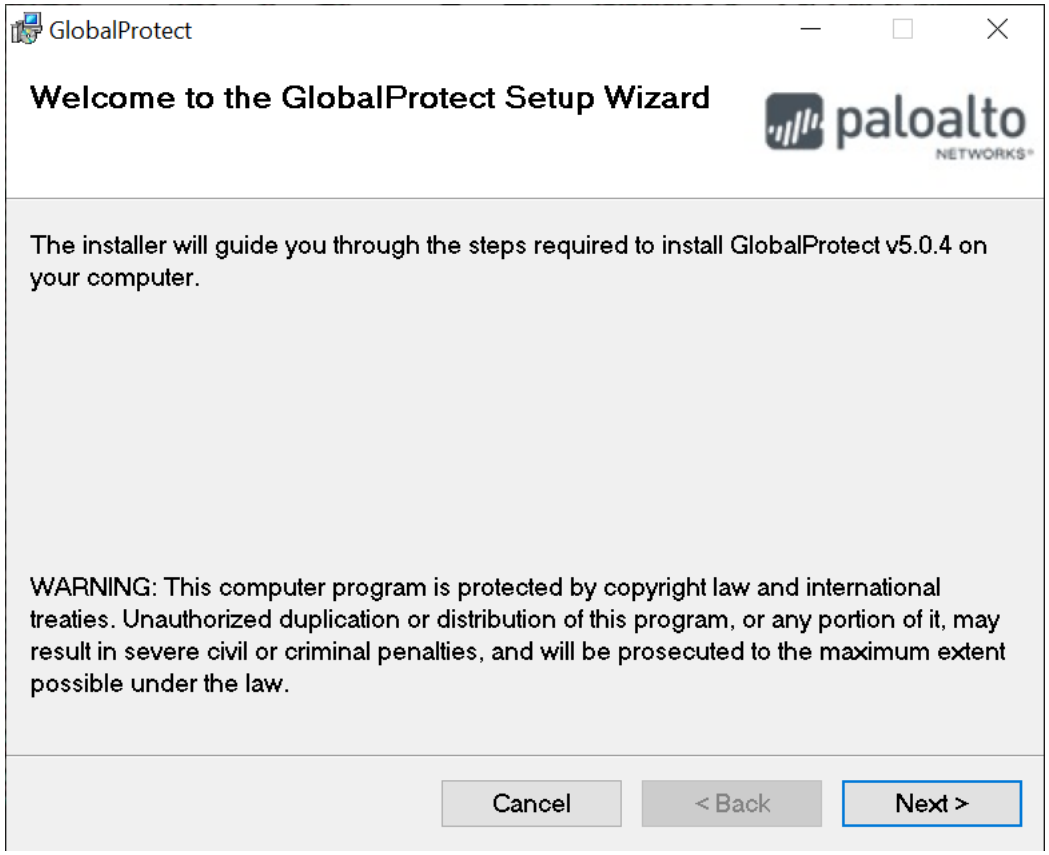
1. Open your favorite browser and go to https://leto.omega.eafit.edu.co and log in with your **username** and **password** given by EAFIT or Apolo's staff



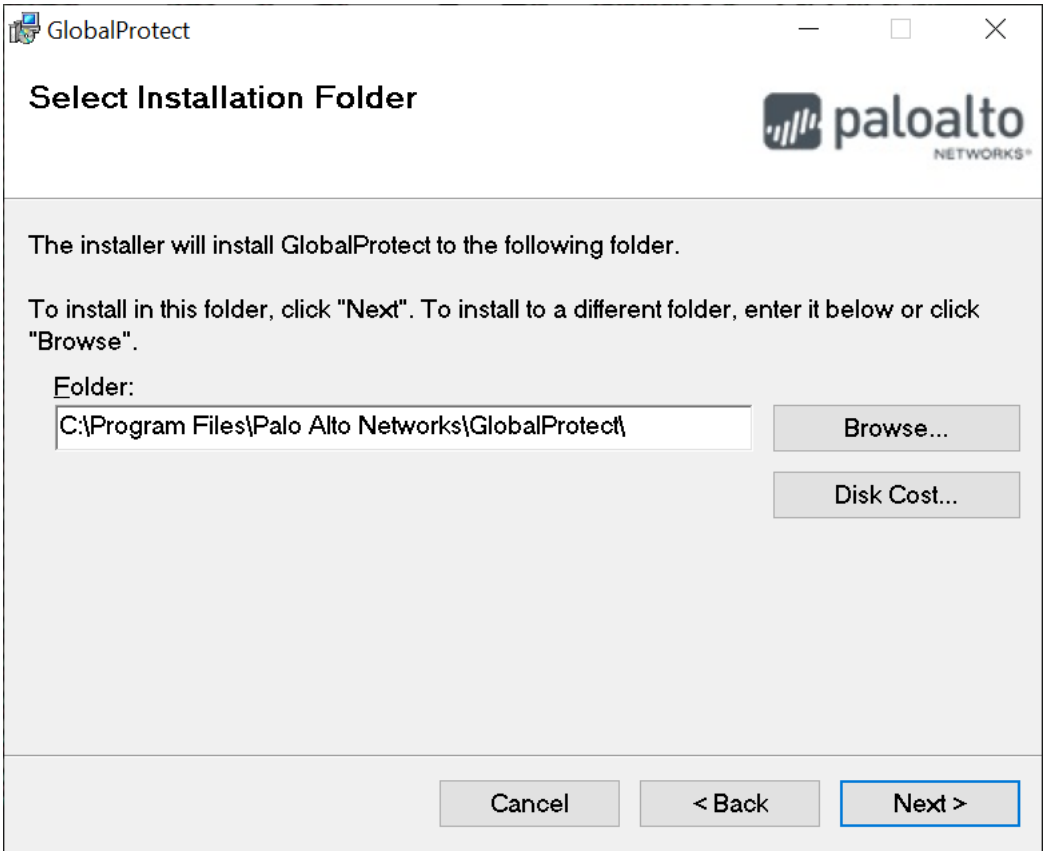
2. Download and install the version of Global Protect client according to your operating system



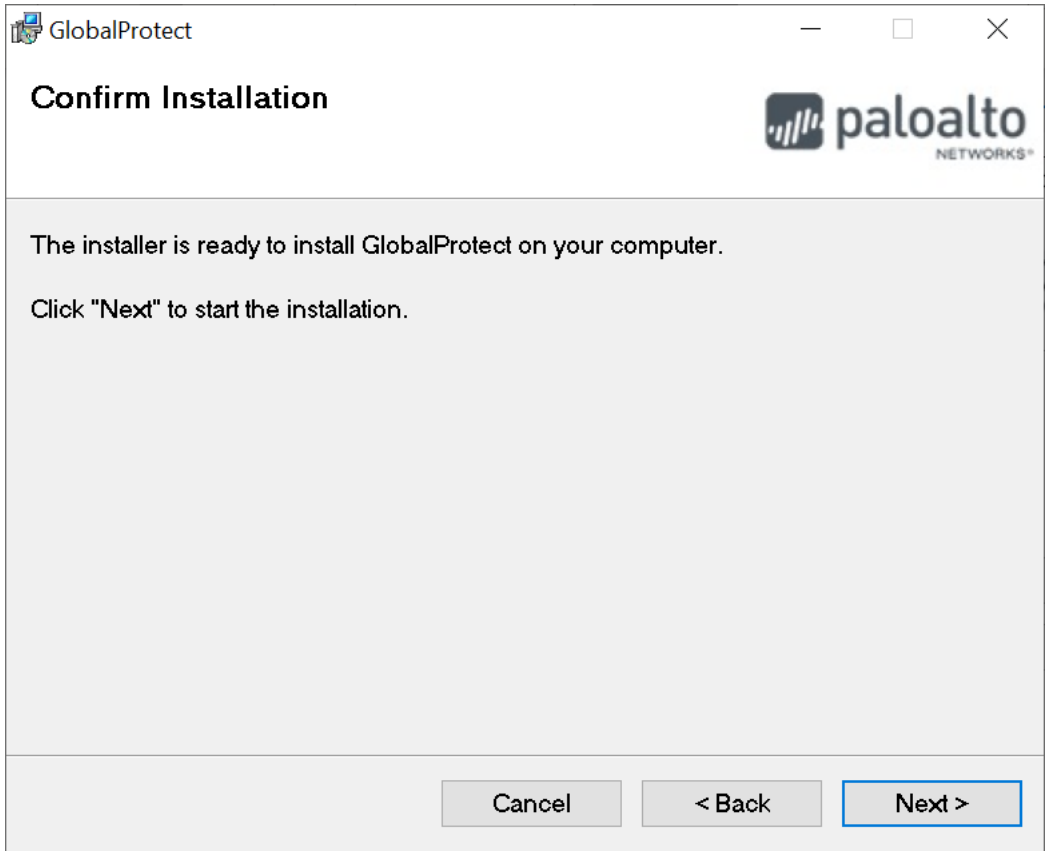
3. Install the Global Protect application



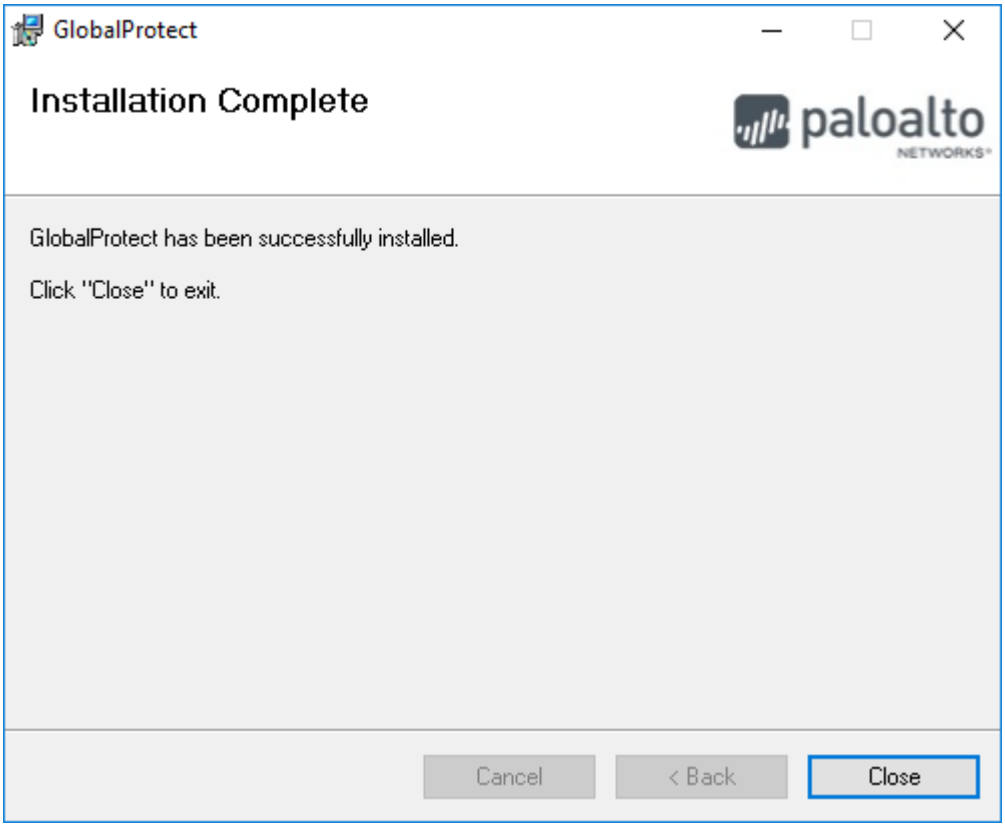
4. Choose where to install it, take into account your permissions on the system



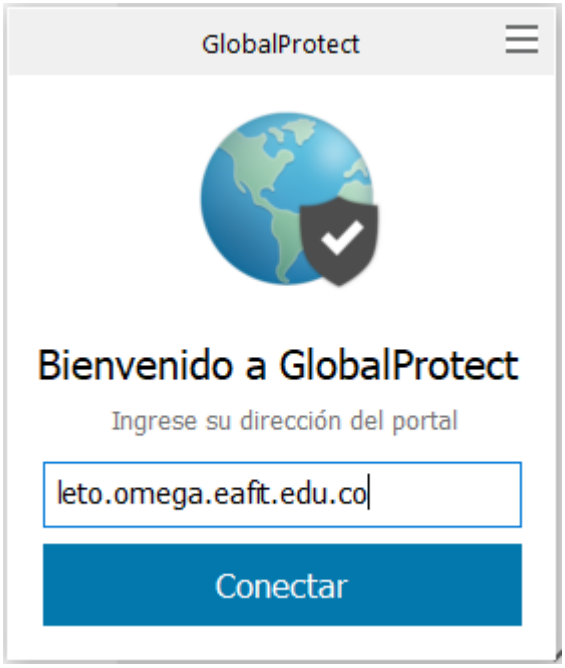
5. Finish installation



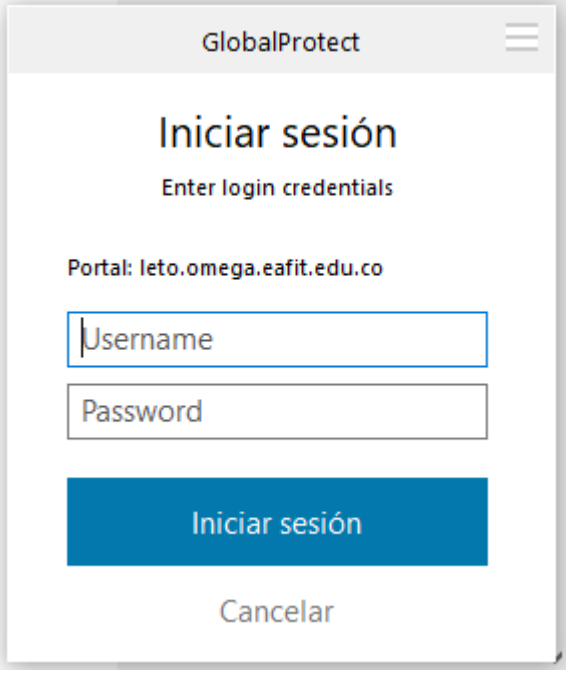
6. Close installation



7. Launch the Global Protect application and fill the portal input **Portal** with *leto.omega.eafit.edu.co*



8. Fill the fields with the following information:



- **Username:** The username assigned by the Apolo's system administrator.
- **Password:** The password used to log in to the clusters.

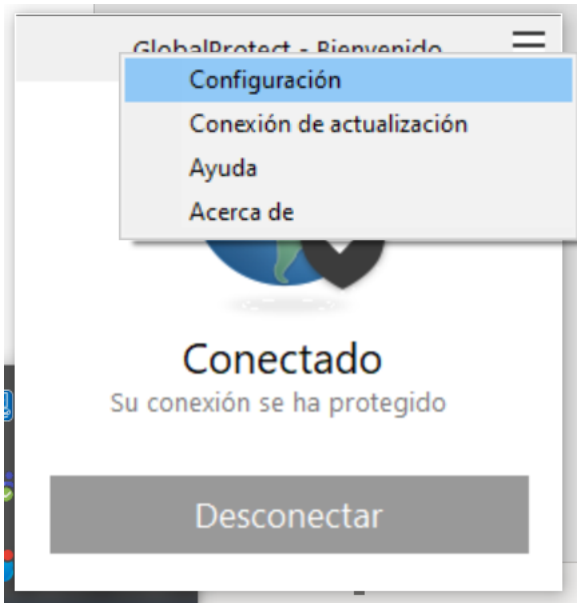
> **Warning:** The password sent to your email is one-time password, the first time you login to our clusters the system will ask you for changing the password, after that the new password will be used to log in to the VPN.

> **Warning:** Remember your password will expire every three (3) months.

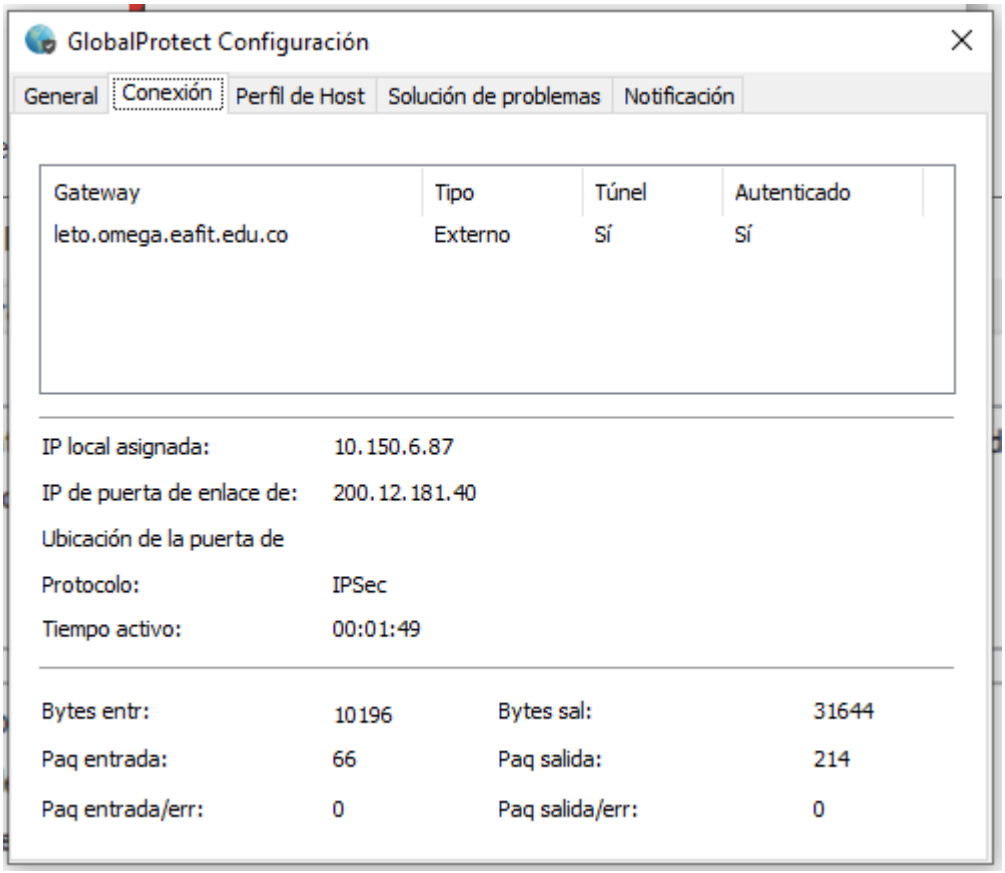
9. Once connected to the VPN, go to the Taskbar as you see in the image

10. You will see the word Connected as shown in the image

11. You can see some network parameter in the Details tab, go to the menu and click on Configuration



12. See the network parameters



Warning: You must login for the next 5 hours or the account will be deactivated.

Using Vagrant to set up a VirtualMachine

If for any reason you can't connect on Mac or Windows using the "Global Protect" application, you can use vagrant to set up a linux virtual machine and connect to the Apolo network through it.

Note: This is something to use as last resource, it may make things a little complex. We highly recommend to use the "Global Protect" application and contact the Apolo staff for any doubts or problems but they might be a situation where the application has problem with a specific version of Windows or Mac and we'll recommend you to use this method.

Explaining each tool is out of the scope of this guide. We are going to show a step by step on how to install and use each of the tool to connect to the Apolo network. For more information we recommend to look at the apropiated doc of each tool:

- https://www.vagrantup.com/docs
- https://www.virtualbox.org/wiki/Documentation

1. Open your favorite browser, go to https://www.virtualbox.org/wiki/Downloads and download the package appropiated to your platform.

2. Open your favorite browser, go to https://www.vagrantup.com/downloads.html and download the package appropiated to your platform.

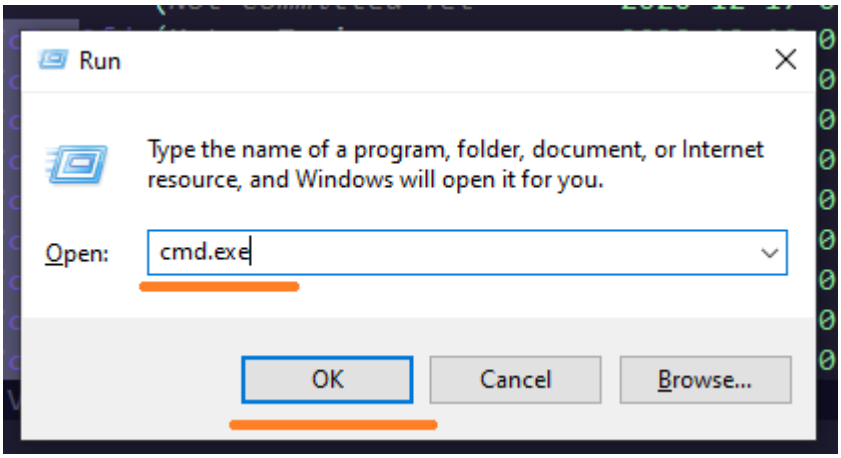
3. After installing both applications, copy and paste the following text on a file with the name *Vagrantfile* and save it to any directory, for the rest of the guide we're going to suppose the *Vagrantfile* was stored at the Desktop.

```
# -*- mode: ruby -*-
# vi: set ft=ruby :
Vagrant.configure("2") do |config|
  # boxes at https://vagrantcloud.com/search.
  config.vm.box = "ubuntu/focal"

  config.vm.provision "shell", inline: <<-SHELL
    apt-get update
    apt-get install -y vpnc openssh
  SHELL
end
```

4. Also copy the file with the extension **.pcf** given to you by the Apolo staff to the same directory where the **Vagrantfile** is.

> **Note:** If you don't have any file with that extension please write to the staff asking for the file needed on linux to connect to the VPN. You'll need it in the next steps.

5. Then you need to open the terminal and navigate to the desktop directory. On Windows you can open the terminal pressing the *Windows Key* + *R* then typing *cmd.exe* and clicking *OK*.



6. While being on the same directory where the Vagrantfile is, start the virtual machine.

```
$ vagrant up
```

> **Note:** If you are running MacOS and this command failed please check the *Security & Privacy* panel on *System Preference* and allow the *System software from Oracle America was blocked from loading* More info at https://appuals.com/virtualbox-the-installation-failed-on-mac/

7. Then connect to the virtual machine.

```
$ vagrant ssh
```

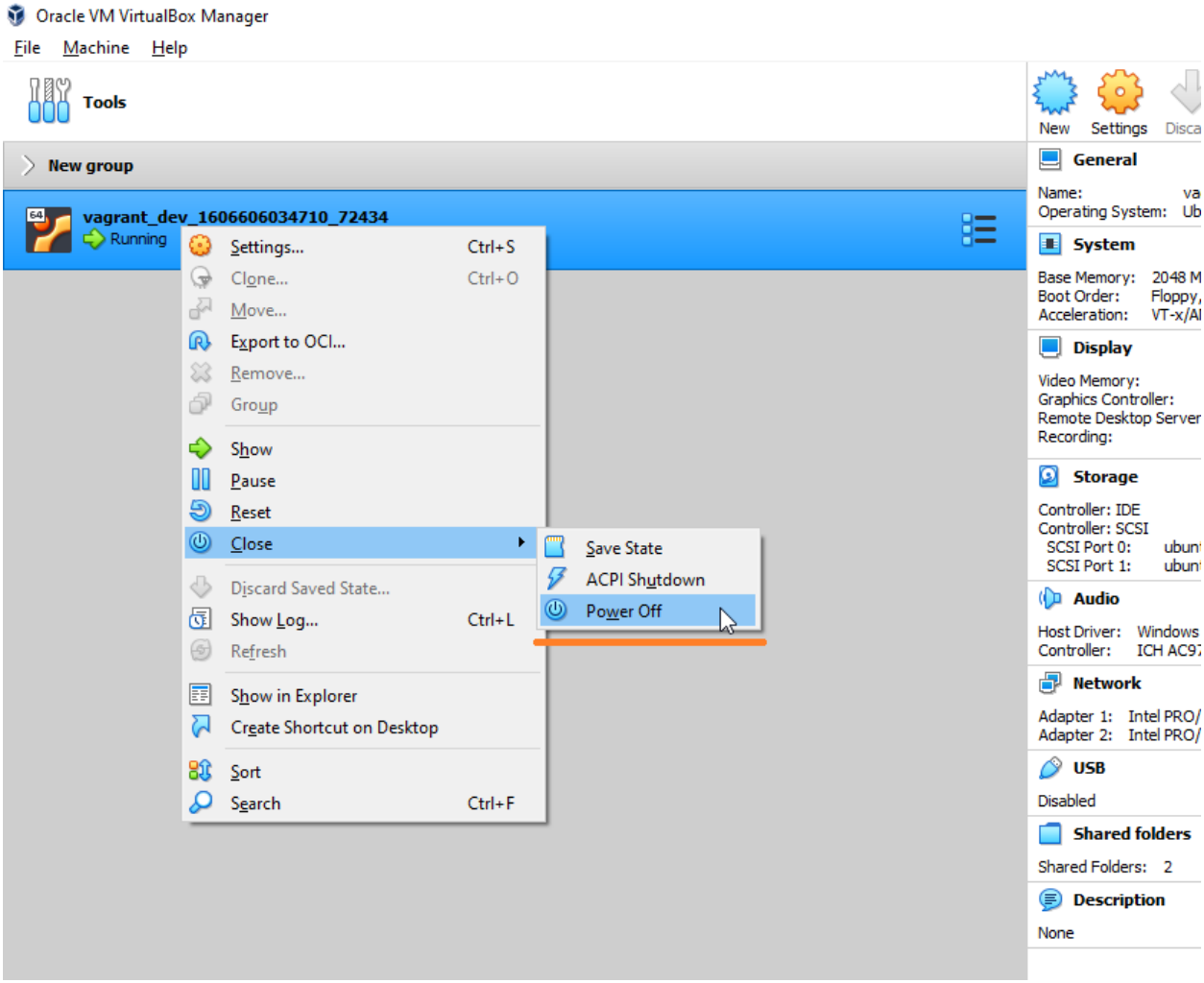
8. Now that you are connected to the virtual machine follow the steps on the Connect through the terminal section.

> **Note:** You can access the same directory where you *Vagrantfile* is within the virtual machine at the path */vagrant*.

To turn off the virtual machine, you can do it from the terminal with.

```
$ vagrant halt
```

Or through the *VirtualBox GUI*.



1.4.2 Linux

Note: Depending on your distribution this procedure could change.

To configure the VPN on Linux, you have to use your package manager to install a Cisco Compatible VPN client. The most common client is vpnc, which is embedded on a set of scripts. Usually, the package with these scripts is called `vpnc`.

Connect through a GUI

If you use Gnome or a Gnome compatible window manager you should install the `network-manager-vpnc-gnome` and `vpnc` packages. If you use KDE or a KDE compatible window manager you'll need to install the `plasma-nm` and `vpnc` packages instead.

Listing 1: **Tested on Ubuntu 18.04 and 20.04**

```
$ sudo apt search vpnc
[sudo] password for user:
kvpnc/bionic 0.9.6a-4build1 amd64
frontend to VPN clients

kvpnc-dbg/bionic 0.9.6a-4build1 amd64
frontend to VPN clients - debugging symbols

network-manager-vpnc/bionic-updates,bionic-security,now 1.2.4-6ubuntu0.1 amd64
network management framework (VPNC plugin core)

network-manager-vpnc-gnome/bionic-updates,bionic-security,now 1.2.4-6ubuntu0.1 amd64
network management framework (VPNC plugin GNOME GUI)

vpnc/bionic,now 0.5.3r550-3 amd64
Cisco-compatible VPN client

vpnc-scripts/bionic,bionic,now 0.1~git20171005-1 all
Network configuration scripts for VPNC and OpenConnect
```

```
$ sudo apt install vpnc network-manager-vpnc-gnome
```

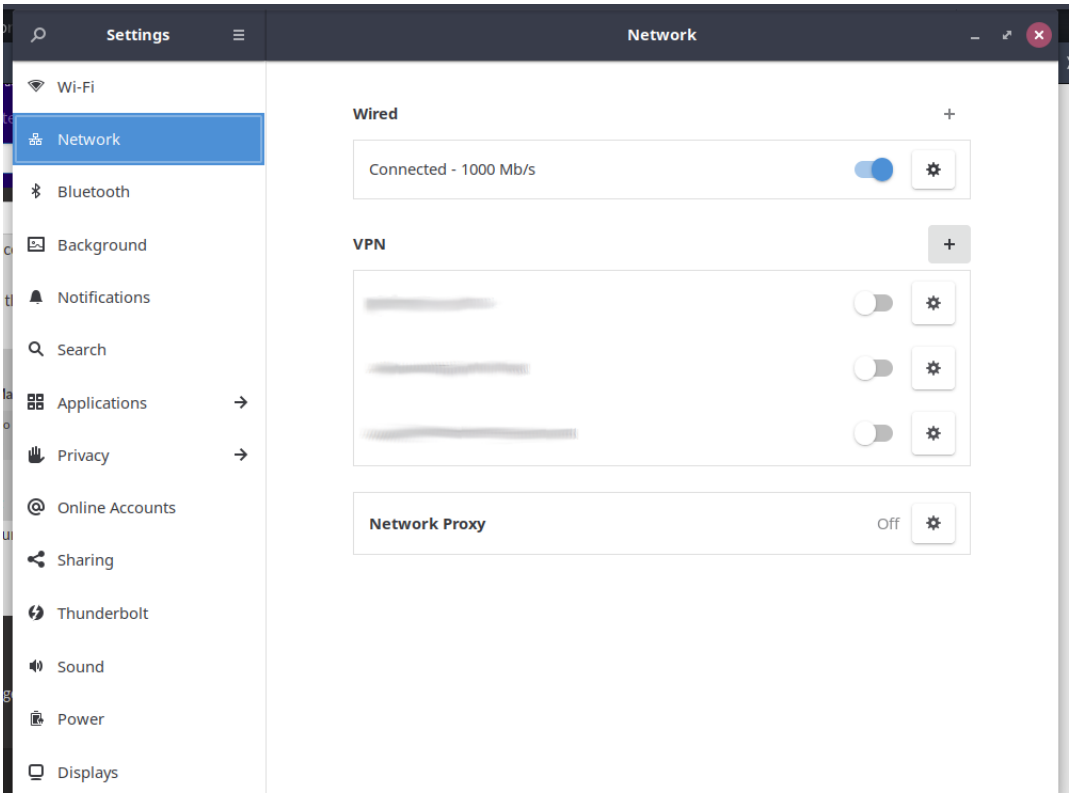
Once the correct package is installed according to your distribution, you can proceed to configure the VPN client.

Warning: It is strongly recommended to log out and log in before to start the following steps because there are some cases where the VPN connection does not work until log out or reboot is performed after the package installation.

Warning: The following procedure may vary depending on the package installed. We are going to use the configuration for network-manager-vpnc-gnome due this is the most common package on usual Linux distributions.

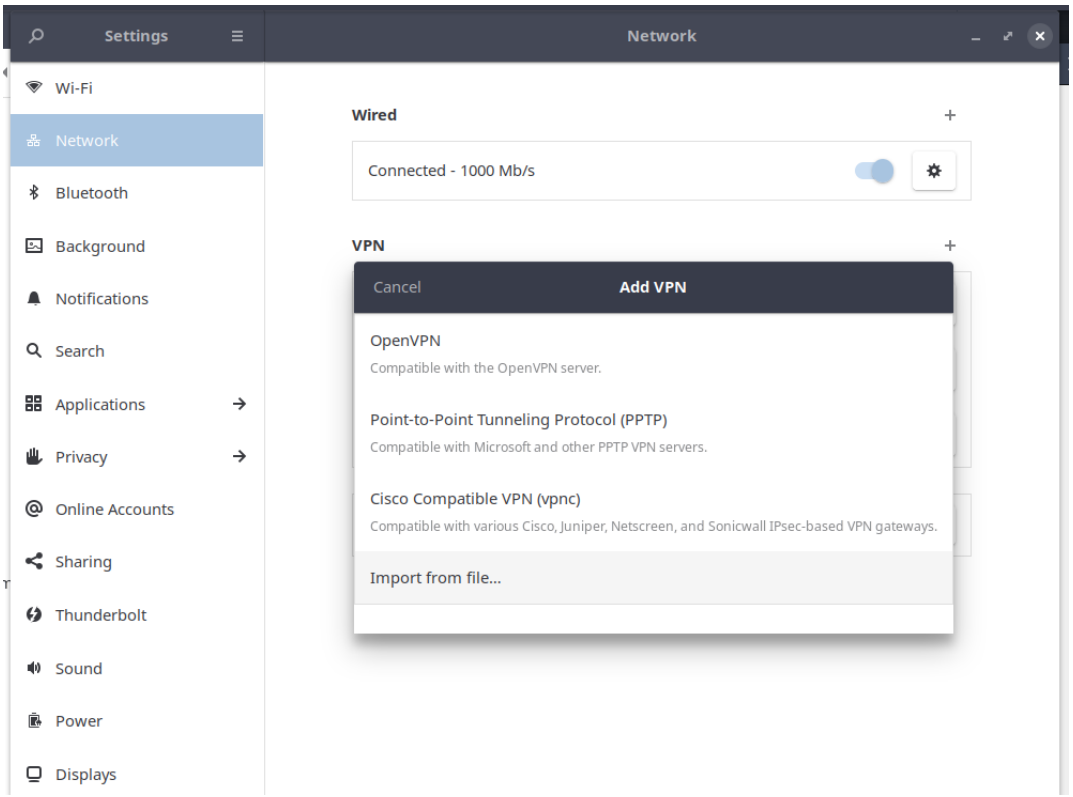
1. Open the main menu and System Settings.

2. Look for Network item and click on the plus symbol to add a new connection.

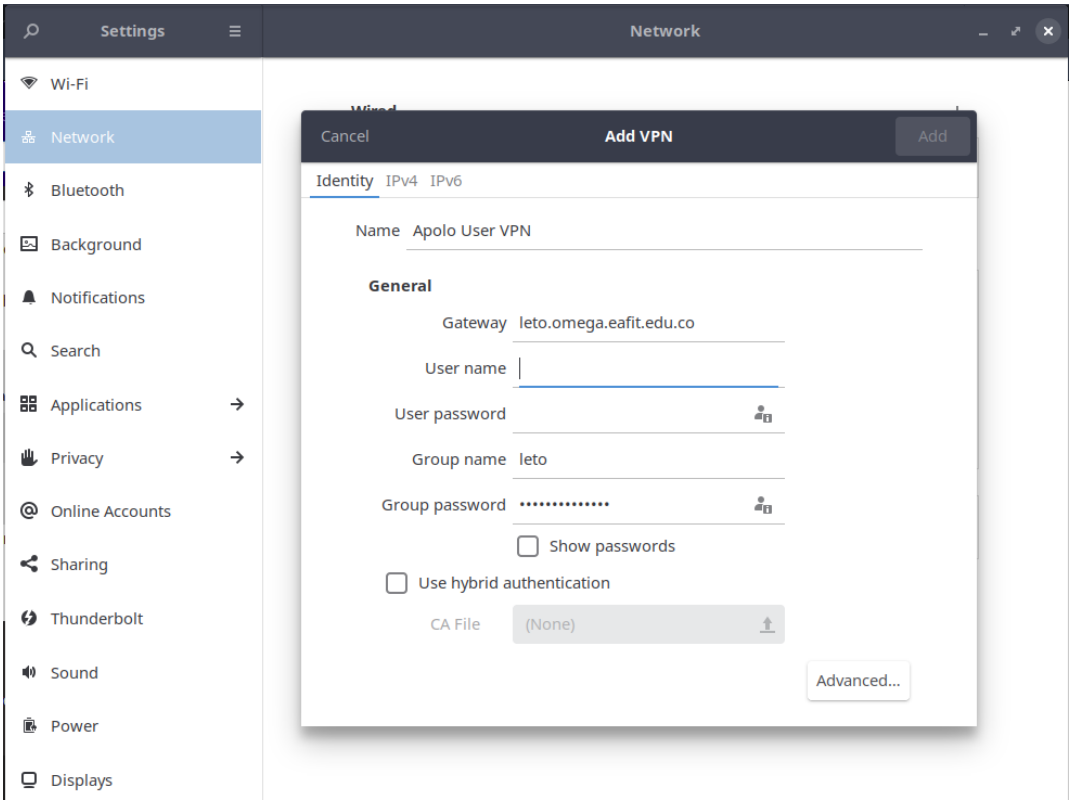


3. Choose Import from file. . .

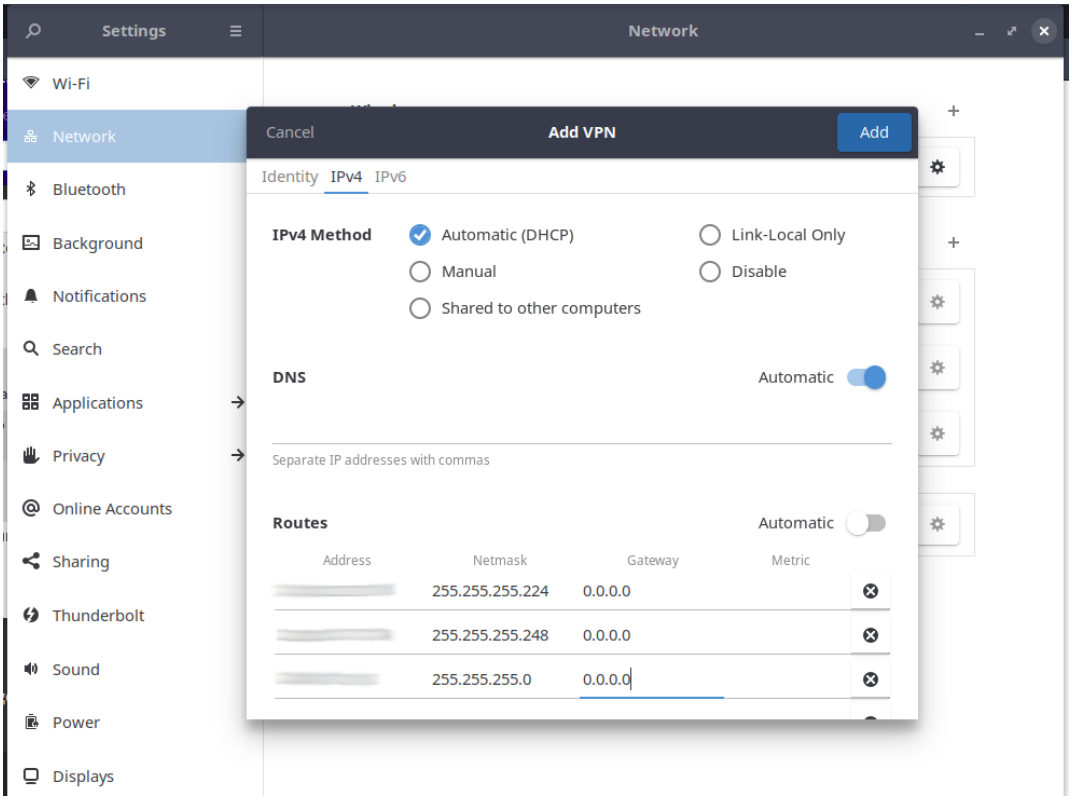
Note: The VPN file will be provided by the system administrator, please request it before to continue with this guide.



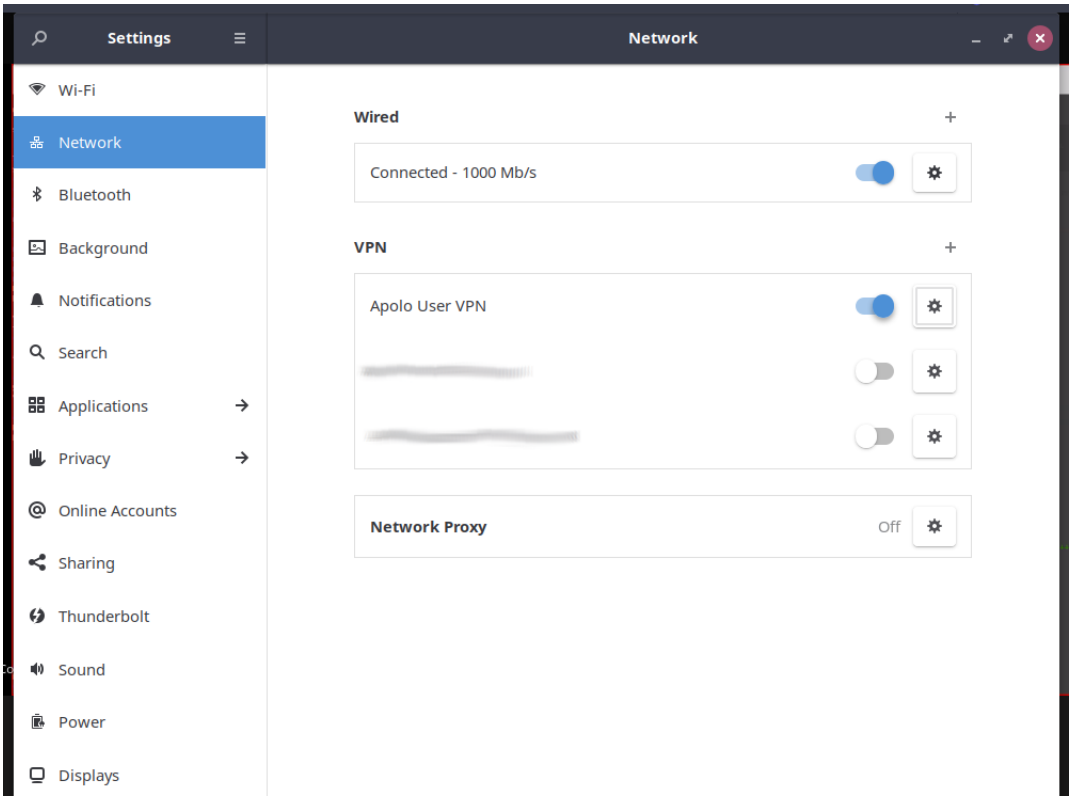
4. Once the file has been imported you just need to add your username and password provided by the administrator. **Note that the group password is filled automatically by the imported file**.



5. On IPv4 options on the left panel, please add the following route and apply the configuration.

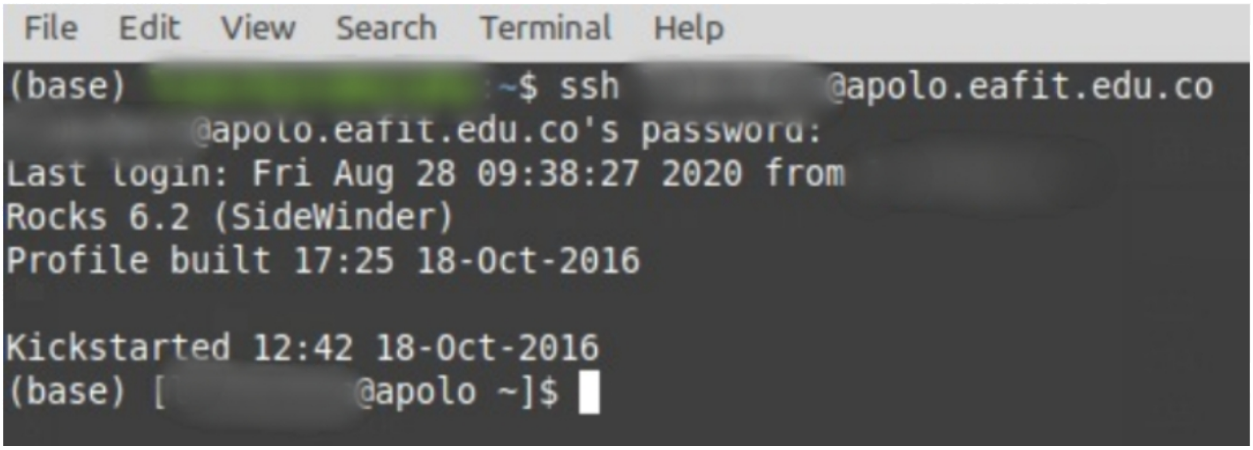


6. Now you can connect to the cluster through the VPN.



7. Once you are connected to the VPN, access Apolo via SSH with the following command and type your password:

```
$ ssh <username>@apolo.eafit.edu.co
```



Note: Remember that the first time it will be necessary to change the password assigned by a new one that must contain a combination of lower case, upper case, numbers and special characters. It must have a minimum of 8 characters.

Connect through the terminal

Note: Depending on your distribution some extra packages might be needed.

To be able to connect to the VPN through the terminal, the `vpnc` package is needed.

Listing 2: **Tested on Ubuntu 20.04**

```
$ sudo apt search vpnc

network-manager-vpnc/focal,now 1.2.6-2 amd64 [installed,automatic]
  network management framework (VPNC plugin core)

network-manager-vpnc-gnome/focal,now 1.2.6-2 amd64 [installed]
  network management framework (VPNC plugin GNOME GUI)

vpnc/focal,now 0.5.3r550-3.1 amd64 [installed]
  Cisco-compatible VPN client

vpnc-scripts/focal,focal,now 0.1~git20190117-1 all [installed,automatic]
  Network configuration scripts for VPNC and OpenConnect
```

```
$ sudo apt install vpnc
```

After installing the package, you'll need to convert the .pcf file to a .conf file.

```
$ pcf2vpnc ./Apolo-vpn-file.pcf Apolo-vpn-file.conf
```

Once you have the .conf file, you'll need to change the following line

```
Xauth username jdpinedac
```

And replace 'jdpinedac' for the username given to you by the Apolo staff. After that, you can initiate the VPN connection using the vpnc program.

```
$ sudo vpnc ./Apolo-vpn-file.conf
Enter password for username@leto.omega.eafit.edu.co:
```

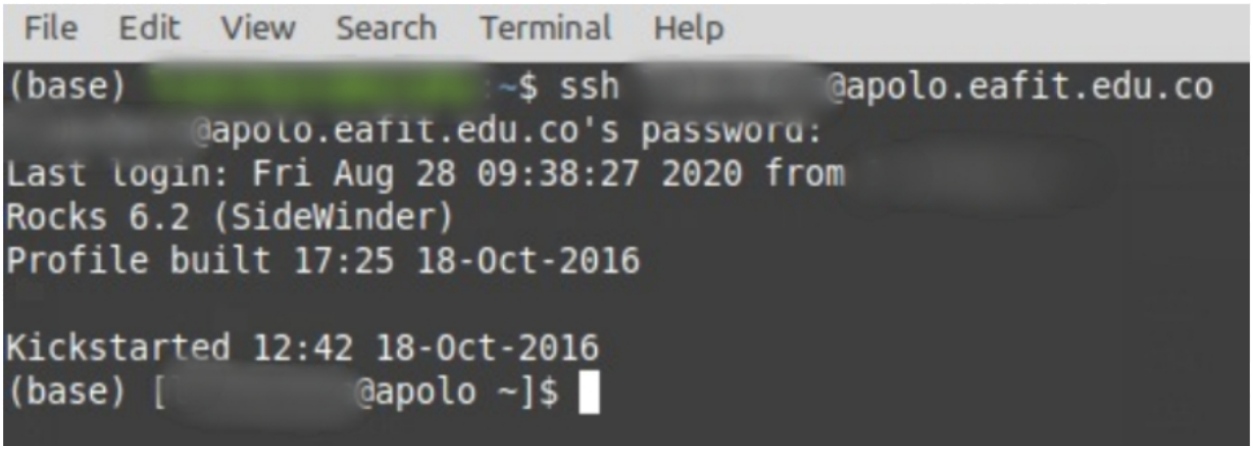
If the given password was correct It'll start the VPN service on the background.

To stop the VPN, just run:

```
$ sudo vpnc-disconnect
Terminating vpnc daemon (pid: 171941)
```

Once you are connected to the VPN, access Apolo via SSH with the following command and type your password:

```
$ ssh <username>@apolo.eafit.edu.co
```



Note: Remember that the first time it will be necessary to change the password assigned by a new one that must contain a combination of lower case, upper case, numbers and special characters. It must have a minimum of 8 characters.

1.4.3 Troubleshooting

See also:

You can find a Global Protect example for windows or mac configuration on the following screencast:

See also:

Issue: After installing or upgrading the Mac GlobalProtect client, the client never connects and just "spins".

Solution:

1. Click the Apple icon in the upper left hand corner, then click 'System Preferences', then 'Security'.

2. Look for a message at the bottom of the window stating "System software from developer was blocked from loading."
3. To allow the software to load again, click the Allow button.

If that doesn't work, try the following: https://docs.paloaltonetworks.com/globalprotect/4-0/globalprotect-agent-user-guide/globalprotect-agent-for-mac/remove-the-globalprotect-enforcer-kernel-extension

See also:

Sometimes, When you close the mac with the VPN open, there may be problems in re-establishing the connection to the VPN, so it is suggested that you close the program and reopen it.

1.5 Educational Resources

For the scientific computing center it is very important that its users have the right knowledge to make good use of resources. For this reason, the center offers workshops, conferences, and consulting services on the tools and skills required in scientific computing. In addition to this, in this section you will find some online resources where you can find the basic knowledge and skills to make good use of all the resources of the center.

1.5.1 Basic Knowledge & Skills

- The Unix Shell - Software Carpentry
- La terminal de Unix - Software Carpentry (Español)
- Version control with Git - Software Carpentry
- Control de versiones con Git - Software Carpentry (Spanish)
- Basic lab skills for research computing on Software Carpentry
- Bash Programming - Introduction HOW-TO - The Linux Documentation Project
- Principiante Emacs (Español)
- Vim Tutorial - Open VIM
- Cluster Design

1.5.2 Parallel Computing and High Performance Computing

- Introduction to parallel computing - Lawrence Livermore National Laboratory

How Science Drives Architecture - NERSC

Many of the algorithms or methods that require High Performance Computing (HPC) will require different types of resources to operate, such as specific libraries, hardware and software.

This section will show some methods used in various scientific fields that need HPC to find a solution and their hardware requirements to work properly. In other words, we will show the scientific nature of some problems and its technological solution[1].

[1] https://crd.lbl.gov/assets/pubs_presos/CDS/ATG/WassermanSOTON.pdf

Table I: Some Methods in Science

Here we can see different types of algorithms or methods that can be used in some science fields or some applications of these. With this information it is possible to know the different cases that can be used to solve problems in a scientific field, also knowing the nature of the algorithm, we can know which would be the libraries to use.

Algorithm / Science areas	Dense linear algebra	Sparse linear algebra	Spectral Methods (FFTs)	Particle Methods	Structured Grids	Unstructured or AMR Grids	Data Intensive
Accelerator Science		X	X	X	X	X	
Astrophysics	X	X	X	X	X	X	X
Chemistry	X	X	X	X			X
Climate			X		X	X	X
Combustion					X	X	X
Fusion	X	X		X	X	X	X
Lattice Gauge		X	X	X	X		
Material Science	X		X	X	X		

This table shows the different methods that can be used to solve specific problems in the listed areas of science.

For example, for problems that arise in Acceleration Science, a research tool that helps reveal the innermost workings of matter[2] , it uses methods such as Sparse linear algebra[4], Spectral and Particle Methods, and Structured and Unstructured Grids.

Astrophysics is the development and study of physics applied to astronomy and the problems that can arise in this area are so varied that their possible solution can cover all the methods of the table, from the amount of operations to be performed to the handling of large volumes of data.

In this table we can see which areas of science face which kind of problems, then we can see which areas like Chemistry face problems whose solution may be among the methods of Dense Linear Algebra[3], Sparse Linear Algebra, Spectral Methods[5], or Particle Methods[6]. The climate area face Spectral Methods, Structured[7] and Unstructured[8] Grids, and Data Intensive[9] problems. The same can be seen with the other areas of science found in the table.

Note: For a better understanding of the problems that each of the listed science areas may face, you can see the references.

Table II: Machine Requirements

In the previous table we can see different methods to solve various types of problems, according to the science area and in the following one we will see what is required (in terms of hardware) to carry out these methods.

[2] https://stfc.ukri.org/research/accelerator-science/
[4] https://patterns.eecs.berkeley.edu/?page_id=202
[3] https://patterns.eecs.berkeley.edu/?page_id=158
[5] https://en.wikipedia.org/wiki/Spectral_method
[6] https://en.wikipedia.org/wiki/Particle_method
[7] https://en.wikipedia.org/wiki/Regular_grid
[8] https://en.wikipedia.org/wiki/Unstructured_grid
[9] http://opencirrus.org/data-intensive-computing-problems/

Algorithm / Science areas	Dense linear algebra	Sparse linear algebra	Spectral Methods (FFT)s	Particle Methods	Structured Grids	Unstructured or AMR Grids	Data Intensive
Accelerator Science		High performance memory system		High performance memory system			
Astrophysics			High bisection bandwidth			Low latency, efficient gather /scatter	Storage, Network Infrastructure
Chemistry	High Flop/s rate				High flop/s rate		
Climate							
Combustion							
Fusion							
Lattice Gauge							
Material Science							

Then we can see that for linear Dense algebra and Structured Grids problems, it is necessary to have a high rate of floating point operations per second (Flop/s).

For the resolution of Sparse linear algebra problems, a high performance memory system is required as well as a high bandwidth must be had to solve Spectral Methods (FFT)s problems

Finally, we can see that for Unstructured or AMR (Adaptive mesh refinement) Grids problems, low latency and efficient memory addressing are needed. For Data intensive problems, as the name suggests, a good storage and network infrastructure is needed to handle large amounts of data.

References

- Supercomputing - Future Learn
- High Performance Computing in the Cloud - Future Learn

Parallel Programming

- Fundamentals of Parallelism on Intel Architecture - Coursera
- Parallel Programming - Coursera
- Introduction to OpenMP - Intel
- A Hands on Introduction to OpenMP - OpenMP
- MPI Tutorial
- Designing and Building Parallel Programs
- OpenACC Nvidia Courses

Job Management

- Slurm Tutorials
- Slurm Quick Start Tutorial - C.E.C.I.

1.5.3 Bibliography

- Encyclopedia of Parallel Computing. Padua, David (Ed.). Springer. 2011. ISBN: 978-0-387-09765-7
- Topics in Parallel and Distributed Computing: Introducing Concurrency in Undergraduate Courses. Sushil K Prasad, Anshul Gupta, Arnold L Rosenberg, Alan Sussman, Charles C Weems. Ed. Morgan Kaufmann. 2015. ISBN: 0128038993
- Python Parallel Programming Cookbook. Giancarlo Zaccone. Ed. Packt Publishing. 2015. ISBN: 1785289586
- Structured Parallel Programming: Patterns for Efficient Computation. Michael McCool and James Reinders and Arch Robinson. Ed. Morgan Kaufmann. 2012. ISBN: 0124159931
- Linux in a Nutshell: A Desktop Quick Reference. Ellen Siever and Stephen Figgins and Robert Love and Arnold Robbins Ed. O'Reilly Media. 2009. ISBN: 0596154488
- bash Cookbook: Solutions and Examples for bash Users. Ed. O'Reilly Media. 2017. ISBN: 0596526784
- Beginning the Linux Command Line. Sander van Vugt. Ed. Apress. 2009. ISBN: 1430218894
- High Performance Computing - Modern Systems And Practices. Thomas Sterling, Matthew Anderson, Maciej Brodowicz. Ed. Morgan Kaufmann. 2018. ISBN: 978-0124201583

1.6 Apolo User Tools

Apolo staff has implemented a repository in GitHub that contains some scripts, tools and aliases to help our users with some needs they have mainly about the system or SLURM. Also, it is a public repository and users could add new aliases or commands to it.

1.6.1 Apolo User Tools

Repository structure

The repository could be found at https://github.com/eafit-apolo/apolo-user-tools

The repo contains two main directories: bash-tools and src

- **bash-tools:** Contains aliases or functions that are written in Bash. If you plan to add aliases put it in the file **default-aliases** in the section related with purpose of the alias. If more complex commands are needed, include it using another file in the same directory with a clear title.
- **src:** Contains directories with more complex commands that are written in any language, we specially encourage the use of Python. The directories inside src are application-specific, e.g: slurm, ansible, file-system, etc.
  - **application-specific:** These directories contain the different scripts that will become available commands to our users. These scripts could be in single files or directories. Commands in the same application-specific directory share the same environnement and language, libraries or other requisites. Each application-specific directory contains a single file that specifies these requisites, feel free to modify it.

Available Commands

- **smyqueue:** This command prints the status of the current jobs in slurm queue of the user who executed the command.
- **sncores:** This command print the total number of cores in the system and groups it by its status (Allocated, idle and other).

- **squeue-stats:** This command prints statistics about current waiting and active jobs.
- **sqos-usage:** This command is used to check how many hours a group has used of the total available in the qos of the group (This command applies just for paid users).
- **du-home:** This command prints the total used space in disk of the user who executed the command.

  Authors

  - Juan David Arcila Moreno <jarcil13@eafit.edu.co>
  - Juan Diego Ocampo García <jocamp18@eafit.edu.co>

1.7 Get Help

Always you can get help in all Apolo's related topics. You can call to +57 (4) 2619592 or write an email to apolo@eafit.edu.co

CHAPTER 2

Supercomputers

2.1 Apolo I

2.2 Apolo II

2.3 Cronos

CHAPTER 3

Software

3.1 Virtualization

This entry contains relevant information related to virtualization tools.

3.1.1 VMware ESXi

OVF TOOL

VMware OVF Tool is an utility that allows the system administrator to package or unpackage virtual machines using the Open Virtualization Format from the command line.[1]

In this entry we will cover the steps needed to package a VM controlled by a VMware host, by connecting to the host from your PC.

Versions

OVF Tool 4.3.0

Basic information

- **Release date:** 17 APR 2018.
- **Official documentation: https://www.vmware.com/support/developer/ovf/**

[1] OVF Tool documentation. Retrieved September 21, 2018, from https://www.vmware.com/support/developer/ovf/

Installation

Note: For Windows systems you need the Visual C++ Redistributable for Visual Studio 2015.

Do the following process in your machine.

You need a VMware user account in order to download this tool. Once you are registered you can download this tool from here.

To install on a **linux system**:

1. Give the necessary permissions to the downloaded file:

```
$ chmod +x VMware-ovftool-4.0.0-2301625-lin.x86_64.bundle
```

2. Execute the file with superuser privileges:

```
$ sudo ./VMware-ovftool-4.0.0-2301625-lin.x86_64.bundle
```

Note: If you are on Windows or OS X install it the way you normally would.

Create a OVF from a virtual machine

1. Find the exact name of the virtual machine that you want to package by using the following command:

```
$ ovftool vi://root@<host's-ip>/
```

After you enter the root password it will raise an error message like the following:

```
Error: Found wrong kind of object (ResourcePool). Possible completions are:
  Virtual-machine1
  Virtual-machine2
  Virtual-machine3
```

2. In order to export the virtual machine:

Note: Ensure that no iso file is mounted to the VM. Otherwise a copy of it will be created.

```
$ ovftool vi://root@<host's-ip>/<vm_name> <exportFileName>.ovf
```

You may be prompted to use the following option: **–allowExtraConfig**:

```
$ ovftool --allowExtraConfig vi://root@<host's-ip>/<vm_name> <exportFileName>.
ovf
```

Note: Is safe to use "–allowExtraConfig". According to *ovftool* official manpage it specifies "whether we allow ExtraConfig options in white list. These options are safe as we have a white list to filter out the low-level and potential unsafe options on the VM."

Authors

- Vincent Alejandro Arcila Larrea (vaarcilal@eafit.edu.co).

- Andrés Felipe Zapata Palacio (azapat47@eafit.edu.co).

3.2 Programming Languages

This entry contains all relevant steps to build, configure and run every programming languages available in the supercomputers (*Apolo II* and *Cronos*).

3.2.1 MATLAB

MATLAB[1] (matrix laboratory) is a multi-paradigm numerical computing environment and proprietary programming language developed by MathWorks.

MATLAB allows matrix manipulations, plotting of functions and data, implementation of algorithms, creation of user interfaces, and interfacing with programs written in other languages, including C, C++, C#, Java, Fortran and Python.

Although MATLAB is intended primarily for numerical computing, an optional toolbox uses the MuPAD symbolic engine, allowing access to symbolic computing abilities. An additional package, Simulink, adds graphical multi-domain simulation and model-based design for dynamic and embedded systems.

MATLAB - R2018a

Basic information

- **Deploy date:** 9 July 2018
- **Official Website:** https://www.mathworks.com
- **License:** Proprietary commercial software (Only for EAFIT academic usage)
- **End date:** 30 April 2019
- **Installed on:** *Apolo II*, *Cronos*
- **Available MATLAB toolboxes:** List

Installation

This entry covers the entire process performed for the installation and configuration of MATLAB and the Distributed Computing Server on a cluster with the conditions described above.

Contents

[1] Wikipedia contributors. (2018, July 9). MATLAB. In Wikipedia, The Free Encyclopedia. Retrieved 15:32, July 13, 2018, from https://en.wikipedia.org/w/index.php?title=MATLAB&oldid=849495321

Tested on (Requirements)

- **License Manager Server:** Virtual machine (CentOS 7 Minimal (x86_64))
- **OS base:** CentOS (x86_64) $\geq$ 6.6
- **MPI:** Intel MPI $\geq$ 17.0.1 (Mandatory to use with Infiniband networks)
- **Scheduler:** SLURM $\geq$ 16.05.6
- **Application:** MATLAB Client (Optional)
- **Extra Libraries:**
  - libXtst (*Troubleshooting*)

License Manager

The *License Manager* provides a network license support to allow the usage of the different MATLAB features on the clusters (Apolo II and Cronos).

In this case we have two types of licenses, the first one is for the MATLAB Distributed Computing Engine (MDCE) and the second one for MATLAB client with all the toolboxes available.

Next steps will describe the installation and configuration process for the MLM (MATLAB License Manager based on FlexLM[1]):

1. Get the online installer using your MATLAB account.
2. Send the installation package to the License Manager server (VM).

```
scp matlab_R2018a_glnxa64.zip root@<FQDN>:$installer_path
```

3. Follow the next steps to run the MATLAB installer.
   1. Unzip and access the installer files.

```
ssh -X root@<FQDN>
cd $installer_path$
mkdir matlab-R2018a
mv matlab-R2018a matlab_R2018a_glnxa64.zip
cd matlab-R2018a
unzip matlab_R2018a_glnxa64.zip
```

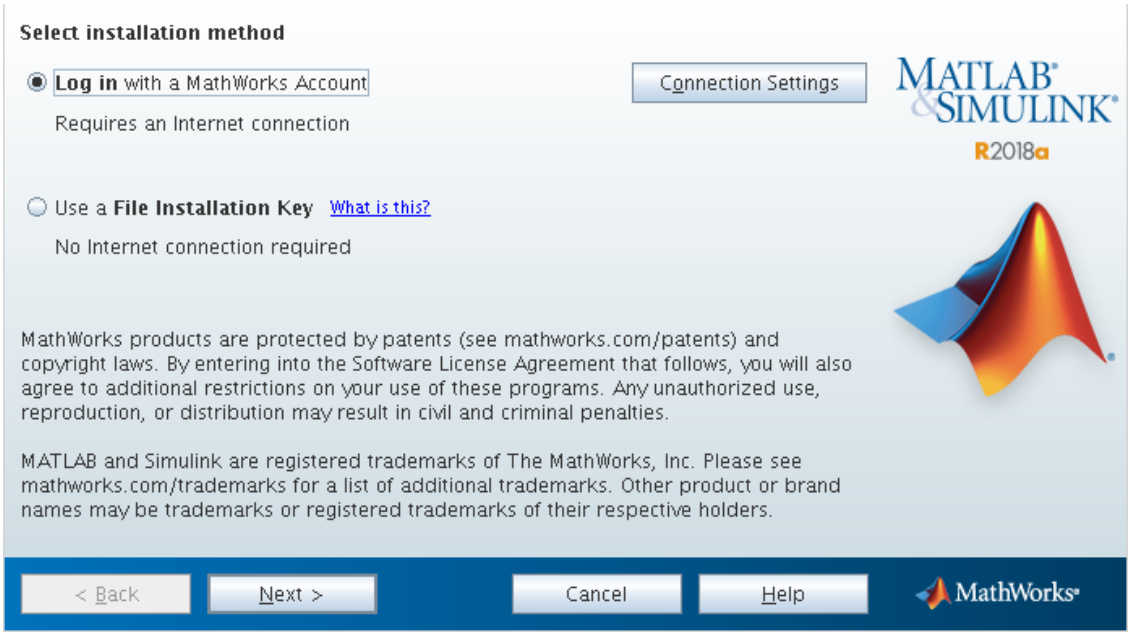
   2. Execute the installer.

```
./install
```

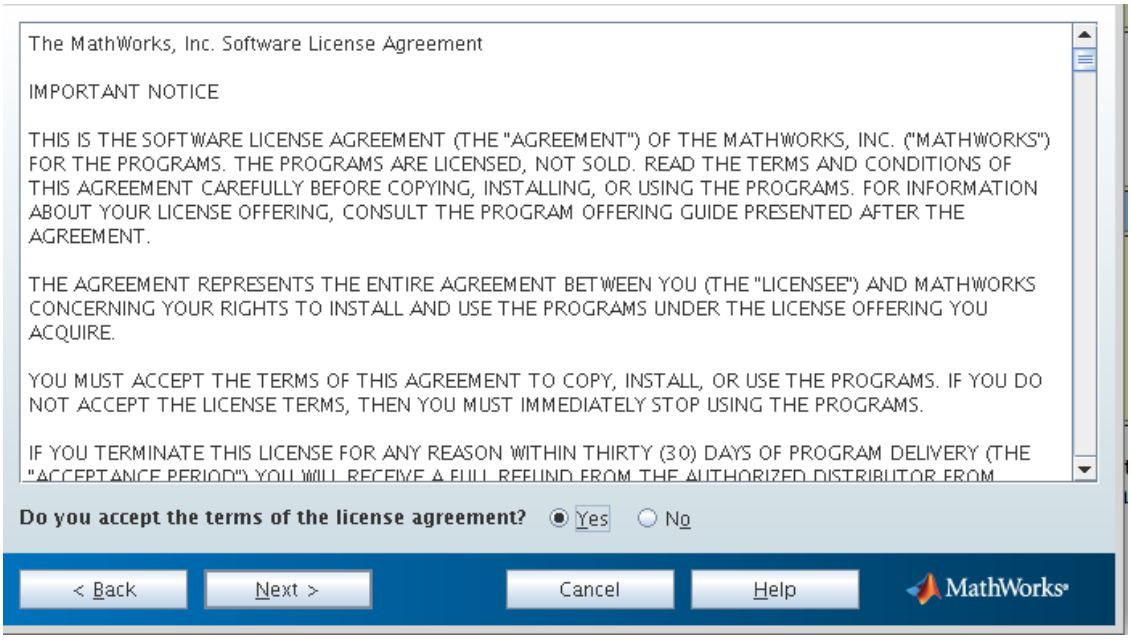
Note: *Troubleshooting*

[1] Wikipedia contributors. (2018, April 13). FlexNet Publisher. In Wikipedia, The Free Encyclopedia. Retrieved 20:44, July 18, 2018, from https://en.wikipedia.org/w/index.php?title=FlexNet_Publisher&oldid=836261861

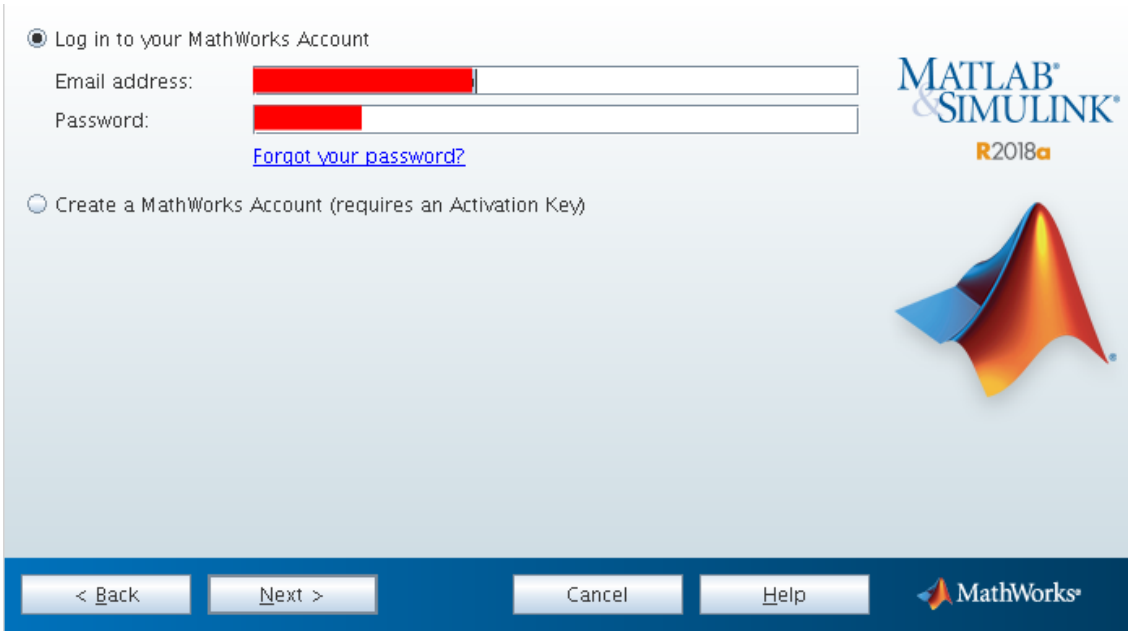
3. Select the installation method (by MATLAB account).



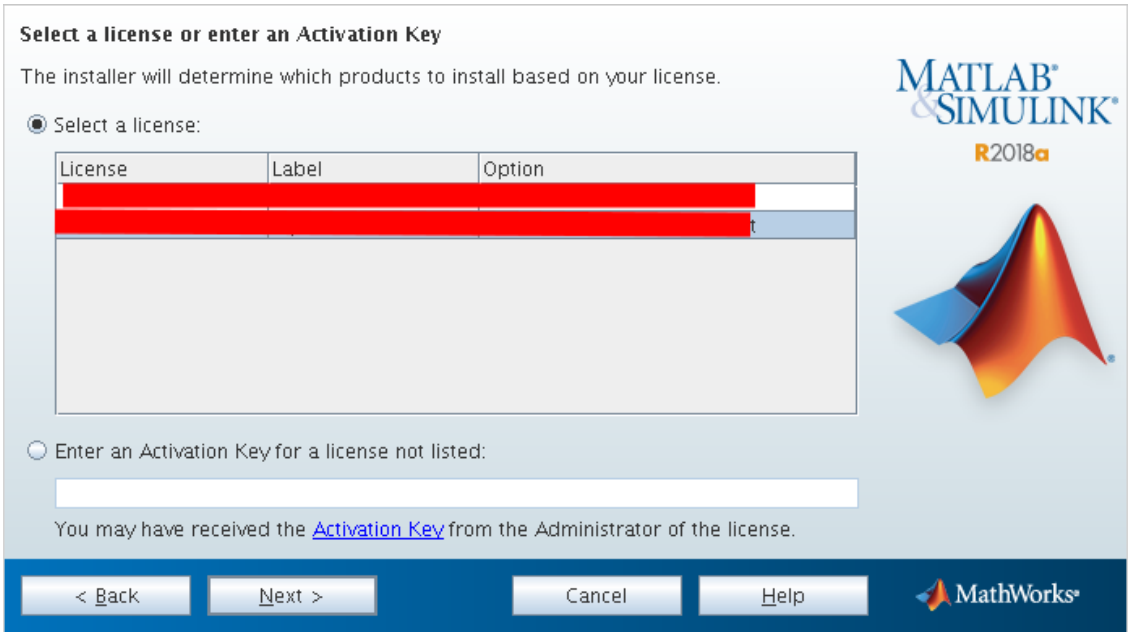
4. Accept license agreement (yes).



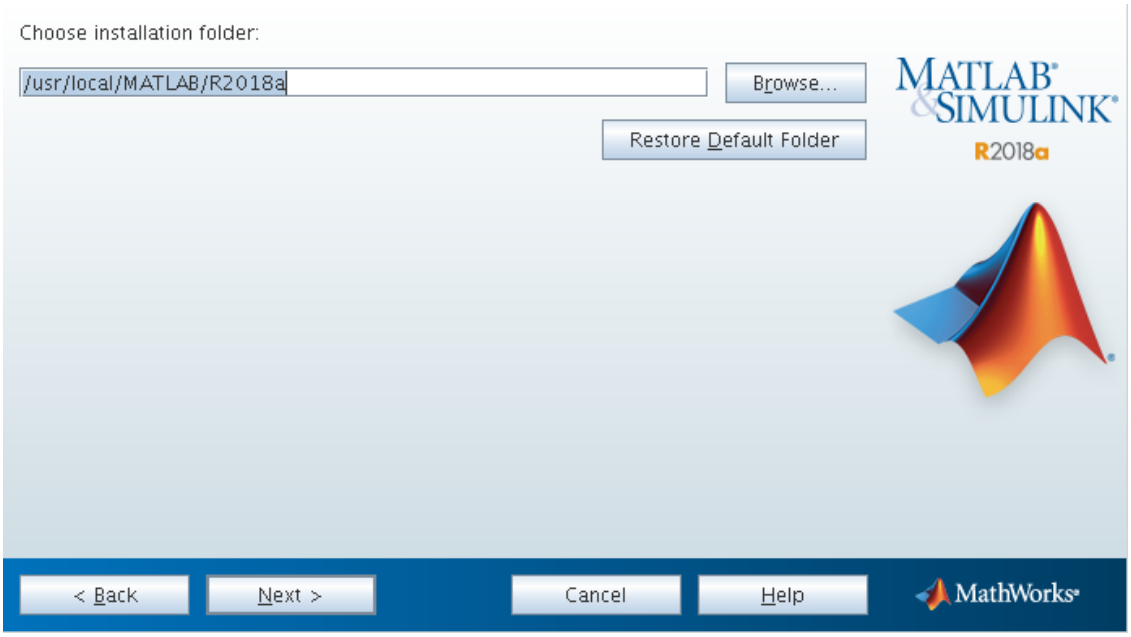
5. Login (username and password).



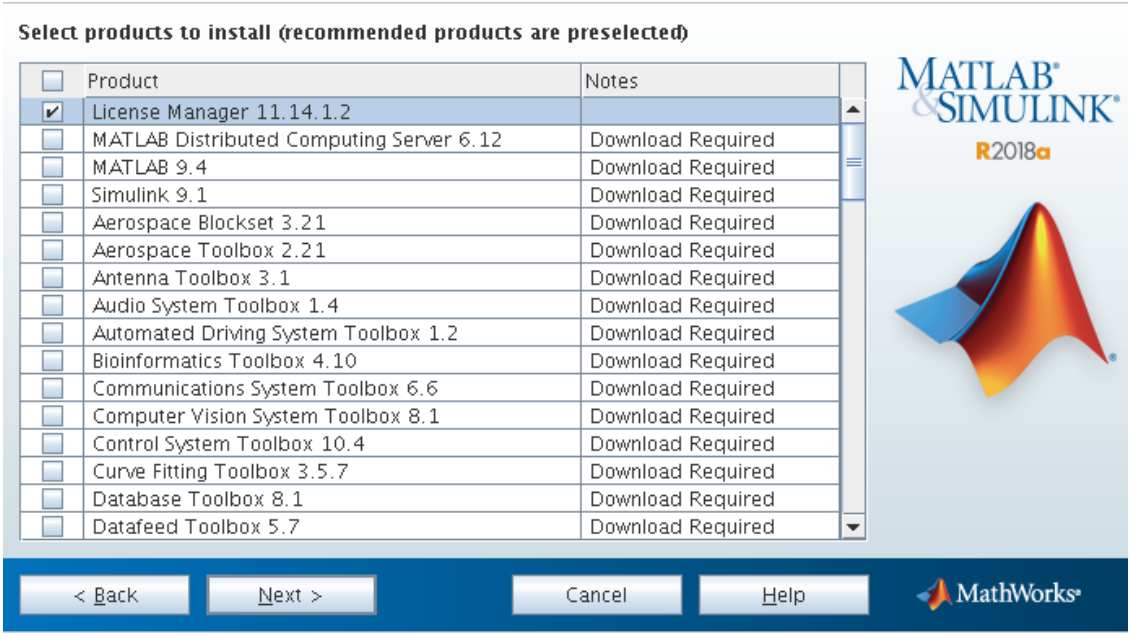
6. Select license (MDCE license).



7. Folder selection (/usr/local/MATLAB/R2018a).



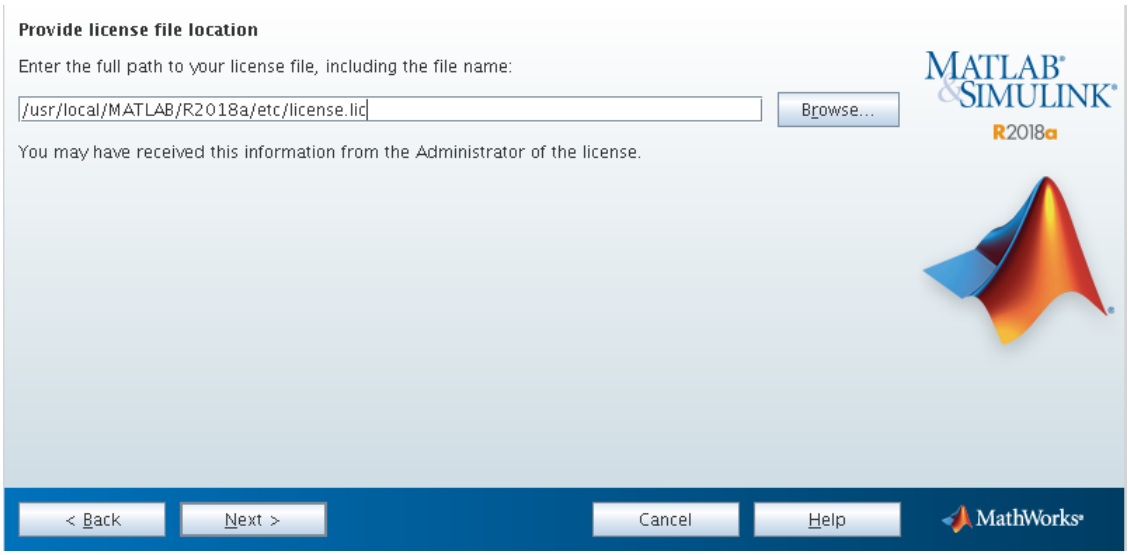
8. Products selection (License Manager 11.14.1.2).



9. License file.

Note: Login to the MATLAB admin account and download the license file (*license.dat*) created for this feature (MDCE - MATLAB Distributed Computign Engine) and upload it to the *License Manager* server in the `/usr/local/MATLAB/R2018a/etc` directory.

- `scp license.lic root@<FQDN>: /usr/local/MATLAB/R2018a/etc`



10. Finish the installation process.

4. Configure MLM (FlexLM).

    1. Access the *License Manager* machine via **SSH**.
    2. Create a system user without privileges to run MLM.

```
# Create a non-root user to launch matlab (security reasons)
## -u uid
## -d homedir
## -r system user
## -s shell (no login user)
useradd -u 110 -c "MDCE" -d /var/tmp -r -s /sbin/nologin matlab
```

    3. Create the daemon service to execute automatically MLM.

Listing 1: `lm-matlab.service`

```
[Unit]
Description=MATLAB FlexLM license manager

[Service]
User=matlab
RemainAfterExit=True
ExecStart=/usr/local/MATLAB/R2018a/etc/lmstart
ExecStop=/usr/local/MATLAB/R2018a/etc/lmdown

[Install]
WantedBy=multi-user.target
```

    4. Configure MLM ports and firewall on the license manager machine.

        - Review the server port (27000) and specify MLM daemon port (53200) at the top of the license file (`/usr/local/MATLAB/R2018a/etc/license.dat`)

```
SERVER <HOSTNAME> <HOSTID> 27000
DAEMON MLM "/usr/local/MATLAB/R2018a/etc/MLM" port=53200
...
```

        - Open those ports in License manager machine firewall (CentOS 7).

```
firewall-cmd --permanent --add-port=53200/tcp
firewall-cmd --permanent --add-port=27000/tcp
```

5. Configure both licenses (MDCE and MATLAB client with all the toolboxes).

Note: After the installation process, the MLM generates a new file license called *license.dat* on the `/usr/local/MATLAB/R2018a/etc` directory with the information given in *license.lic* file during the installation process (MDCE license).

- Download the `license.lic` file related with MATLAB client and its toolboxes from the MATLAB administrator account, then open it with a text editor to copy all the **INCREMENTS** lines.
- Append all (MATLAB client and its toolboxes) **INCREMENTS** lines (licensed products) to end of the `license.dat` on the *License Manager* server.

```
SERVER <FQDN> <HOSTID> 27000
DAEMON MLM "/usr/local/MATLAB/R2018a/etc/MLM" port=53200
# BEGIN--------------BEGIN--------------BEGIN
# MathWorks license passcode file.
# LicenseNo: ########   HostID: ############
#
# R2018a
#
INCREMENT MATLAB_Distrib_Comp_Engine MLM 39 <END_DATE> <NUM_WORKES> \
...
INCREMENT MATLAB MLM 39 <END_DATE> ##################### \
...
INCREMENT SIMULINK MLM 39 <END_DATE> ##################### \
...
... continue ...
...
```

6. Enable and start the daemon.

```
systemctl enable lm-matlab
systemctl start  lm-matlab
```

7. Check the log file to see if everything works properly. `/var/tmp/lm_TMW.log`

```
8:49:38 (lmgrd) -----------------------------------------------
8:49:38 (lmgrd)   Please Note:
8:49:38 (lmgrd)
8:49:38 (lmgrd)   This log is intended for debug purposes only.
8:49:38 (lmgrd)   In order to capture accurate license
8:49:38 (lmgrd)   usage data into an organized repository,
8:49:38 (lmgrd)   please enable report logging. Use Flexera Software LLC's
8:49:38 (lmgrd)   software license administration  solution,
8:49:38 (lmgrd)   FlexNet Manager, to  readily gain visibility
8:49:38 (lmgrd)   into license usage data and to create
8:49:38 (lmgrd)   insightful reports on critical information like
8:49:38 (lmgrd)   license availability and usage. FlexNet Manager
8:49:38 (lmgrd)   can be fully automated to run these reports on
8:49:38 (lmgrd)   schedule and can be used to track license
8:49:38 (lmgrd)   servers and usage across a heterogeneous
8:49:38 (lmgrd)   network of servers including Windows NT, Linux
8:49:38 (lmgrd)   and UNIX.
```

(continues on next page)

(continued from previous page)

```
8:49:38 (lmgrd)
8:49:38 (lmgrd) -----------------------------------------------
8:49:38 (lmgrd)
8:49:38 (lmgrd)
8:49:38 (lmgrd) Server's System Date and Time: Wed Jul 18 2018 08:49:38 -05
8:49:38 (lmgrd) SLOG: Summary LOG statistics is enabled.
8:49:38 (lmgrd) FlexNet Licensing (v11.14.1.2 build 208719 x64_lsb) started␣
↪on <FQDN> (linux) (7/18/2018)
8:49:38 (lmgrd) Copyright (c) 1988-2017 Flexera Software LLC. All Rights␣
↪Reserved.
8:49:38 (lmgrd) World Wide Web:  http://www.flexerasoftware.com
8:49:38 (lmgrd) License file(s): /var/tmp/lm_TMW.dat
8:49:38 (lmgrd) lmgrd tcp-port 27000
...
8:49:38 (lmgrd) (@lmgrd-SLOG@) ===============================================
8:49:38 (lmgrd) (@lmgrd-SLOG@) === LMGRD ===
8:49:38 (lmgrd) (@lmgrd-SLOG@) Start-Date: Wed Jul 18 2018 08:49:38 -05
8:49:38 (lmgrd) (@lmgrd-SLOG@) PID: 19339
8:49:38 (lmgrd) (@lmgrd-SLOG@) LMGRD Version: v11.14.1.2 build 208719 x64_lsb␣
↪( build 208719 (ipv6))
8:49:38 (lmgrd) (@lmgrd-SLOG@)
8:49:38 (lmgrd) (@lmgrd-SLOG@) === Network Info ===
8:49:38 (lmgrd) (@lmgrd-SLOG@) Listening port: 27000
...
8:49:38 (lmgrd) (@lmgrd-SLOG@)
8:49:38 (lmgrd) (@lmgrd-SLOG@) === Startup Info ===
8:49:38 (lmgrd) (@lmgrd-SLOG@) Server Configuration: Single Server
8:49:38 (lmgrd) (@lmgrd-SLOG@) Command-line options used at LS startup: -z -c␣
↪/var/tmp/lm_TMW.dat
8:49:38 (lmgrd) (@lmgrd-SLOG@) License file(s) used:  /var/tmp/lm_TMW.dat
8:49:38 (lmgrd) (@lmgrd-SLOG@) ===============================================
8:49:38 (lmgrd) Starting vendor daemons ...
8:49:38 (lmgrd) Using vendor daemon port 53200 specified in license file
...
8:49:38 (lmgrd) Started MLM (internet tcp_port 53200 pid 19341)
...
8:49:38 (MLM) FlexNet Licensing version v11.14.1.2 build 208719 x64_lsb
8:49:38 (MLM) SLOG: Summary LOG statistics is enabled.
8:49:38 (MLM) SLOG: FNPLS-INTERNAL-CKPT1
8:49:38 (MLM) SLOG: VM Status: 0
...
8:49:38 (lmgrd) MLM using TCP-port 53200
8:49:38 (MLM) License verification completed successfully.
...
8:49:38 (MLM) SLOG: Statistics Log Frequency is 240 minute(s).
8:49:38 (MLM) SLOG: TS update poll interval is 600 seconds.
8:49:38 (MLM) SLOG: Activation borrow reclaim percentage is 0.
8:49:38 (MLM) (@MLM-SLOG@) ===========================================
8:49:38 (MLM) (@MLM-SLOG@) === Vendor Daemon ===
8:49:38 (MLM) (@MLM-SLOG@) Vendor daemon: MLM
8:49:38 (MLM) (@MLM-SLOG@) Start-Date: Wed Jul 18 2018 08:49:38 -05
8:49:38 (MLM) (@MLM-SLOG@) PID: 19341
8:49:38 (MLM) (@MLM-SLOG@) VD Version: v11.14.1.2 build 208719 x64_lsb (␣
↪build 208719 (ipv6))
8:49:38 (MLM) (@MLM-SLOG@)
8:49:38 (MLM) (@MLM-SLOG@) === Startup/Restart Info ===
8:49:38 (MLM) (@MLM-SLOG@) Options file used: None
```

(continues on next page)

(continued from previous page)

```
8:49:38 (MLM) (@MLM-SLOG@) Is vendor daemon a CVD: No
8:49:38 (MLM) (@MLM-SLOG@) Is TS accessed: No
8:49:38 (MLM) (@MLM-SLOG@) TS accessed for feature load: -NA-
8:49:38 (MLM) (@MLM-SLOG@) Number of VD restarts since LS startup: 0
8:49:38 (MLM) (@MLM-SLOG@)
8:49:38 (MLM) (@MLM-SLOG@) === Network Info ===
8:49:38 (MLM) (@MLM-SLOG@) Listening port: 53200
8:49:38 (MLM) (@MLM-SLOG@) Daemon select timeout (in seconds): 1
8:49:38 (MLM) (@MLM-SLOG@)
8:49:38 (MLM) (@MLM-SLOG@) === Host Info ===
8:49:38 (MLM) (@MLM-SLOG@) Host used in license file: <FQDN>
...
```

8. After that, the license manager service should run without problems, if there is any trouble with the service you can debug this process checking the log file (`/var/tmp/lm_TMW.log`) to understand what is happening.

```
tailf /var/tmp/lm_TMW.log
```

MATLAB Distributed Computing Server (MDCS)

This entry described the installation process of MDCS on the cluster and its integration with the *License Manager*.

1. Get the online installer using your MATLAB account.
2. Send the installation file to the master node on your cluster.

```
scp matlab_R2018a_glnxa64.zip root@<FQDN>:$installer_path$
```

3. Follow next steps to run the MATLAB installer.
    1. Unzip and access the installer files.

```
ssh -X root@<FQDN>
cd $installer_path$
mkdir matlab-R2018a
mv matlab-R2018a matlab_R2018a_glnxa64.zip
cd matlab-R2018a
unzip matlab_R2018a_glnxa64.zip
```

    2. Create the installation directory.

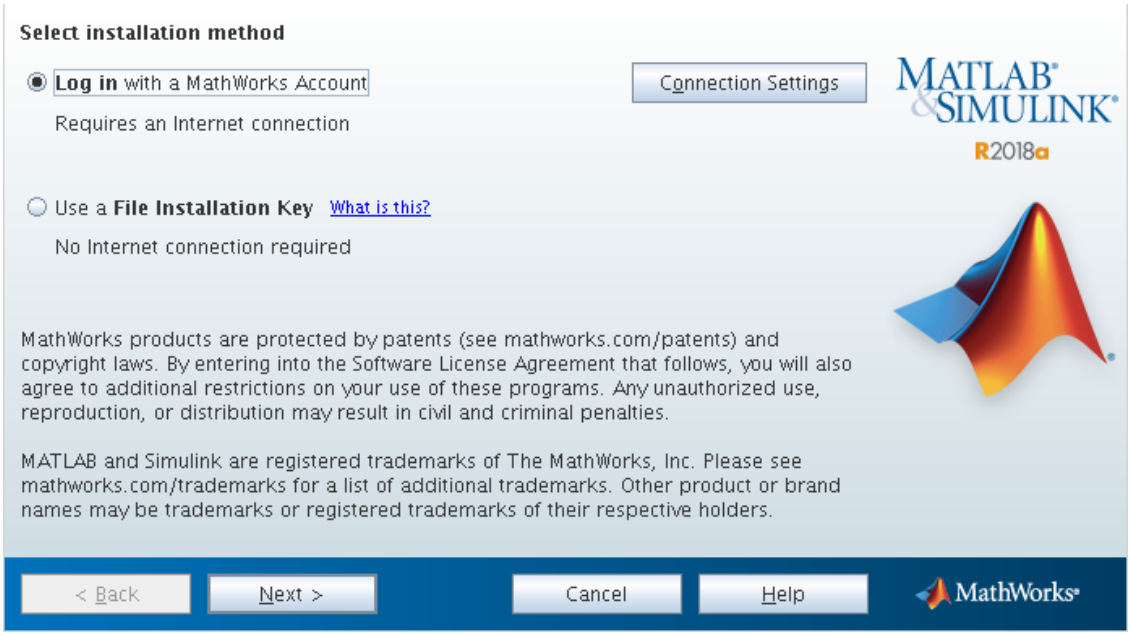
```
mkdir -p /share/apps/matlab/r2018a
```

    3. Execute the installer.

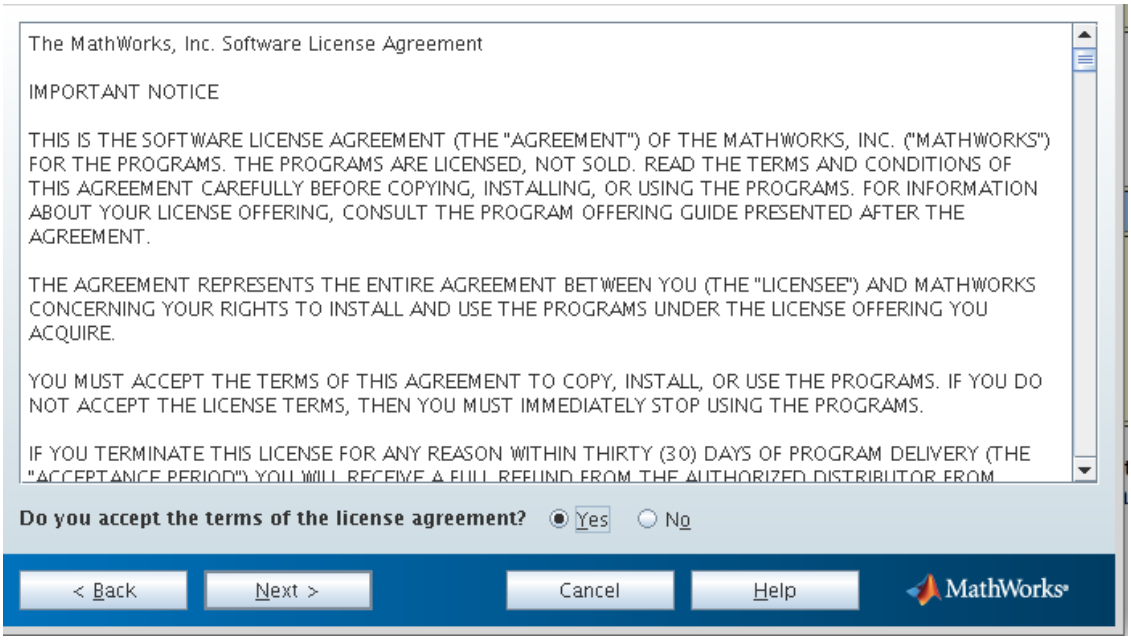
```
./install
```

> **Note:** *Troubleshooting*

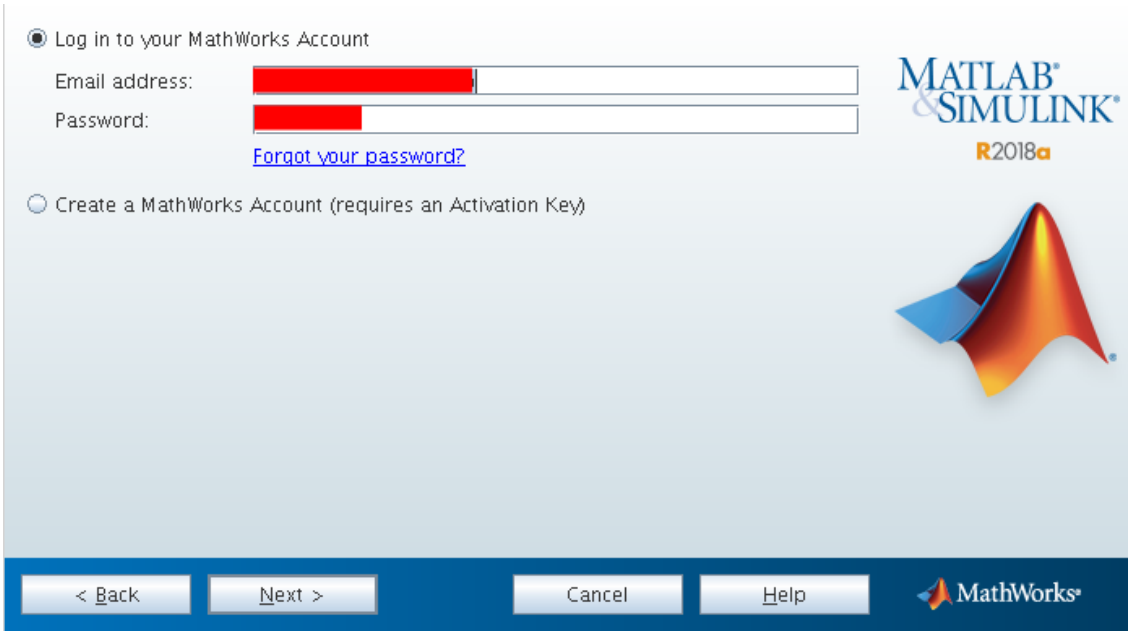
    4. Select the installation method (by MATLAB account).



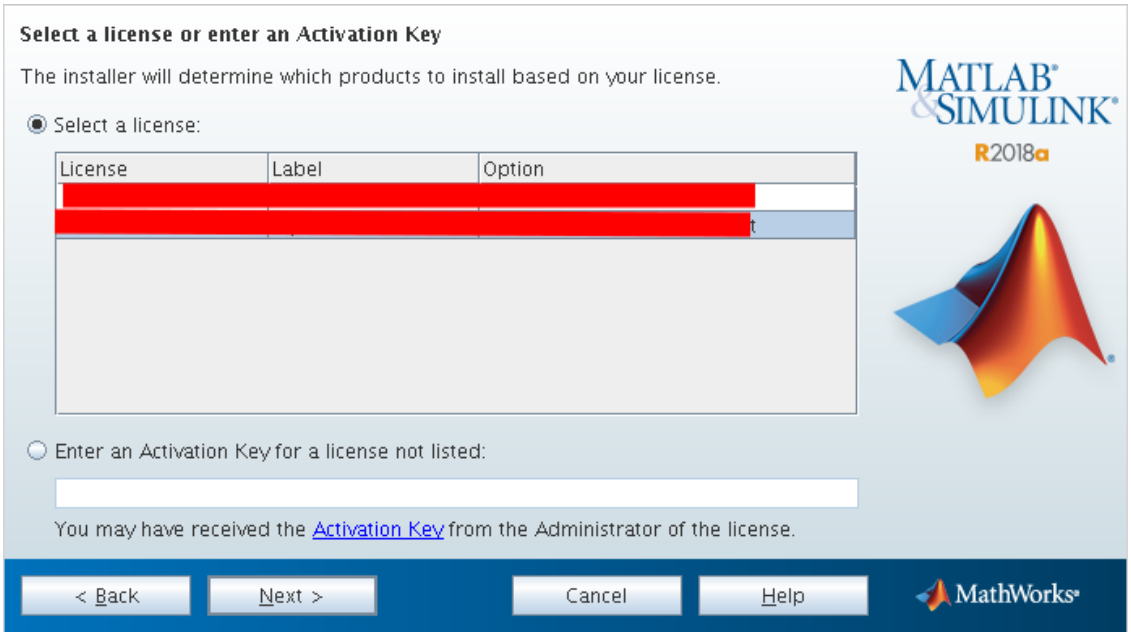
5. Accept license agreement (yes).



6. Login (username and password).



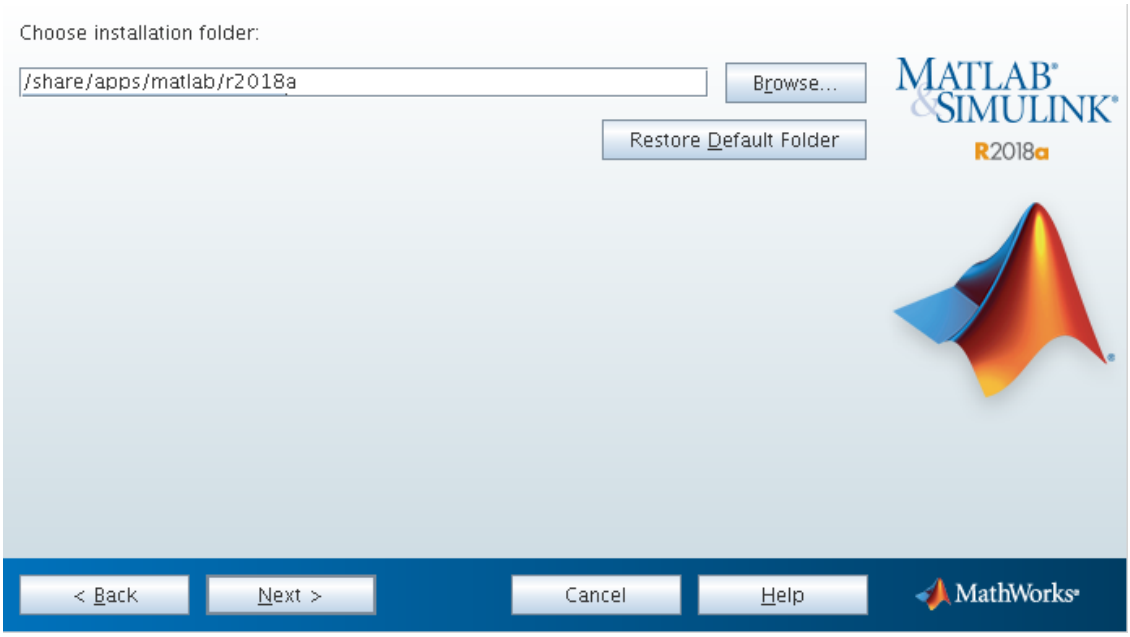
7. Select license (MDCE license).



8. Folder selection (`/share/apps/matlab/r2018a`).

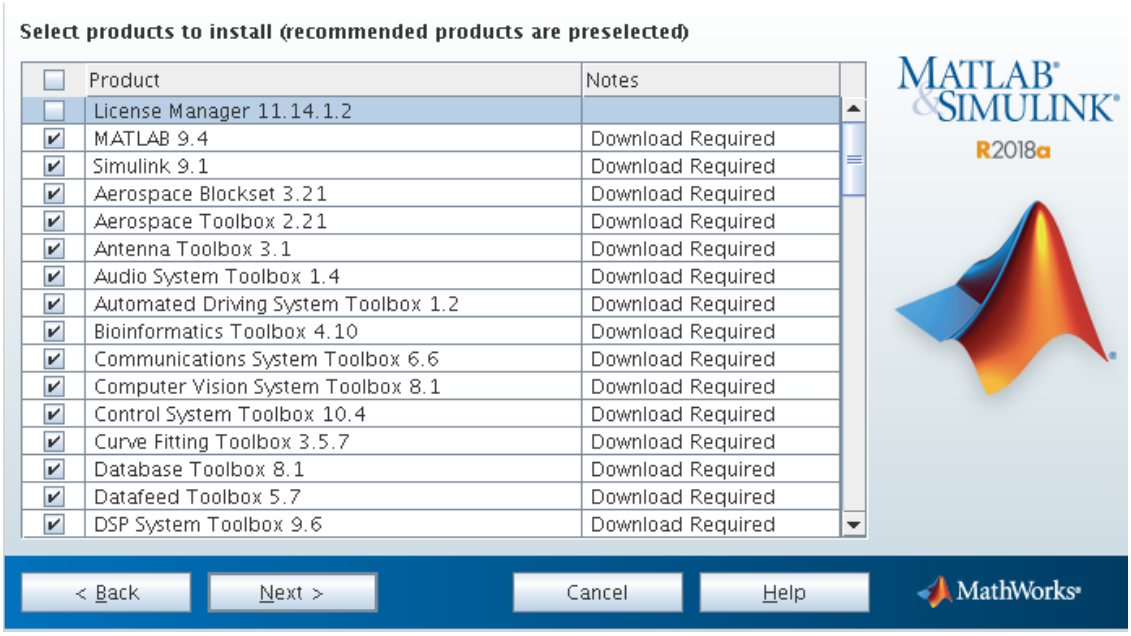
Note: Use a shared file system to do an unique installtion across all the nodes in the cluster.

- /share/apps/matlab



9. Products selection (All products except License Manager 11.14.1.2).

Note: MATLAB recommends install every *Toolbox* available because in this way they can be used by MDCE workers.



10. License file (`/share/apps/matlab/r2018a/etc`).

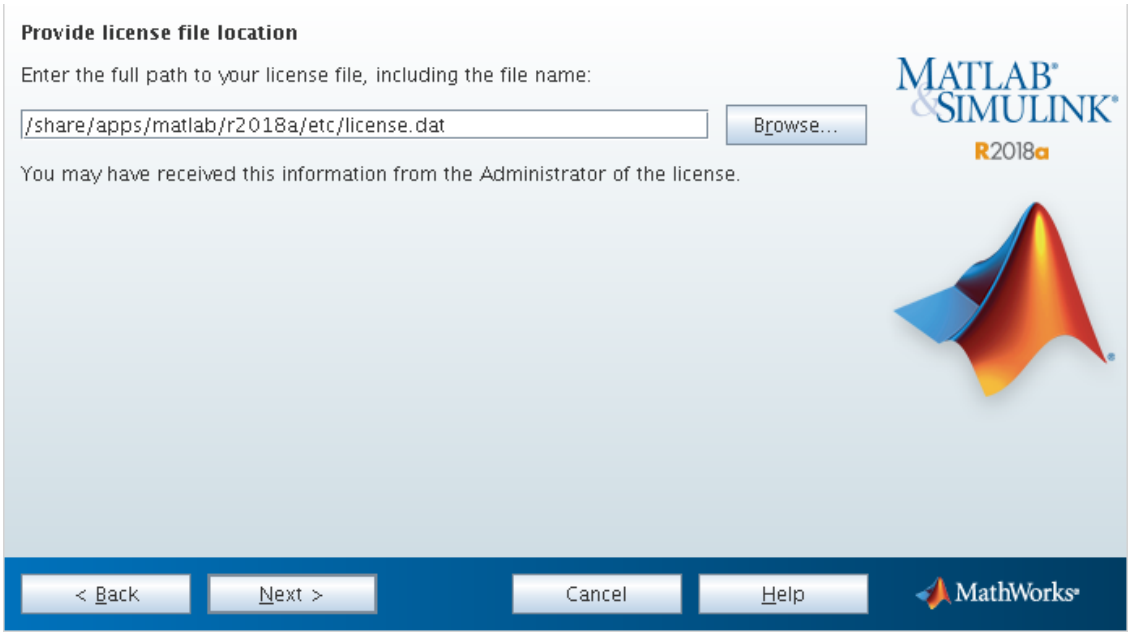
Note: Download and upload the modified `license.dat` file on the *License Manager* server to the `/share/apps/matlab/r2018a/etc` directory on the cluster.

```
mkdir -p /share/apps/matlab/r2018a/etc
```

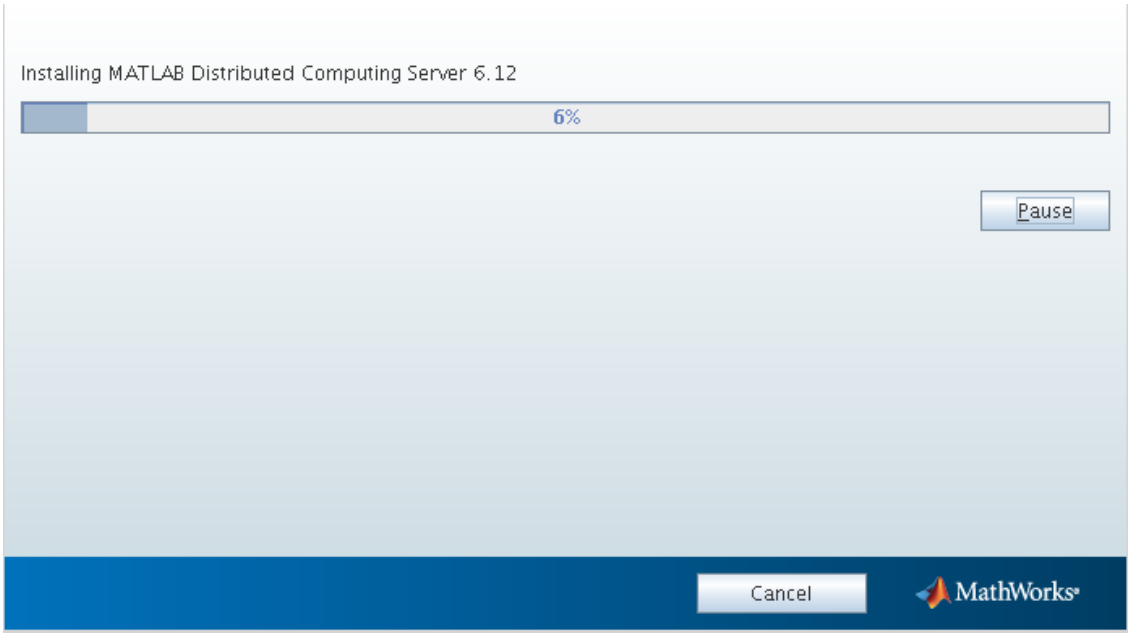
(continues on next page)

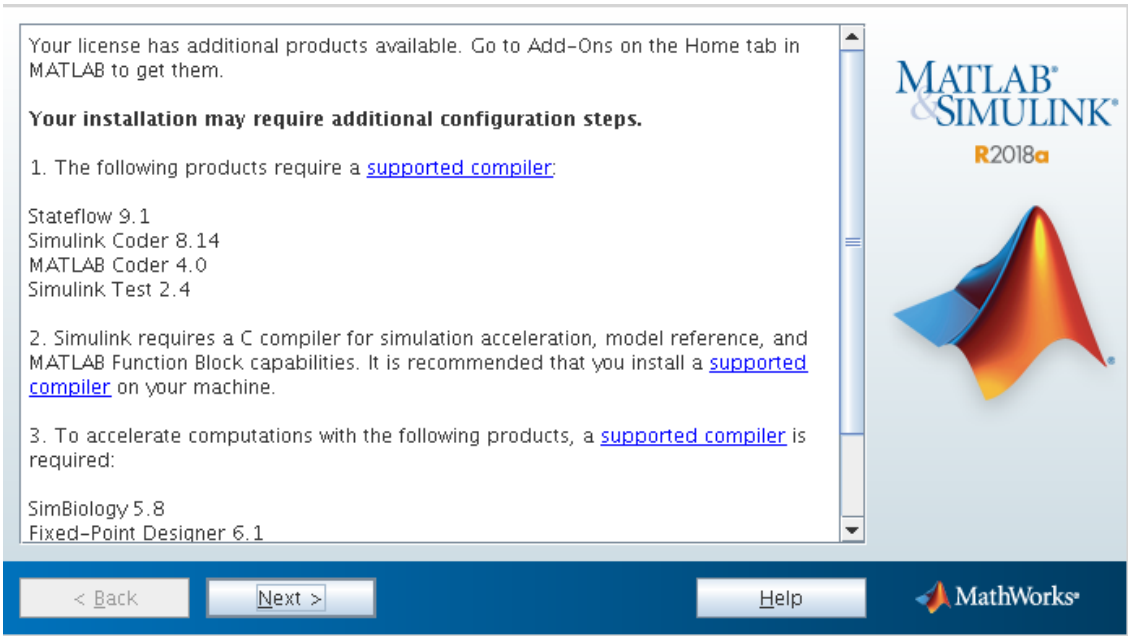
(continued from previous page)

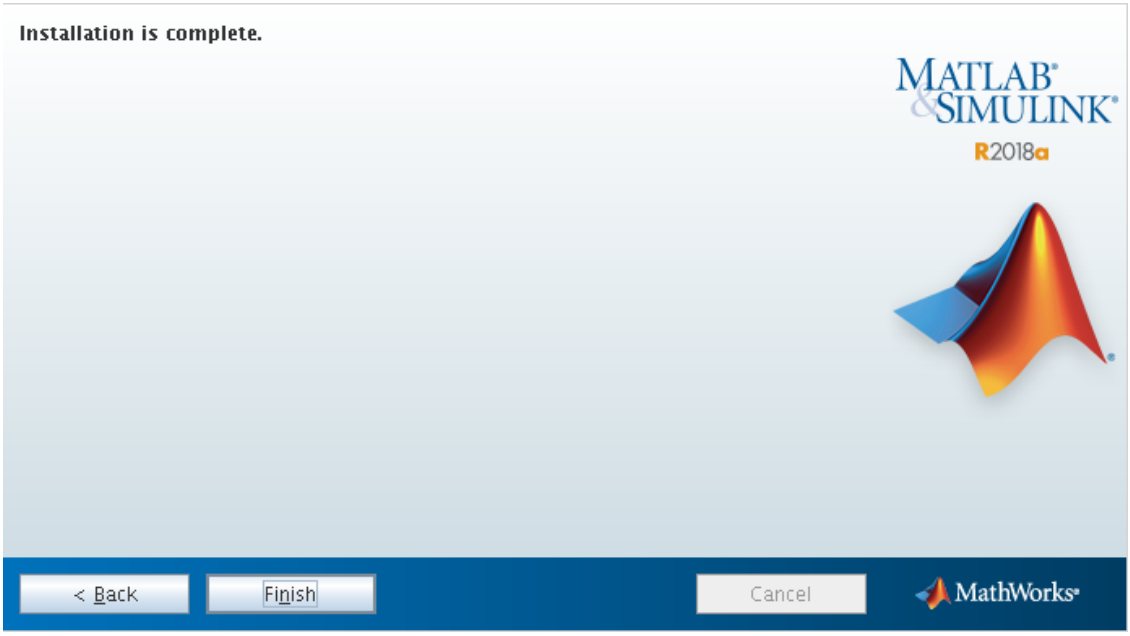
```
cd /share/apps/matlab/r2018a/etc
sftp user@<LICENSE_MANAGER_SERVER>
cd /usr/local/MATLAB/R2018a/etc
mget license.dat
```



11. Finish the installation process.







Intregration with SLURM

To integrate the MATLAB client on the cluster to use SLURM as resource manager you have to follow next steps:

1. Add the MATLAB integration scripts to its MATLAB PATH by placing the integration scripts into `/share/apps/matlab/r2018a/toolbox/local` directory (`Apolo II` or `Cronos`).

Linux

```
scp apolo.local.zip or cronos.local.zip <user>@cluster:$path/to/file
mv $path/to/file/matlab-apolo.zip$ /share/apps/matlab/r2018a/toolbox/local
```

(continues on next page)

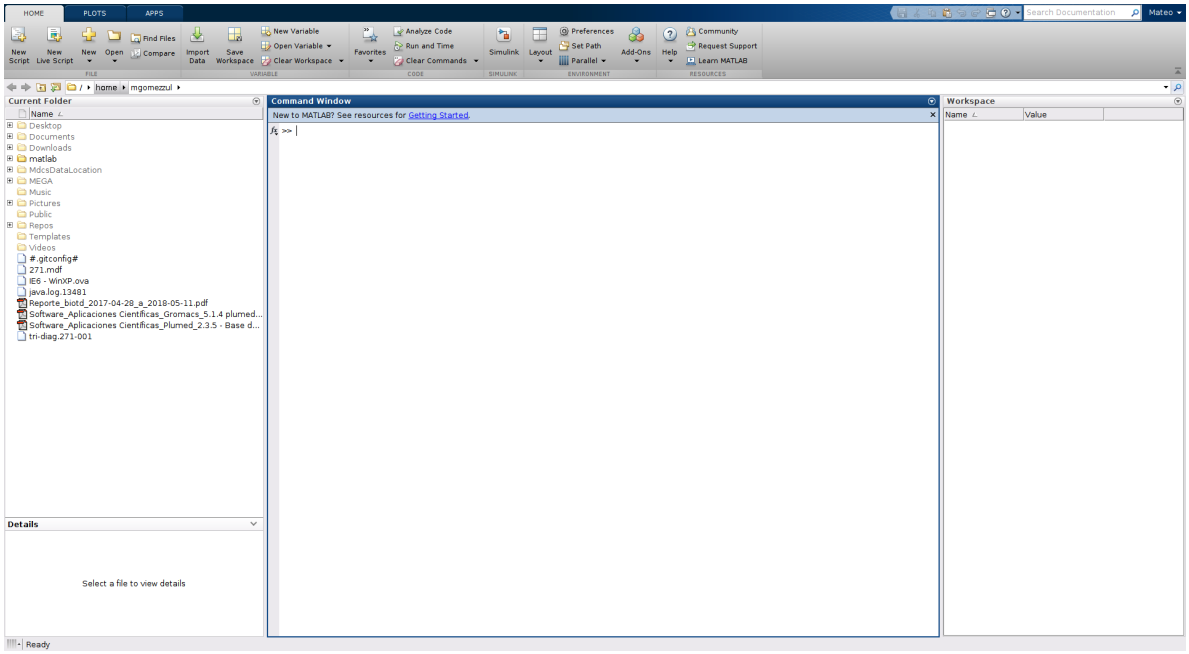
(continued from previous page)

```
cd /share/apps/matlab/r2018a/toolbox/local
unzip apolo.local.zip or cronos.local.zip
rm apolo.local.zip or cronos.local.zip
```

2. Open the MATLAB client on the cluster to configure it.

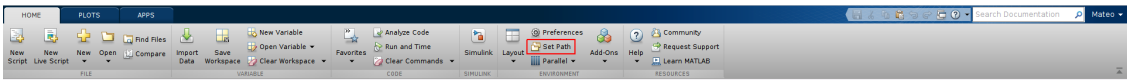
   (If MATLAB client is installed in a system directory, we strongly suggest to open it with admin privileges, it is only necessary the first time to configure it).

```
ssh -X username@<master>
module load matlab/r2018a
matlab
```

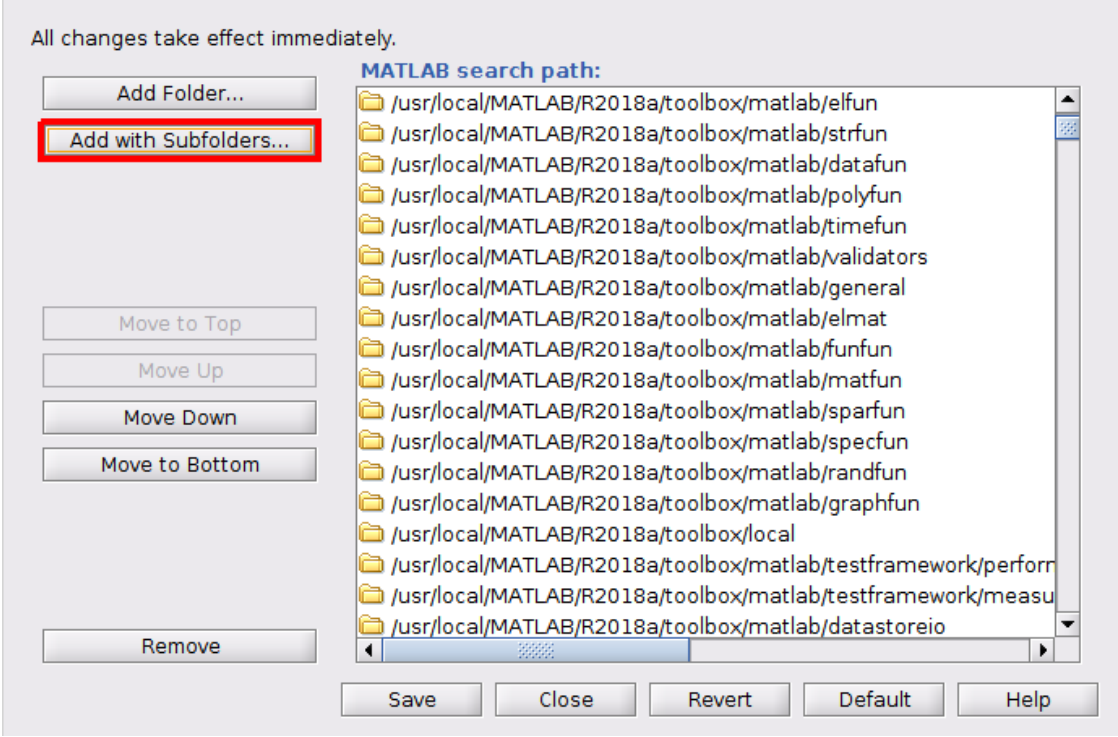


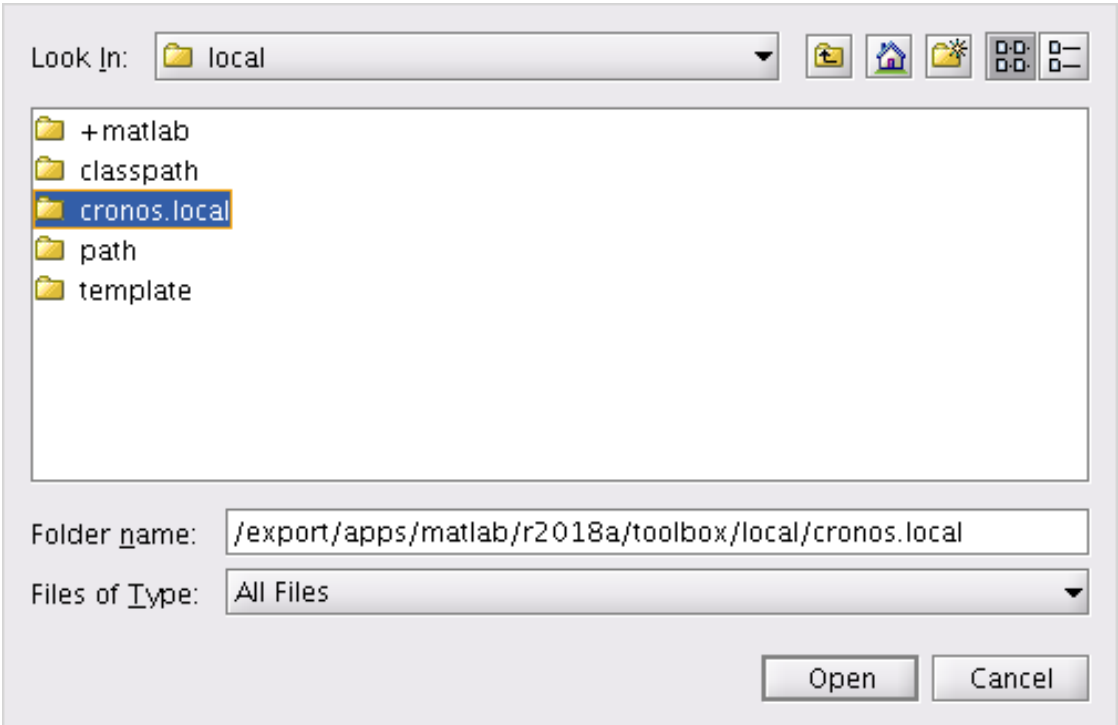
3. Add the integrations scripts to the MATLAB PATH

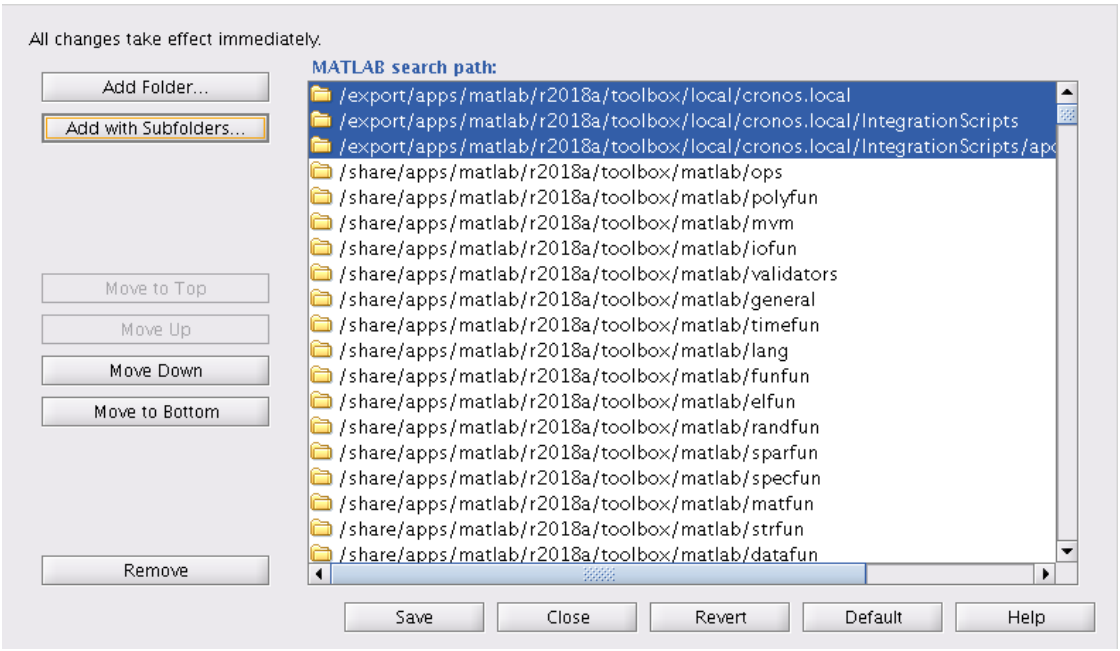
   - Press the **"Set Path"** button



   - Press the **"Add with Subfolders"** button and choose the directory where you unzip the integrations scripts and finally press the **"Save"** button:
     - `/share/apps/matlab/r2018a/toolbox/local/cronos.local \or\ apolo.local`

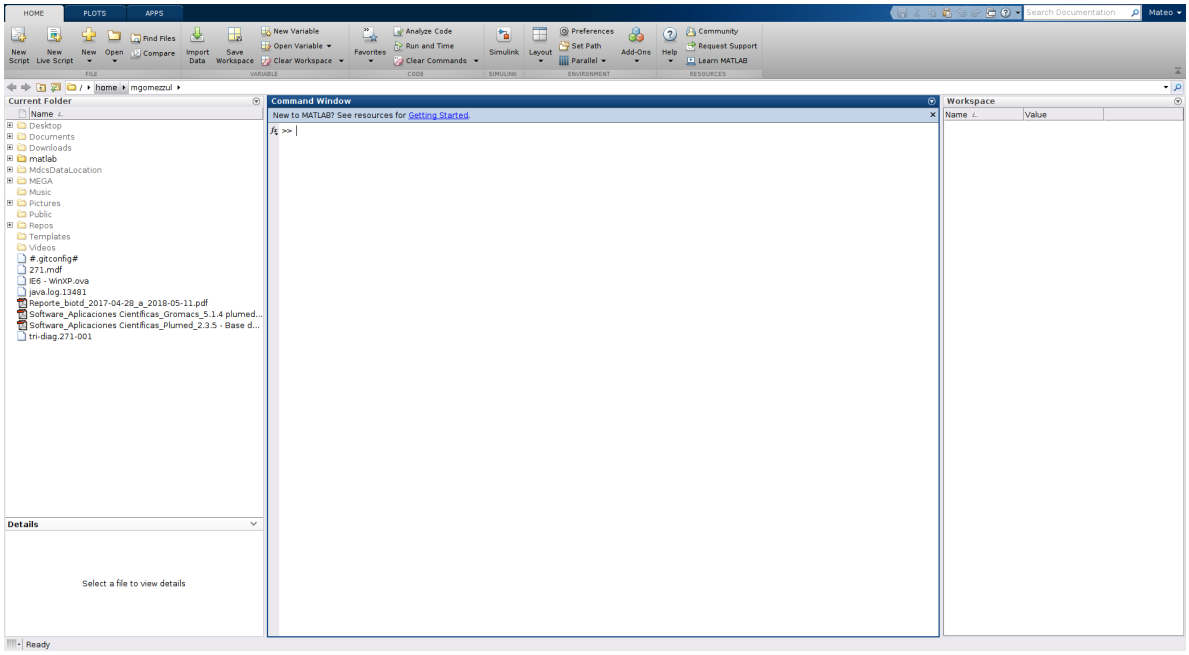

All changes take effect immediately.
MATLAB search path:
Add Folder...
Add with Subfolders...
Move to Top
Move Up
Move Down
Move to Bottom
Remove
/usr/local/MATLAB/R2018a/toolbox/matlab/elfun
/usr/local/MATLAB/R2018a/toolbox/matlab/strfun
/usr/local/MATLAB/R2018a/toolbox/matlab/datafun
/usr/local/MATLAB/R2018a/toolbox/matlab/polyfun
/usr/local/MATLAB/R2018a/toolbox/matlab/timefun
/usr/local/MATLAB/R2018a/toolbox/matlab/validators
/usr/local/MATLAB/R2018a/toolbox/matlab/general
/usr/local/MATLAB/R2018a/toolbox/matlab/elmat
/usr/local/MATLAB/R2018a/toolbox/matlab/funfun
/usr/local/MATLAB/R2018a/toolbox/matlab/matfun
/usr/local/MATLAB/R2018a/toolbox/matlab/sparfun
/usr/local/MATLAB/R2018a/toolbox/matlab/specfun
/usr/local/MATLAB/R2018a/toolbox/matlab/randfun
/usr/local/MATLAB/R2018a/toolbox/matlab/graphfun
/usr/local/MATLAB/R2018a/toolbox/local
/usr/local/MATLAB/R2018a/toolbox/matlab/testframework/perforn
/usr/local/MATLAB/R2018a/toolbox/matlab/testframework/measu
/usr/local/MATLAB/R2018a/toolbox/matlab/datastoreio
Save
Close
Revert
Default
Help


Look In:
local
+matlab
classpath
cronos.local
path
template
Folder name:
/export/apps/matlab/r2018a/toolbox/local/cronos.local
Files of Type:
All Files
Open
Cancel



Configuring Cluster Profiles

1. Open again your MATLAB Client (without admin privilages)



2. Load the cluster profile and configure it to submit jobs using SLURM via MDCS.

```
>> configCluster
>> % Must set TimeLimit before submitting jobs to Cronos

>> % e.g. to set the TimeLimit and Partition
>> c = parcluster('cronos');
>> c.AdditionalProperties.TimeLimit = '1:00:00';
>> c.AdditionalProperties.Partition = 'longjobs';
>> c.saveProfile
```

3. Custom options

- **TimeLimit** Set a limit on the total run time of the job allocation (more info).
  - e.g. `c.AdditionalProperties.TimeLimit = '3-10:00:00';`
- **AccountName** Change the default user account on Slurm.
  - e.g. `c.AdditionalProperties.AccountName = 'apolo';`
- **ClusterHost** Another way to change the cluster hostname to sumbit jobs.
  - e.g. `c.AdditionalProperties.ClusterHost = 'apolo.eafit.edu.co';`
- **EmailAddress** Get all job notifications by e-mail.
  - e.g. `c.AdditionalProperties.EmailAddress = 'apolo@eafit.edu.co';`
- **EmailType** Get only the desired notifications based on sbatch options.
  - e.g. `c.AdditionalProperties.EmailType = 'END,TIME_LIMIT_50';`
- **MemUsage** Total amount of memory per machine (more info).
  - e.g. `c.AdditionalProperties.MemUsage = '5G';`
- **NumGpus** Number of GPUs to use in a job (currently the maximum possible NumGpus value is two, also if you select this option you have to use the *'accel'* partition on *Apolo II*).
  - e.g. `c.AdditionalProperties.NumGpus = '2';`
- **Partition** Select the desire partition to submit jobs (by default *longjobs* partition will be used)
  - e.g. `c.AdditionalProperties.Partition = 'bigmem';`
- **Reservation** Submit a job into a reservation (more info).
  - e.g. `c.AdditionalProperties.Reservation = 'reservationName';`
- **AdditionalSubmitArgs** Any valid sbatch parameter (raw) (more info)
  - e.g. `c.AdditionalProperties.AdditionalSubmitArgs = '-no-requeue';`

Troubleshooting

1. When you ran the MATLAB installer with the command `./install`, it prints:

```
Preparing installation files ...
Installing ...
```

Then a small MATLAB window appears and after a while it closes and prints on prompt:

```
Finished
```

To solve this problem, you have to find the root cause modifying `$MATLABINSTALLERPATH/bin/glnxa64/install_unix` script to look the `stderror` and understand what is happening.

- At line *918* change this statement `eval "$java_cmd 2> /dev/null"` to `eval "$java_cmd"`, by this way you can see the related errors launching the MATLAB installer.
  - e.g. missing library *libXtst.so.6*

Module file

Listing 2: `Module file`

```
#%Module1.0#####################################################################
##
## module load matlab/r2018a
##
## /share/apps/modules/matlab/r2018a
## Written by Mateo Gómez Zuluaga
##

proc ModulesHelp {} {
     global version modroot
     puts stderr "Sets the environment for using Matlab R2018a\
                  \nin the shared directory /share/apps/matlab/r2018a."
}

module-whatis "(Name________) matlab"
module-whatis "(Version_____) r2018a"
module-whatis "(Compilers___) "
module-whatis "(System______) x86_64-redhat-linux"
module-whatis "(Libraries___) "

# for Tcl script use only
set         topdir        /share/apps/matlab/r2018a
set         version       r2018a
set         sys           x86_64-redhat-linux

conflict matlab

prepend-path              PATH        $topdir/bin
```

Usage

This subsection describes three integrations methods to submit jobs to the cluster and in this way use the computational resources through the available licenses in Apolo for MATLAB.

Contents

MDCS using a local MATLAB client

To submit jobs through a local MATLAB client in Apolo II or Cronos using SLURM follow next steps to got the integration:

Integration scripts

1. Add the MATLAB integration scripts to your MATLAB PATH by placing the integration scripts into `$HOME/Documents/matlab-integration` directory (`matlab-apolo.zip`).

   Linux

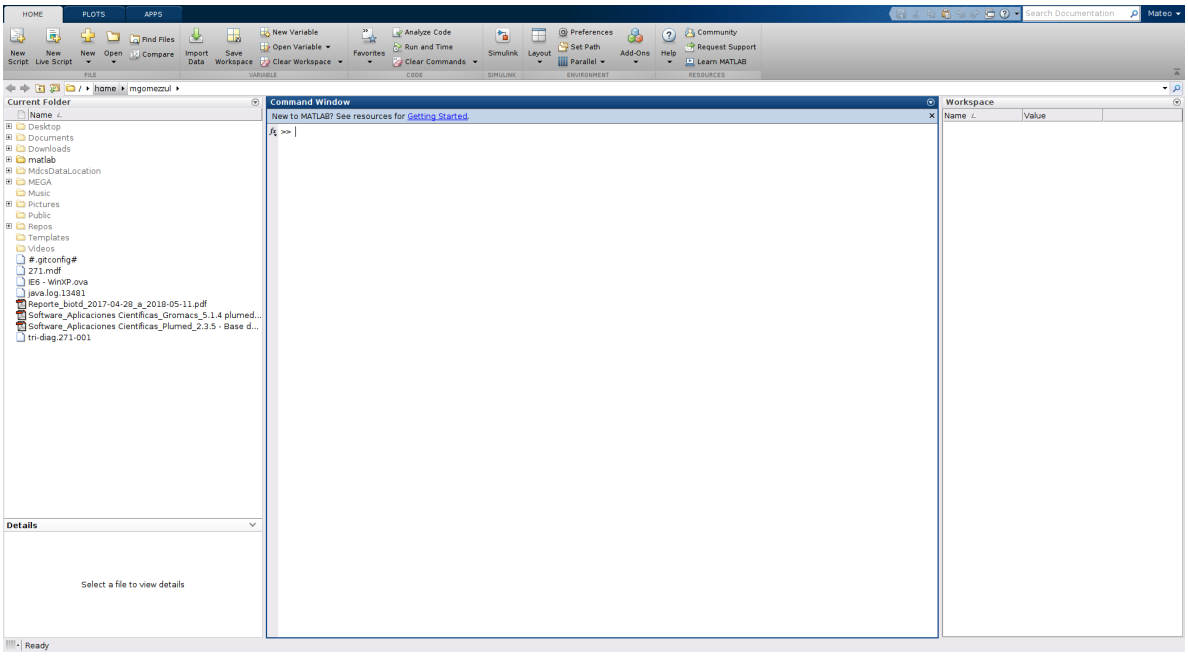
   ```
   mkdir $HOME/Documents/matlab-integration
   mv path-to-file/matlab-apolo.zip $HOME/matlab-integration/
   cd $HOME/Documents/matlab-integration
   unzip matlab-apolo.zip
   rm matlab-apolo.zip
   ```

   Windows

   To-Do

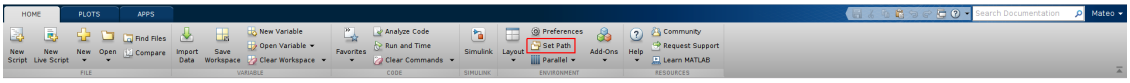
2. Open your MATLAB client to configure it.

   (If MATLAB client is installed in a system directory, we strongly suggest to open it with admin privileges, it is only necessary the first time to configure it).

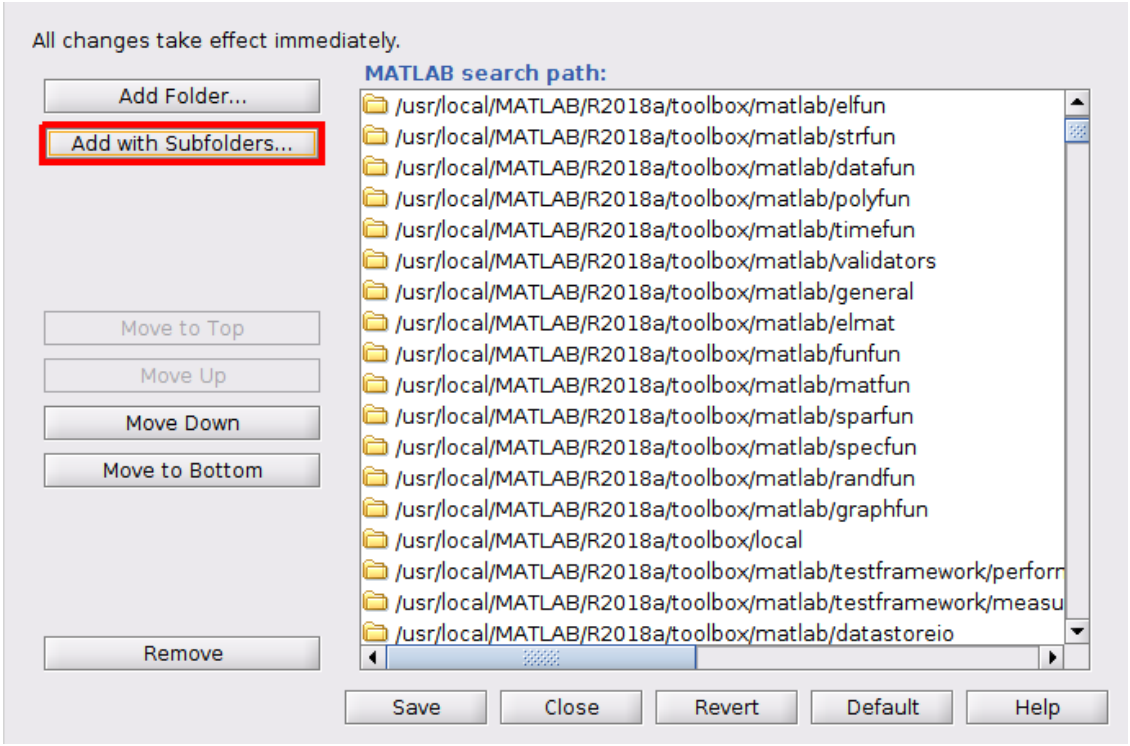


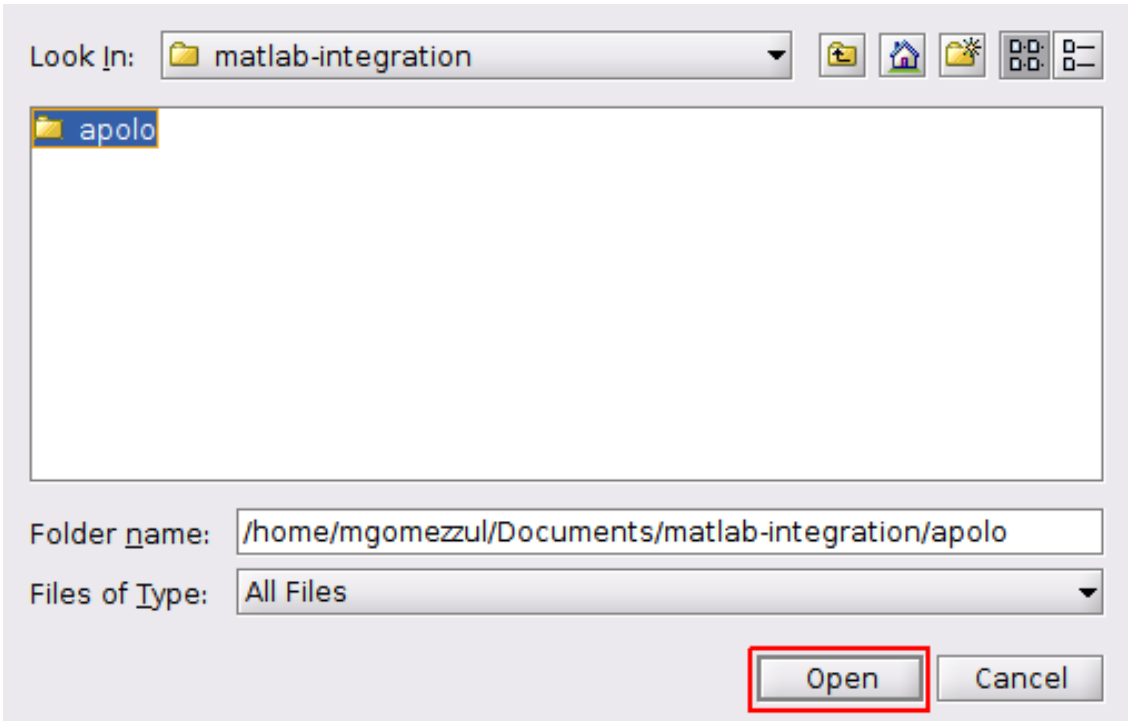
3. Add the integrations scripts to the MATLAB PATH

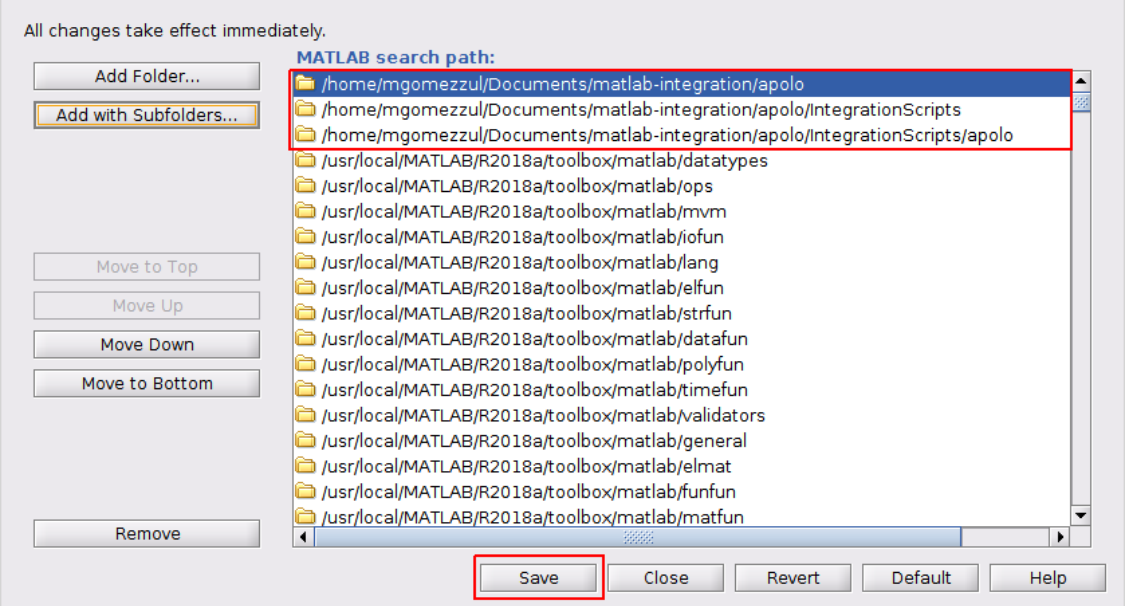
   - Press the **"Set Path"** button



   - Press the **"Add with Subfolders"** button and choose the directories where you unzip the integrations scripts (Apolo II and Cronos) and finally press the **"Save"** button:
     - `/home/$USER/matlab-integration/apolo`

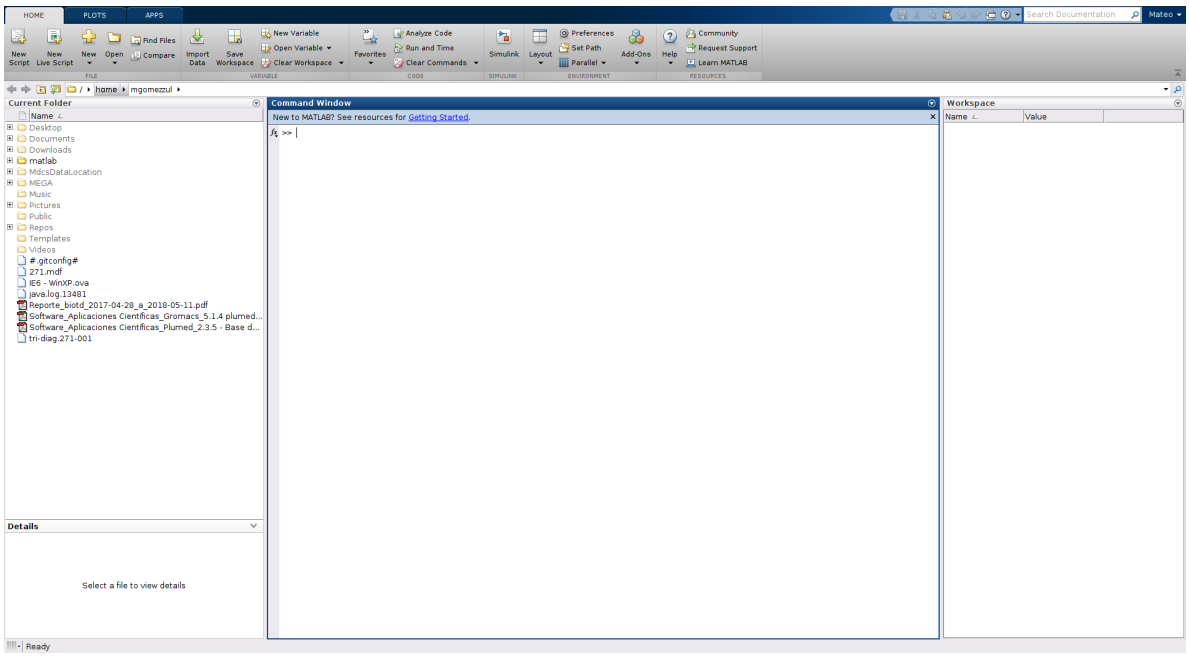

All changes take effect immediately.
Add Folder...
Add with Subfolders...
Move to Top
Move Up
Move Down
Move to Bottom
Remove
MATLAB search path:
/usr/local/MATLAB/R2018a/toolbox/matlab/elfun
/usr/local/MATLAB/R2018a/toolbox/matlab/strfun
/usr/local/MATLAB/R2018a/toolbox/matlab/datafun
/usr/local/MATLAB/R2018a/toolbox/matlab/polyfun
/usr/local/MATLAB/R2018a/toolbox/matlab/timefun
/usr/local/MATLAB/R2018a/toolbox/matlab/validators
/usr/local/MATLAB/R2018a/toolbox/matlab/general
/usr/local/MATLAB/R2018a/toolbox/matlab/elmat
/usr/local/MATLAB/R2018a/toolbox/matlab/funfun
/usr/local/MATLAB/R2018a/toolbox/matlab/matfun
/usr/local/MATLAB/R2018a/toolbox/matlab/sparfun
/usr/local/MATLAB/R2018a/toolbox/matlab/specfun
/usr/local/MATLAB/R2018a/toolbox/matlab/randfun
/usr/local/MATLAB/R2018a/toolbox/matlab/graphfun
/usr/local/MATLAB/R2018a/toolbox/local
/usr/local/MATLAB/R2018a/toolbox/matlab/testframework/perforn
/usr/local/MATLAB/R2018a/toolbox/matlab/testframework/measu
/usr/local/MATLAB/R2018a/toolbox/matlab/datastoreio
Save
Close
Revert
Default
Help


Look In:
matlab-integration
apolo
Folder name:
/home/mgomezzul/Documents/matlab-integration/apolo
Files of Type:
All Files
Open
Cancel



Configuring cluster profiles

1. Open again your MATLAB client (without admin privileges)



2. Configure MATLAB to run parallel jobs on your cluster by calling `configCluster` .

```
>> configCluster
Cluster FQDN (e.g. apolo.eafit.edu.co): cronos.eafit.edu.co
Username on Apolo (e.g. mgomezz): mgomezzul

>> % Must set TimeLimit before submitting jobs to Apolo II or
>> % Cronos cluster
```

(continues on next page)

(continued from previous page)

```
>> % e.g. to set the TimeLimit and Partition
>> c = parcluster('apolo remote R2018a');
>> c.AdditionalProperties.TimeLimit = '1:00:00';
>> c.AdditionalProperties.Partition = 'longjobs';
>> c.saveProfile

>> % e.g. to set the NumGpus, TimeLimit and Partition
>> c = parcluster('apolo remote R2018a');
>> c.AdditionalProperties.TimeLimit = '1:00:00';
>> c.AdditionalProperties.Partition = 'accel';
>> c.AdditionalProperties.NumGpus = 2;
>> c.saveProfile
```

3. Custom options

- `TimeLimit` Set a limit on the total run time of the job allocation (more info).
  - e.g. `c.AdditionalProperties.TimeLimit = '3-10:00:00';`
- `AccountName` Change the default user account on Slurm.
  - e.g. `c.AdditionalProperties.AccountName = 'apolo';`
- `ClusterHost` Another way to change the cluster hostname to sumbit jobs.
  - e.g. `c.AdditionalProperties.ClusterHost = 'apolo.eafit.edu.co';`
- `EmailAddress` Get all job notifications by e-mail.
  - e.g. `c.AdditionalProperties.EmailAddress = 'apolo@eafit.edu.co';`
- `EmailType` Get only the desired notifications based on sbatch options.
  - e.g. `c.AdditionalProperties.EmailType = 'END,TIME_LIMIT_50';`
- `MemUsage` Total amount of memory **per machine** (more info).
  - e.g. `c.AdditionalProperties.MemUsage = '5G';`
- `NumGpus` Number of GPUs (double) to use in a job.
  - e.g. `c.AdditionalProperties.NumGpus = 2;`

Note: The maximum value for `NumGpus` is two, also if you select this option you should use the *'accel'* partition on *Apolo II*.

- `Partition` Select the desire partition to submit jobs (by default *longjobs* partition will be used)
  - e.g. `c.AdditionalProperties.Partition = 'bigmem';`
- `Reservation` Submit a job into a reservation (more info).
  - e.g. `c.AdditionalProperties.Reservation = 'reservationName';`
- `AdditionalSubmitArgs` Any valid sbatch parameter (raw) (more info)
  - e.g. `c.AdditionalProperties.AdditionalSubmitArgs = '-no-requeue';`

Submitting jobs

General steps

1. Load *'apolo remote R2018a'* cluster profile and load the desired properties to submit a job.

```
>> % Run cluster configuration
>> configCluster

Cluster FQDN (e.g. apolo.eafit.edu.co): cronos.eafit.edu.co
Username on Apolo (e.g. mgomezz): mgomezzul

>> c = parcluster('apolo remote R2018a');
>> c.AdditionalProperties.TimeLimit = '1:00:00';
>> c.AdditionalProperties.Partition = 'longjobs';
>> c.saveProfile
```

2. To see the values of the current configuration options, call the specific `AdditionalProperties` method.

```
>> % To view current properties
>> c.AdditionalProperties
```

3. To clear a value, assign the property an empty value (`''`, `[]`, or `false`).

```
>> % Turn off email notifications
>> c.AdditionalProperties.EmailAddress = '';
```

4. If you have to cancel a job (queued or running) type.

```
>> j.cancel
```

5. Delete a job after results are no longer needed.

```
>> j.delete
```

Serial jobs

1. Use the batch command to submit asynchronous jobs to the cluster. The batch command will return a job object which is used to access the output of the submitted job.

   (See the MATLAB documentation for more help on batch.)

Listing 3: `serial_example.m`

```
function t = serial_example(n)

t0 = tic;
A = 500;
a = zeros(n);

for i = 1:n
    a(i) = max(abs(eig(rand(A))));
end

t = toc(t0);
```

(continues on next page)

(continued from previous page)

```
end
```

```
>> % Get a handle to the cluster
>> c = parcluster('apolo remote R2018a');

>> % Submit job to query where MATLAB is running on the cluster (script)
>> j = c.batch(@serial_example, 1, {1000});

>> % Query job for state
>> j.State

>> % Load results
>> j.fetchOutputs{:}

>> % Delete the job after results are no longer needed
>> j.delete
```

2. To retrieve a list of currently running or completed jobs, call `parcluster` to retrieve the cluster object. The cluster object stores an array of jobs that were run, are running, or are queued to run. This allows us to fetch the results of completed jobs. Retrieve and view the list of jobs as shown below.

```
>> c = parcluster('apolo remote R2018a');
>> jobs = c.Jobs
```

3. Once we have identified the job we want, we can retrieve the results as we have done previously. `fetchOutputs` is used to retrieve function output arguments; if using batch with a script, use `load` instead.

   Data that has been written to files on the cluster needs be retrieved directly from the file system. To view results of a previously completed job:

```
>> % Get a handle on job with ID 2
>> j2 = c.Jobs(2);
>> j2.fetchOutputs{:}
```

Note: You can view a list of your jobs, as well as their IDs, using the above `c.Jobs` command.

4. Another example using a MATLAB script.

Listing 4: `serial_example_script.m`

```
t0 = tic;
A = 500;
a = zeros(100);
fileID = fopen('/home/mgomezzul/time.txt','wt');
for i = 1:100
    a(i) = max(abs(eig(rand(A))));
end
t = toc(t0);
fprintf(fileID, '%6.4f\n', t);
fclose(fileID);
```

- Job submission

```
>> % Get a handle to the cluster
>> c = parcluster('apolo remote R2018a');

>> % Submit job to query where MATLAB is running on the cluster (script)
>> j = c.batch('serial_example_script');

>> % Query job for state
>> j.State

>> %Load results into the client workspace
>> j.load

>> % Delete the job after results are no longer needed
>> j.delete
```

5. Another example using a MATLAB script that supports GPU.

Listing 5: `gpu_script.m`

```
maxIterations = 500;
gridSize = 1000;
xlim = [-0.748766713922161, -0.748766707771757];
ylim = [ 0.123640844894862,  0.123640851045266];

% Setup
t = tic();
x = gpuArray.linspace( xlim(1), xlim(2), gridSize );
y = gpuArray.linspace( ylim(1), ylim(2), gridSize );
[xGrid,yGrid] = meshgrid( x, y );
z0 = complex( xGrid, yGrid );
count = ones( size(z0), 'gpuArray' );

% Calculate
z = z0;
for n = 0:maxIterations
          z = z.*z + z0;
inside = abs( z )<=2;
count = count + inside;
end
count = log( count );

% Show
count = gather( count ); % Fetch the data back from the GPU
naiveGPUTime = toc( t );
fig = gcf;
fig = figure('visible', 'off');
fig.Position = [200 200 600 600];
imagesc( x, y, count );
colormap( [jet();flipud( jet() );0 0 0] );
axis off;
title( sprintf( '%1.2fsecs (GPU)', naiveGPUTime ) );
saveas(gcf,'/home/mgomezzul/GPU.png');
```

- Job submission

```
>> % Get a handle to the cluster
>> c = parcluster('apolo remote R2018a');
```

(continues on next page)

(continued from previous page)

```
>> % Submit job to query where MATLAB is running on the cluster (script)
>> j = c.batch('gpu_script');

>> % Query job for state
>> j.State
```

6. Another example using Simulink via MATLAB.

Listing 6: `parsim_test_script.m`

```
% Example running a Simulink model.

% The Simulink model is called |parsim_test.slx| and it *must be* in
% the cluster.

% Number of simulations
numSims = 10;
W = zeros(1,numSims);

% Changing to the |parsim_test.slx| path
cd /home/mgomezzul/tests/matlab/slurm

% Create an array of |SimulationInput| objects and specify the argument value
% for each simulation. The variable |x| is the input variable in the Simulink
% model.
for x = 1:numSims
  simIn(x) = Simulink.SimulationInput('parsim_test');
  simIn(x) = setBlockParameter(simIn(x), 'parsim_test/Transfer Fcn', 'Denominator
↪', num2str(x));
end

% Running the simulations.
simOut = parsim(simIn);

% The variable |y| is the output variable in the Simulink model.
for x = 1:numSims
  W(1,x) = max(simOut(x).get('y'));
end

save('/home/mgomezzul/output_file.mat','W');
```

`parsim_test.slx (Simulink model)`

- Job submission

```
>> % Get a handle to the cluster
>> c = parcluster('apolo remote R2018a');

>> % Submit job to query where MATLAB is running on the cluster (script)
>> j = c.batch('parsim_test_script');

>> % Query job for state
>> j.State

>> % Load data to client workspace
>> j.load
```

Parallel or distributed jobs

Users can also submit parallel or distributed workflows with batch command. Let's use the following example for a parallel job.

Listing 7: `parallel_example.m`

```
function t = parallel_example(n)

t0 = tic;
A = 500;
a = zeros(n);
parfor i = 1:n
  a(i) = max(abs(eig(rand(A))));
end
t = toc(t0);
end
```

1. We will use the batch command again, but since we are running a parallel job, we will also specify a MATLAB pool.

```
>> % Get a handle to the cluster
>> c = parcluster('apolo remote R2018a');

>> % Submit a batch pool job using 4 workers
>> j = c.batch(@parallel_example, 1, {1000}, 'Pool', 4);

>> % View current job status
>> j.State

>> % Fetch the results after a finished state is retrieved
>> j.fetchOutputs{:}
ans =
   41.7692
```

   - The job ran in 41.7692 seconds using 4 workers.

Note: Note that these jobs will always request N+1 CPU cores, since one worker is required to manage the batch job and pool of workers. For example, a job that needs eight workers will consume nine CPU cores.

Note: For some applications, there will be a diminishing return when allocating too many workers, as the overhead may exceed computation time (communication).

2. We will run the same simulation, but increase the pool size. This time, to retrieve the results later, we will keep track of the job ID.

```
>> % Get a handle to the cluster
>> c = parcluster('apolo remote R2018a');

>> % Submit a batch pool job using 8 workers
>> j = c.batch(@parallel_example, 1, {1000}, 'Pool', 8);

>> % Get the job ID
>> id = j.ID
```

(continues on next page)

(continued from previous page)

```
Id =
4
>> % Clear workspace, as though we quit MATLAB
>> clear
```

3. Once we have a handle to the cluster, we will call the `findJob` method to search for the job with the specified job ID.

```
>> % Get a handle to the cluster
>> c = parcluster('apolo remote R2018a');

>> % Find the old job
>> j = c.findJob('ID', 4);

>> % Retrieve the state of the job
>> j.State
ans
finished
>> % Fetch the results
>> j.fetchOutputs{:}
ans =
22.2082
```

   - The job now runs 22.2082 seconds using 8 workers.

     Run code with different number of workers to determine the ideal number to use.

4. Another example using a parallel script.

Listing 8: `parallel_example_script.m`

```
n = 1000;
t0 = tic;
A = 500;
a = zeros(n);
fileID = fopen('/home/mgomezzul/time.txt','wt');

parfor i = 1:n
  a(i) = max(abs(eig(rand(A))));
end

t = toc(t0);
fprintf(fileID, '%6.4f\n', t);
fclose(fileID);
```

```
>> % Get a handle to the cluster
>> c = parcluster('apolo remote R2018a');

>> % Submit job to query where MATLAB is running on the cluster (script)
>> j = c.batch('parallel_example_script', 'Pool', 8);

>> % Query job for state
```

(continues on next page)

(continued from previous page)

```
>> j.State

>> %Load results
>> j.load

>> % Delete the job after results are no longer needed
>> j.delete
```

Debugging

1. If a serial job produces an error, we can call the `getDebugLog` method to view the error log file using *j.Tasks(1)*. Additionally when submitting independent jobs, with multiple tasks, you will have to specify the task number.

```
>> % If necessary, retrieve output/error log file
>> j.Parent.getDebugLog(j.Tasks(1))
```

2. For pool jobs, do not diference into the job object.

```
>> % If necessary, retrieve output/error log file
>> j.Parent.getDebugLog(j)
>> % or
>> c.getDebugLog(j)
```

3. To get information about the job in SLURM, we can consult the scheduler ID by calling `schedID`.

```
>> schedID(j)
   ans =
      25539
```

MDCS using cluster's MATLAB client

Submitting jobs from within MATLAB client on the cluster

General steps

1. Connect to Apolo II or Cronos via SSH.

```
# Without graphical user interface
ssh username@cronos/apolo.eafit.edu.co
# or with graphical user interface
ssh -X username@cronos/apolo.eafit.edu.co
```

2. Load MATLAB modufile.

```
module load matlab/r2018a
```

3. Run MATLAB client

```
matlab
```

4. First time, you have to define the cluster profile running the following command.

```
configCluster
```

5. Load *'apolo'* or *'cronos'* cluster profile and load the desired properties to submit a job (MATLAB GUI or command line).

```
>> % Must set TimeLimit before submitting jobs to Apolo II or
>> % Cronos cluster

>> % e.g. to set the TimeLimit and Partition
>> c = parcluster('apolo/cronos');
>> c.AdditionalProperties.TimeLimit = '1:00:00';
>> c.AdditionalProperties.Partition = 'longjobs';
>> c.saveProfile
>> % or
>> % e.g. to set the NumGpus, TimeLimit and Partition
>> c = parcluster('apolo');
>> c.AdditionalProperties.TimeLimit = '1:00:00';
>> c.AdditionalProperties.Partition = 'accel';
>> c.AdditionalProperties.NumGpus = '2';
>> c.saveProfile
```

6. To see the values of the current configuration options, call the specific `AdditionalProperties` method.

```
>> % To view current properties
>> c.AdditionalProperties
```

7. To clear a value, assign the property an empty value (`''`, `[]`, or `false`).

```
>> % Turn off email notifications
>> c.AdditionalProperties.EmailAddress = '';
```

Submitting jobs

Note: Users can submit serial, parallel or distributed jobs with batch command as the previous examples.

Submitting jobs directly through SLURM

MDCS jobs could be submitted directly from the Unix command line through SLURM.

For this, in addition to the MATLAB source, one needs to prepare a MATLAB submission script with the job specifications.

1. An example is shown below:

Listing 9: `matlab_batch.m`

```
%========================================================
% MATLAB job submission script: matlab_batch.m
%========================================================

workers = str2num(getenv('SLURM_NTASKS'));
c = parcluster('apolo');
```

(continues on next page)

(continued from previous page)

```
c.AdditionalProperties.TimeLimit = '1:00:00';
c.AdditionalProperties.Partition = 'longjobs';
j = c.batch(@parallel_example_slurm, 1, {1000}, 'pool', workers);
exit;
```

Listing 10: `parallel_example_slurm.m`

```
function t = parallel_example_slurm(n)

t0 = tic;
A = 500;
a = zeros(n);

parfor i = 1:n
  a(i) = max(abs(eig(rand(A))));
end

t = toc(t0);
save prueba.txt t -ascii

end
```

2. It is submitted to the queue with help of the following SLURM batch-job submission script:

Listing 11: `matlab.slurm`

```
#!/bin/bash

#SBATCH -J test_matlab
#SBATCH -o test_matlab-%j.out
#SBATCH -e test_matlab-%j.err
#SBATCH -p longjobs
#SBATCH -n 8
#SBATCH -t 20:00

module load matlab/r2018a

matlab -nosplash -nodesktop -r "matlab_batch"
```

3. Job is submitted as usual with:

```
sbatch matlab.slurm
```

Note: This scheme dispatches 2 jobs - one serial that spawns the actual MDCS parallel jobs, and another, the actual parallel job.

---

4. Once submitted, the job can be monitored and managed directly through SLURM

   - `squeue` command output

```
 JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
326582  longjobs Gamma_1- jdavidca PD       0:00      1 (Priority)
326602  longjobs hexa298s ltenelan PD       0:00      1 (Priority)
326603  longjobs tetra599 ltenelan PD       0:00      1 (Priority)
326604  longjobs tricli91 ltenelan PD       0:00      1 (Priority)
326618  longjobs    RAxML apolo-jo PD       0:00      1 (Priority)
326619  longjobs  lattice  ohenaoz PD       0:00      1 (Priority)
326620  longjobs bayescan apolo-jo PD       0:00      1 (Priority)
326621  longjobs  lattice  ohenaoz PD       0:00      1 (Priority)
326580  longjobs alpha_1- jdavidca PD       0:00      1 (Resources)
326581  longjobs Beta-fre jdavidca PD       0:00      1 (Priority)
326583     accel edificio apolo-dr  R 1-01:14:26      1 compute-0-5
326578  longjobs    MED3P  maberce  R 1-19:49:23      2 compute-0-[1,3]
326534  longjobs BootInfl larteag7  R 4-19:12:58      2 compute-0-[0,2]
```

5. After the job completes, one can fetch results and delete job object from within MATLAB client on the cluster. If program writes directly to disk fetching is not necessary.

```
>> c = parcluster('apolo');
>> jobs = c.Jobs
>> j = c.Jobs(7);
>> j.fetchOutputs{:};
>> j.delete;
```

MATLAB directly on the cluster

Next steps describes how to use MATLAB and its toolboxes without MDCS (workers) toolbox, but this way has next pros and cons.

- **Pros**
  - No workers limitations
- **Cons**
  - No distributed jobs (Only parallel or serial jobs)

Unattended job

To run unattended jobs on the cluster follow next steps:

1. Connect to Apolo II or Cronos via SSH.

```
ssh username@cronos.eafit.edu.co
```

2. Enter to the matlab directory project.

```
cd ~/test/matlab/slurm
```

3. Create a SLURM batch-job submission script

Listing 12: `slurm.sh`

```
#!/bin/bash
```

(continues on next page)

(continued from previous page)

```
#SBATCH -J test_matlab
#SBATCH -o test_matlab-%j.out
#SBATCH -e test_matlab-%j.err
#SBATCH -p bigmem
#SBATCH -n 8
#SBATCH -t 20:00

module load matlab/r2018a

matlab -nosplash -nodesktop < parallel_example_unattended.m
```

Listing 13: `parallel_example_unattended.m`

```
p = parpool(str2num(getenv('SLURM_NTASKS')));

t0 = tic;
A = 500;
a = zeros(1000);
parfor i = 1:1000
  a(i) = max(abs(eig(rand(A))));
end
t = toc(t0)
exit
```

4. Submit the job.

```
sbatch slurm.sh
```

1. Check the `stdout` file (`test_matlab_xxxx.out`).

```
MATLAB is selecting SOFTWARE OPENGL rendering.

                      < M A T L A B (R) >
            Copyright 1984-2018 The MathWorks, Inc.
             R2018a (9.4.0.813654) 64-bit (glnxa64)
                        February 23, 2018


To get started, type one of these: helpwin, helpdesk, or demo.
For product information, visit www.mathworks.com.

>> Starting parallel pool (parpool) using the 'local' profile ...
connected to 8 workers.
>> >> >> >> >> >>
t =

  22.5327
```

Interactive job (No GUI)

If it is necessary the user can run interactive jobs following next steps:

1. Connect to Apolo II or Cronos via SSH.

```
ssh username@apolo.eafit.edu.co
```

2. Submit a interactive request to the resource manager

```
srun -N 1 --ntasks-per-node=2 -t 20:00 -p debug --pty bash
# If resources are available you get inmediatily a shell in a slave node
# e.g. compute-0-6
module load matlab/r2018a
matlab
```

```
MATLAB is selecting SOFTWARE OPENGL rendering.

                      < M A T L A B (R) >
            Copyright 1984-2018 The MathWorks, Inc.
            R2018a (9.4.0.813654) 64-bit (glnxa64)
                       February 23, 2018


To get started, type one of these: helpwin, helpdesk, or demo.
For product information, visit www.mathworks.com.

>> p = parpool(str2num(getenv('SLURM_NTASKS')));
Starting parallel pool (parpool) using the 'local' profile ...
>> p.NumWorkers

ans =

     2
```

Note: At this point you have an interactive MATLAB session through the resource manager (SLURM), giving you the possibility to test and check different MATLAB features.

3. To finish this job, you have to close the MATLAB session and then the bash session granted in the slave node.

References

- Parallel Computing Toolbox
- MATLAB Distributed Computing Server
- "Portions of our documentation contain content originally created by Harvard FAS Research Computing and adapted by us under the Creative Commons Attribution-NonCommercial 4.0 International License. More information: https://rc.fas.harvard.edu/about/attribution/"

Authors

- Mateo Gómez-Zuluaga <mgomezz@eafit.edu.co>

MATLAB - R2019a

Basic information

- **Deploy date:** 30 May 2019
- **Official Website:** https://www.mathworks.com
- **License:** Proprietary commercial software (Only for EAFIT academic usage)
- **End date:** 30 May 2020
- **Installed on:** *Apolo II*, *Cronos*
- **Available MATLAB toolboxes:** List

Installation

This entry covers the entire process performed for the installation and configuration of MATLAB and the MATLAB Parallel Server (formerly known as MATLAB Distributed Computing Server) on a clusters with the conditions described above.

Note: Matlab installation is shared between the 2 clusters (Apolo and Cronos), so the process is performed only one time.

Contents

Tested on (Requirements)

- **License Manager Server:** Virtual machine (CentOS 7 Minimal (x86_64))
- **OS base:** CentOS (x86_64) $\geq$ 6.6
- **MPI:** Intel MPI $\geq$ 19.0.4 (Mandatory to use with Infiniband networks)
- **Scheduler:** SLURM $\geq$ 18.08.1
- **Application:** MATLAB Client (Optional)
- **Extra Libraries:**
  - libXtst (*Troubleshooting*)

License Manager

The *License Manager* provides a network license support to allow the usage of the different MATLAB features on the clusters (Apolo II and Cronos).

In this case we have two types of licenses, the first one is for the MATLAB Parallel Server and the second one for MATLAB client with all the toolboxes available.

Next steps will describe the installation and configuration process for the MLM (MATLAB License Manager based on FlexLM[1]):

1. Get the online installer using your MATLAB account.
2. Send the installation package to the License Manager server (VM).

```
scp matlab_R2019a_glnxa64.zip root@<FQDN>:$installer_path
```

3. Follow the next steps to run the MATLAB installer.
    1. Unzip and access the installer files.

```
ssh -X root@<FQDN>
cd $installer_path$
mkdir matlab-R2019a
mv matlab_R2019a_glnxa64.zip matlab-R2019a
cd matlab-R2019a
unzip matlab_R2019a_glnxa64.zip
```

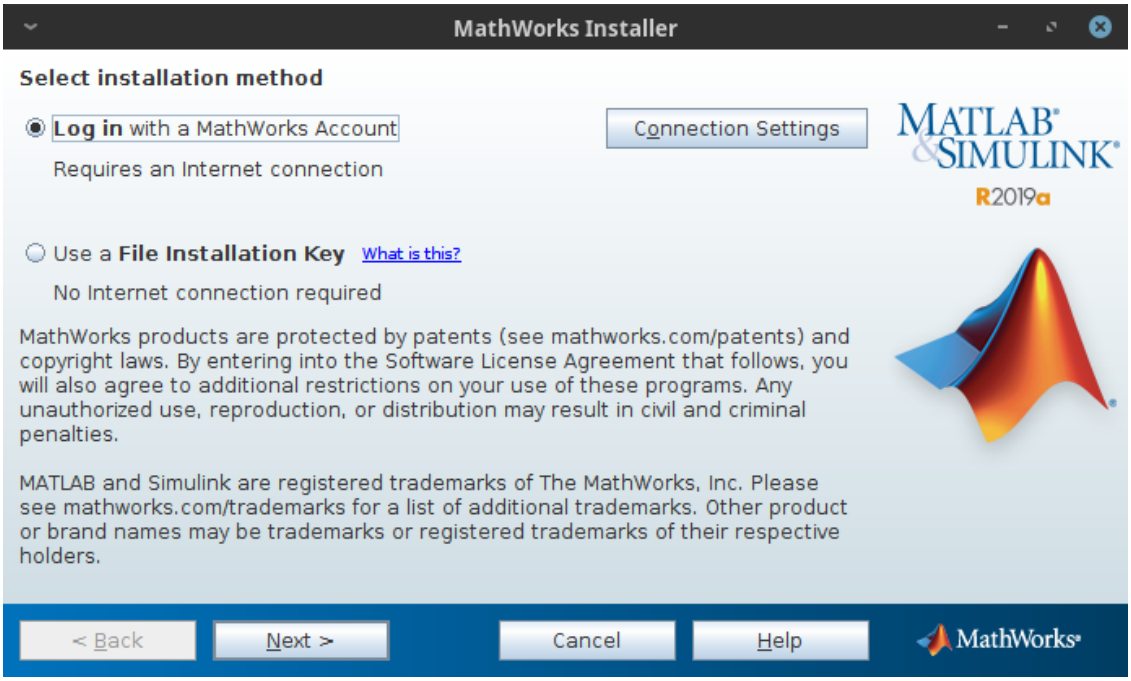
    2. Execute the installer.

```
./install
```

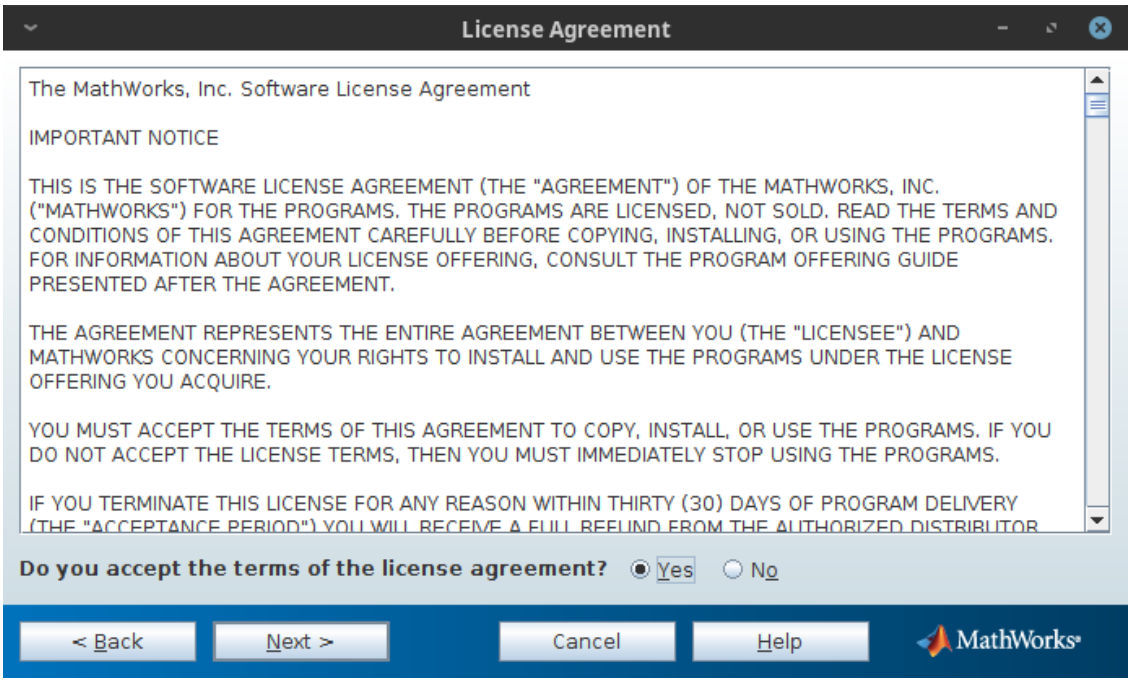
> **Note:** *Troubleshooting*

    3. Select the installation method (by MATLAB account).

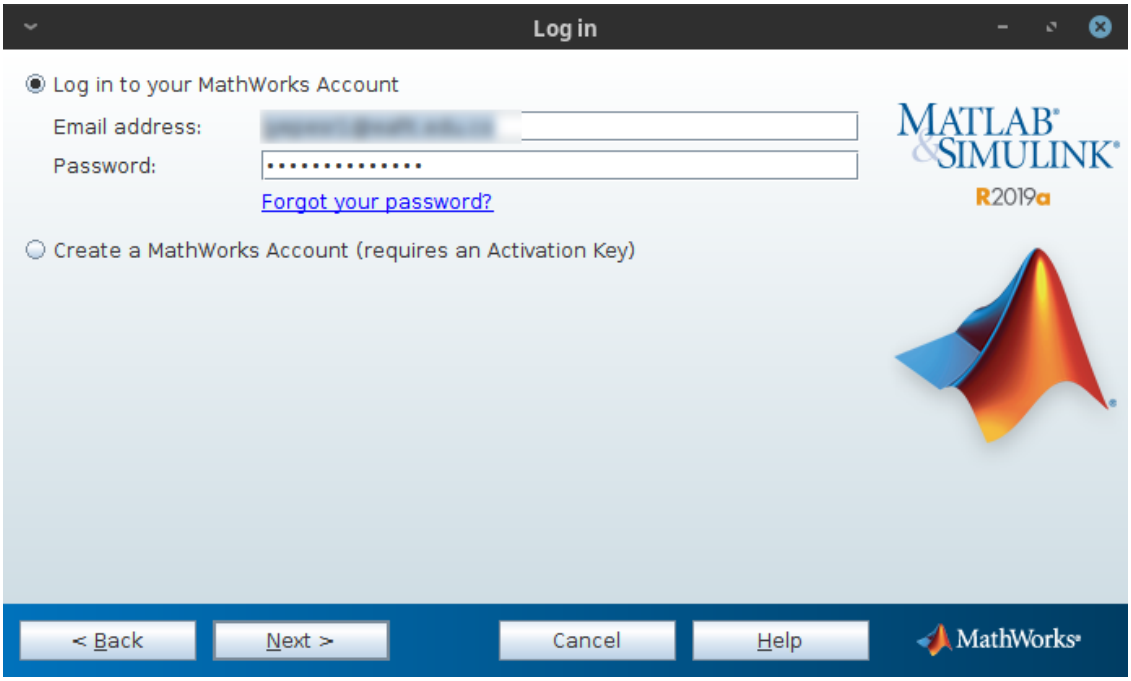
[1] Wikipedia contributors. (2018, April 13). FlexNet Publisher. In Wikipedia, The Free Encyclopedia. Retrieved 20:44, July 18, 2018, from https://en.wikipedia.org/w/index.php?title=FlexNet_Publisher&oldid=836261861



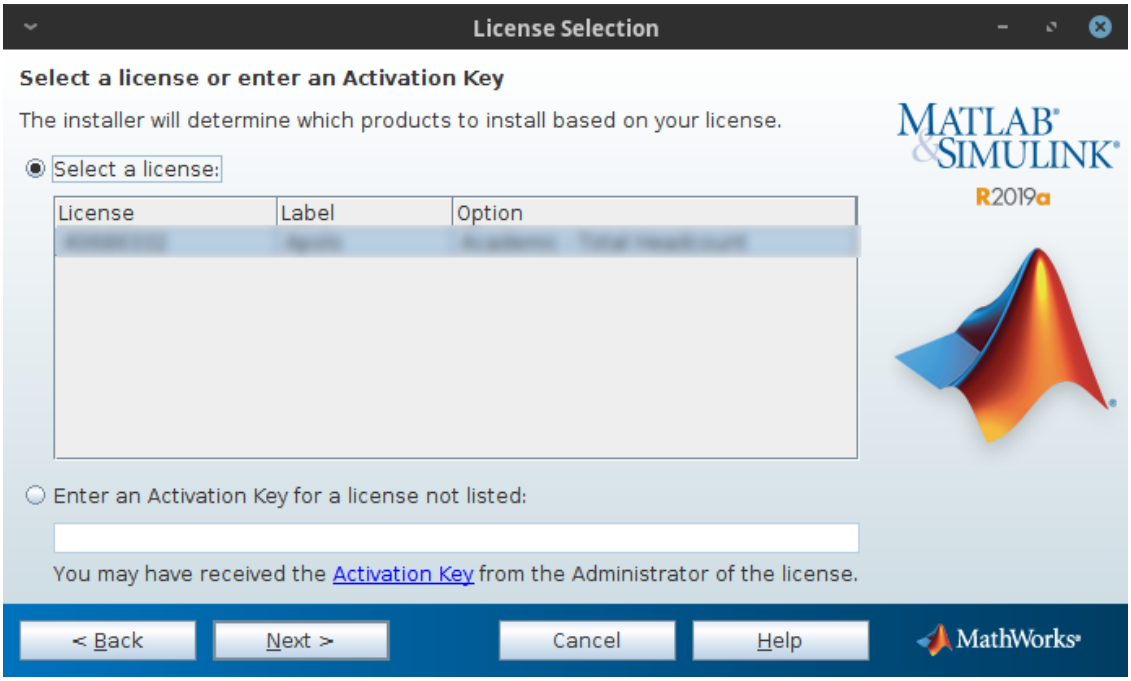
4. Accept license agreement (yes).



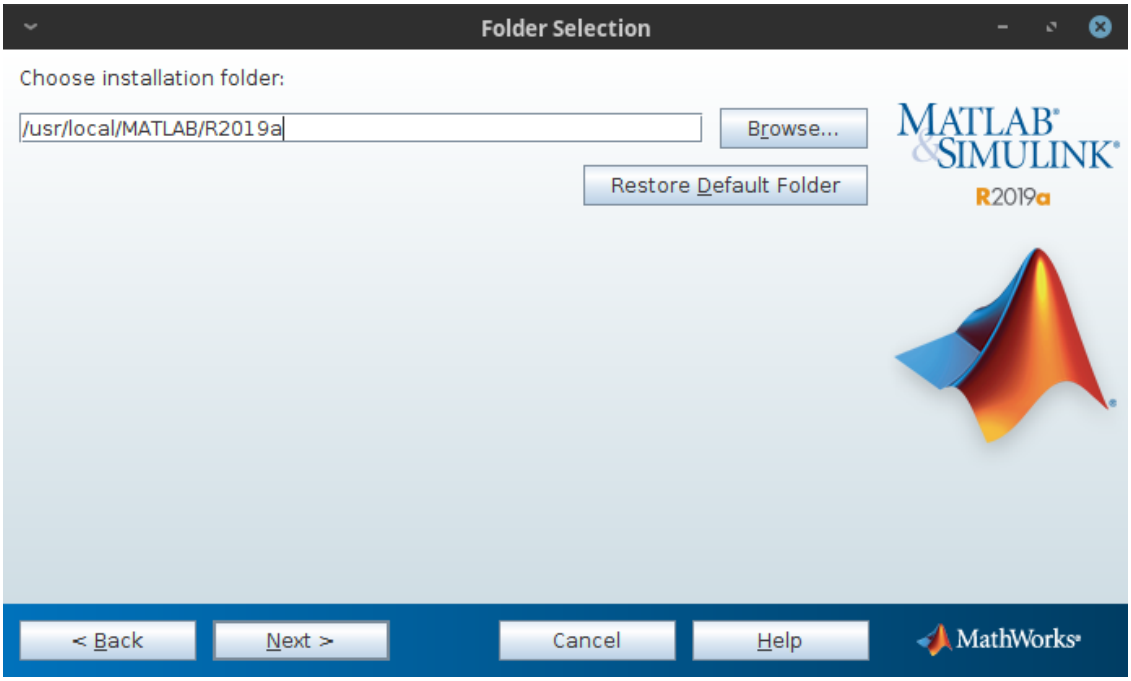
5. Login (username and password).



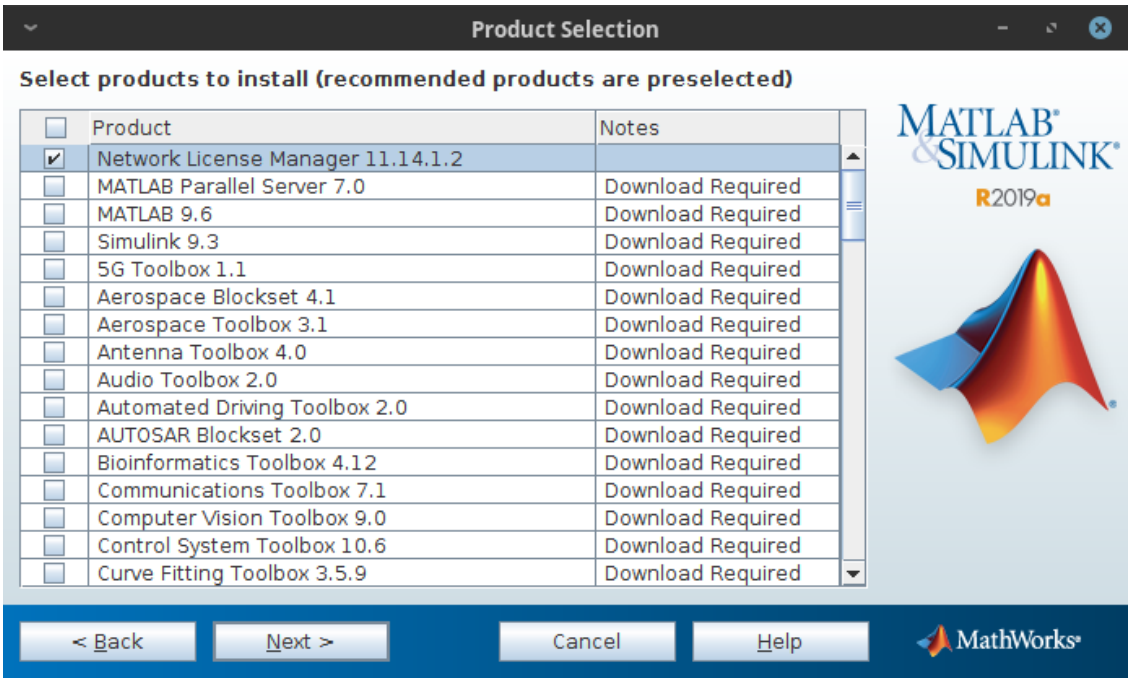
6. Select license (Parallel Server license).



7. Folder selection (/usr/local/MATLAB/R2019a).



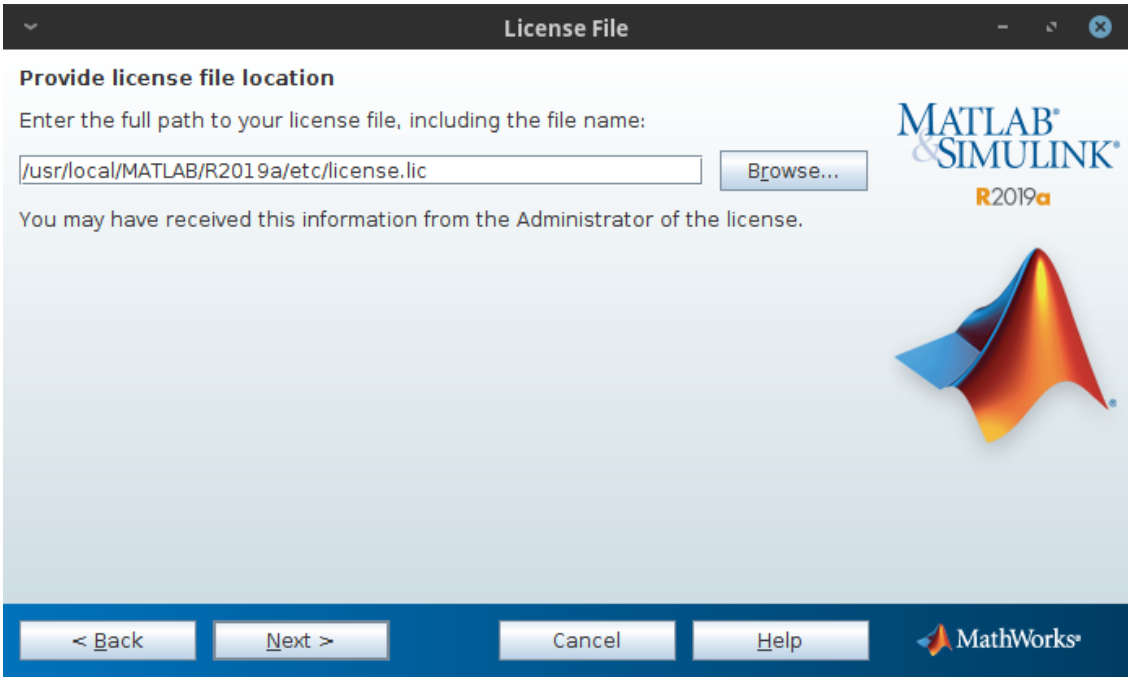
8. Products selection (License Manager 11.14.1.2).



9. License file.

Note: Login to the MATLAB admin account and download the license file (*license.lic*) created for the MATLAB Parallel Server (MATLAB Distribuited Computing Engine) and upload it to the *License Manager* server in the `/usr/local/MATLAB/R2019a/etc` directory.

```
scp license.lic root@<FQDN>:/usr/local/MATLAB/R2019a/etc
```



10. Finish the installation process.

4. Configure MLM (FlexLM).

    1. Access the *License Manager* machine via **SSH**.
    2. Create a system user without privileges to run MLM.

```
# Create a non-root user to launch matlab (security reasons)
## -u uid
## -d homedir
## -r system user
## -s shell (no login user)
useradd -u 110 -c "MDCE" -d /var/tmp -r -s /sbin/nologin matlab
```

    3. Create the daemon service to execute automatically MLM.

Listing 14: `lm-matlab.service`

```
[Unit]
Description=MATLAB FlexLM license manager

[Service]
User=matlab
RemainAfterExit=True
ExecStart=/usr/local/MATLAB/R2018a/etc/lmstart
ExecStop=/usr/local/MATLAB/R2018a/etc/lmdown

[Install]
WantedBy=multi-user.target
```

    4. Configure MLM ports and firewall on the license manager machine.

Note: After the installation process, the MLM generates a new license file called `license.dat` on the `/usr/local/MATLAB/R2019a/etc` directory with the information given in `license.lic` file

during the installation process and some additional information showing at following.

- Review the server port (27000) and specify MLM daemon port (53200) at the top of the license file (`/usr/local/MATLAB/R2019a/etc/license.dat`)

```
SERVER <HOSTNAME> <HOSTID> 27000
DAEMON MLM "/usr/local/MATLAB/R2019a/etc/MLM" port=53200
...
```

> **Warning:** If you do NOT have this file, please create it and add the above mentioned information.

- Open those ports in License manager machine firewall (CentOS 7).

```
firewall-cmd --permanent --add-port=53200/tcp
firewall-cmd --permanent --add-port=27000/tcp
```

5. Configure both licenses (Parallel Server and MATLAB client with all the toolboxes).
   - Download the `license.lic` file related with MATLAB client and its toolboxes from the MATLAB administrator account, then open it with a text editor to copy all the **INCREMENTS** lines.
   - Append all (MATLAB client and its toolboxes) **INCREMENTS** lines (licensed products) to end of the `license.dat` on the *License Manager* server.

```
SERVER <FQDN> <HOSTID> 27000
DAEMON MLM "/usr/local/MATLAB/R2019a/etc/MLM" port=53200
# BEGIN--------------BEGIN--------------BEGIN
# MathWorks license passcode file.
# LicenseNo: ########   HostID: ############
#
# R2019a
#
INCREMENT MATLAB_Distrib_Comp_Engine MLM 39 <END_DATE> <NUM_WORKES> \
...
INCREMENT MATLAB MLM 39 <END_DATE> ##################### \
...
INCREMENT SIMULINK MLM 39 <END_DATE> ##################### \
...
... continue ...
...
```

6. Enable and start the daemon.

```
systemctl enable lm-matlab
systemctl start  lm-matlab
```

7. Check the log file to see if everything works properly. `/var/tmp/lm_TMW.log`

```
10:55:13 (lmgrd) -----------------------------------------------
10:55:13 (lmgrd)   Please Note:
10:55:13 (lmgrd)
10:55:13 (lmgrd)   This log is intended for debug purposes only.
10:55:13 (lmgrd)   In order to capture accurate license
10:55:13 (lmgrd)   usage data into an organized repository,
10:55:13 (lmgrd)   please enable report logging. Use Flexera Software LLC's
```

(continues on next page)

(continued from previous page)

```
10:55:13 (lmgrd)   software license administration  solution,
10:55:13 (lmgrd)   FlexNet Manager, to  readily gain visibility
10:55:13 (lmgrd)   into license usage data and to create
10:55:13 (lmgrd)   insightful reports on critical information like
10:55:13 (lmgrd)   license availability and usage. FlexNet Manager
10:55:13 (lmgrd)   can be fully automated to run these reports on
10:55:13 (lmgrd)   schedule and can be used to track license
10:55:13 (lmgrd)   servers and usage across a heterogeneous
10:55:13 (lmgrd)   network of servers including Windows NT, Linux
10:55:13 (lmgrd)   and UNIX.
10:55:13 (lmgrd)
10:55:13 (lmgrd) ------------------------------------------------
10:55:13 (lmgrd)
10:55:13 (lmgrd)
10:55:13 (lmgrd) Server's System Date and Time: Wed Jul 17 2019 10:55:13 -05
10:55:13 (lmgrd) SLOG: Summary LOG statistics is enabled.
10:55:13 (lmgrd) FlexNet Licensing (v11.14.1.2 build 208719 x64_lsb) started␣
↪on <FQDN> (linux) (7/17/2019)
10:55:13 (lmgrd) Copyright (c) 1988-2017 Flexera Software LLC. All Rights␣
↪Reserved.
10:55:13 (lmgrd) World Wide Web:  http://www.flexerasoftware.com
10:55:13 (lmgrd) License file(s): /var/tmp/lm_TMW.dat
10:55:13 (lmgrd) lmgrd tcp-port 27000
10:55:13 (lmgrd) (@lmgrd-SLOG@)␣
↪============================================
10:55:13 (lmgrd) (@lmgrd-SLOG@) === LMGRD ===
10:55:13 (lmgrd) (@lmgrd-SLOG@) Start-Date: Wed Jul 17 2019 10:55:13 -05
10:55:13 (lmgrd) (@lmgrd-SLOG@) PID: 14118
10:55:13 (lmgrd) (@lmgrd-SLOG@) LMGRD Version: v11.14.1.2 build 208719 x64_
↪lsb ( build 208719 (ipv6))
10:55:13 (lmgrd) (@lmgrd-SLOG@)
10:55:13 (lmgrd) (@lmgrd-SLOG@) === Network Info ===
10:55:13 (lmgrd) (@lmgrd-SLOG@) Listening port: 27000
10:55:13 (lmgrd) (@lmgrd-SLOG@)
10:55:13 (lmgrd) (@lmgrd-SLOG@) === Startup Info ===
10:55:13 (lmgrd) (@lmgrd-SLOG@) Server Configuration: Single Server
10:55:13 (lmgrd) (@lmgrd-SLOG@) Command-line options used at LS startup: -z -
↪c /var/tmp/lm_TMW.dat
10:55:13 (lmgrd) (@lmgrd-SLOG@) License file(s) used:  /var/tmp/lm_TMW.dat
10:55:13 (lmgrd) (@lmgrd-SLOG@)␣
↪============================================
10:55:13 (lmgrd) Starting vendor daemons ...
10:55:13 (lmgrd) Using vendor daemon port 53200 specified in license file
10:55:13 (lmgrd) Started MLM (internet tcp_port 53200 pid 14120)
10:55:13 (MLM) FlexNet Licensing version v11.14.1.2 build 208719 x64_lsb
10:55:13 (MLM) SLOG: Summary LOG statistics is enabled.
10:55:13 (MLM) SLOG: FNPLS-INTERNAL-CKPT1
10:55:13 (MLM) SLOG: VM Status: 0
10:55:13 (MLM) Server started on <FQDN> for:    MATLAB_Distrib_Comp_Engine
10:55:13 (MLM) MATLAB           SIMULINK        MATLAB_5G_Toolbox
10:55:13 (MLM) AUTOSAR_Blockset Aerospace_Blockset Aerospace_Toolbox
10:55:13 (MLM) Antenna_Toolbox Audio_System_Toolbox Automated_Driving_Toolbox
10:55:13 (MLM) Bioinformatics_Toolbox Communication_Toolbox Video_and_Image_
↪Blockset
10:55:13 (MLM) Control_Toolbox Curve_Fitting_Toolbox Signal_Blocks
10:55:13 (MLM) Data_Acq_Toolbox Database_Toolbox Datafeed_Toolbox
10:55:13 (MLM) Neural_Network_Toolbox Econometrics_Toolbox RTW_Embedded_Coder
```

(continues on next page)

(continued from previous page)

```
10:55:13 (MLM) Filter_Design_HDL_Coder Fin_Instruments_Toolbox Financial_
↪Toolbox
10:55:13 (MLM) Fixed_Point_Toolbox Fuzzy_Toolbox        GPU_Coder
10:55:13 (MLM) GADS_Toolbox     Simulink_HDL_Coder EDA_Simulator_Link
10:55:13 (MLM) Image_Acquisition_Toolbox Image_Toolbox  Instr_Control_Toolbox
10:55:13 (MLM) LTE_HDL_Toolbox LTE_Toolbox      MATLAB_Coder
10:55:13 (MLM) MATLAB_Builder_for_Java Compiler MATLAB_Report_Gen
10:55:13 (MLM) MAP_Toolbox      Mixed_Signal_Blockset MPC_Toolbox
10:55:13 (MLM) MBC_Toolbox      OPC_Toolbox     Optimization_Toolbox
10:55:13 (MLM) Distrib_Computing_Toolbox PDE_Toolbox    Phased_Array_System_
↪Toolbox
10:55:13 (MLM) Powertrain_Blockset Pred_Maintenance_Toolbox RF_Blockset
10:55:13 (MLM) RF_Toolbox       Reinforcement_Learn_Toolbox Risk_Management_
↪Toolbox
10:55:13 (MLM) Robotics_System_Toolbox Robust_Toolbox   Sensor_Fusion_and_
↪Tracking
10:55:13 (MLM) SerDes_Toolbox   Signal_Toolbox  SimBiology
10:55:13 (MLM) SimEvents        SimDriveline    Power_System_Blocks
10:55:13 (MLM) SimHydraulics    SimMechanics    Simscape
10:55:13 (MLM) Virtual_Reality_Toolbox SL_Verification_Validation Simulink_
↪Code_Inspector
10:55:13 (MLM) Real-Time_Workshop Simulink_Control_Design Simulink_Coverage
10:55:13 (MLM) Simulink_Design_Optim Simulink_Design_Verifier Real-Time_Win_
↪Target
10:55:13 (MLM) Simulink_PLC_Coder XPC_Target    SIMULINK_Report_Gen
10:55:13 (MLM) Simulink_Requirements Simulink_Test      SoC_Blockset
10:55:13 (MLM) Excel_Link       Stateflow       Statistics_Toolbox
10:55:13 (MLM) Symbolic_Toolbox System_Composer Identification_Toolbox
10:55:13 (MLM) Text_Analytics_Toolbox Trading_Toolbox Vehicle_Dynamics_
↪Blockset
10:55:13 (MLM) Vehicle_Network_Toolbox Vision_HDL_Toolbox WLAN_System_Toolbox
10:55:13 (MLM) Wavelet_Toolbox SimElectronics
10:55:13 (MLM) EXTERNAL FILTERS are OFF
10:55:14 (lmgrd) MLM using TCP-port 53200
10:55:14 (MLM) License verification completed successfully.
10:55:14 (MLM) SLOG: Statistics Log Frequency is 240 minute(s).
10:55:14 (MLM) SLOG: TS update poll interval is 600 seconds.
10:55:14 (MLM) SLOG: Activation borrow reclaim percentage is 0.
10:55:14 (MLM) (@MLM-SLOG@) ===============================================
10:55:14 (MLM) (@MLM-SLOG@) === Vendor Daemon ===
10:55:14 (MLM) (@MLM-SLOG@) Vendor daemon: MLM
10:55:14 (MLM) (@MLM-SLOG@) Start-Date: Wed Jul 17 2019 10:55:14 -05
10:55:14 (MLM) (@MLM-SLOG@) PID: 14120
10:55:14 (MLM) (@MLM-SLOG@) VD Version: v11.14.1.2 build 208719 x64_lsb (␣
↪build 208719 (ipv6))
10:55:14 (MLM) (@MLM-SLOG@)
10:55:14 (MLM) (@MLM-SLOG@) === Startup/Restart Info ===
10:55:14 (MLM) (@MLM-SLOG@) Options file used: None
10:55:14 (MLM) (@MLM-SLOG@) Is vendor daemon a CVD: No
10:55:14 (MLM) (@MLM-SLOG@) Is TS accessed: No
10:55:14 (MLM) (@MLM-SLOG@) TS accessed for feature load: -NA-
10:55:14 (MLM) (@MLM-SLOG@) Number of VD restarts since LS startup: 0
10:55:14 (MLM) (@MLM-SLOG@)
10:55:14 (MLM) (@MLM-SLOG@) === Network Info ===
10:55:14 (MLM) (@MLM-SLOG@) Listening port: 53200
10:55:14 (MLM) (@MLM-SLOG@) Daemon select timeout (in seconds): 1
10:55:14 (MLM) (@MLM-SLOG@)
```

(continues on next page)

(continued from previous page)

```
10:55:14 (MLM) (@MLM-SLOG@) === Host Info ===
10:55:14 (MLM) (@MLM-SLOG@) Host used in license file: <FQDN>
10:55:14 (MLM) (@MLM-SLOG@) Running on Hypervisor: Not determined - treat as␣
↪Physical
10:55:14 (MLM) (@MLM-SLOG@) ================================================
...
```

8. After that, the license manager service should run without problems, if there is any trouble with the service you can debug this process checking the log file (`/var/tmp/lm_TMW.log`) to understand what is happening.

```
tailf /var/tmp/lm_TMW.log
```

MATLAB Parallel Server

This entry described the installation process of MATLAB Parallel Server on the cluster and its integration with the *License Manager*.

1. Get the online installer using your MATLAB account.
2. Send the installation file to the master node on your cluster.

```
scp matlab_R2019a_glnxa64.zip root@<FQDN>:$installer_path$
```

3. Follow next steps to run the MATLAB installer.
   1. Unzip and access the installer files.

```
ssh -X root@<FQDN>
cd $installer_path$
mkdir matlab-R2019a
mv matlab_R2019a_glnxa64.zip matlab-R2019a
cd matlab-R2019a
unzip matlab_R2019a_glnxa64.zip
```

   2. Create the installation directory.

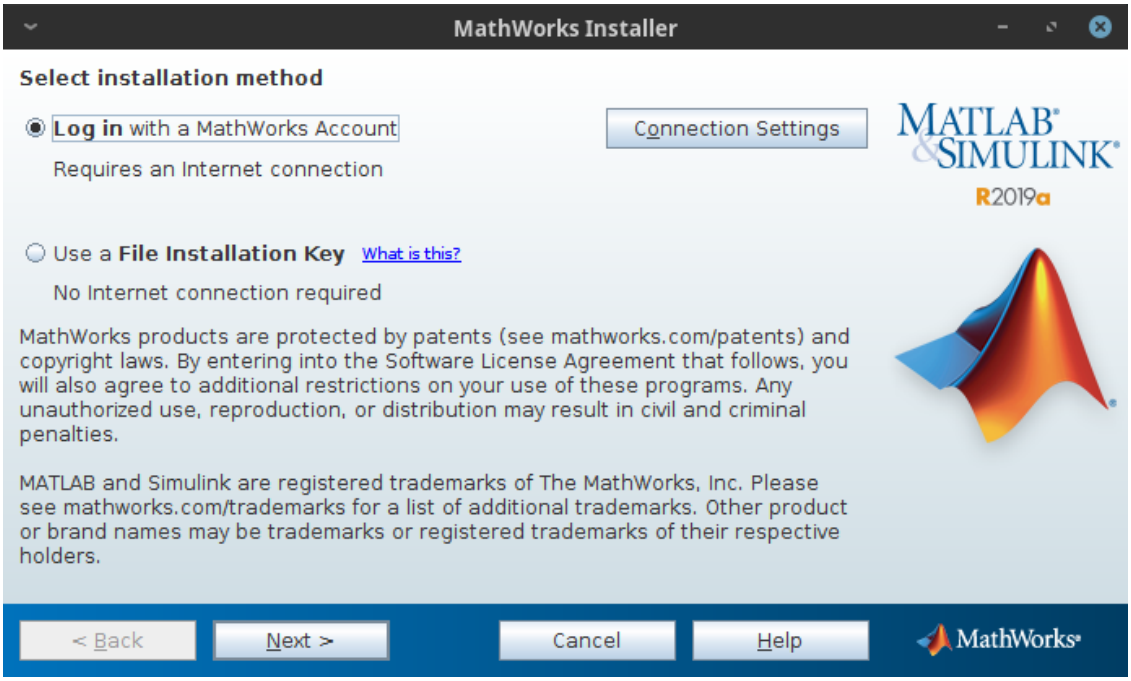
```
mkdir -p /share/apps/matlab/r2019a
```

   3. Execute the installer.

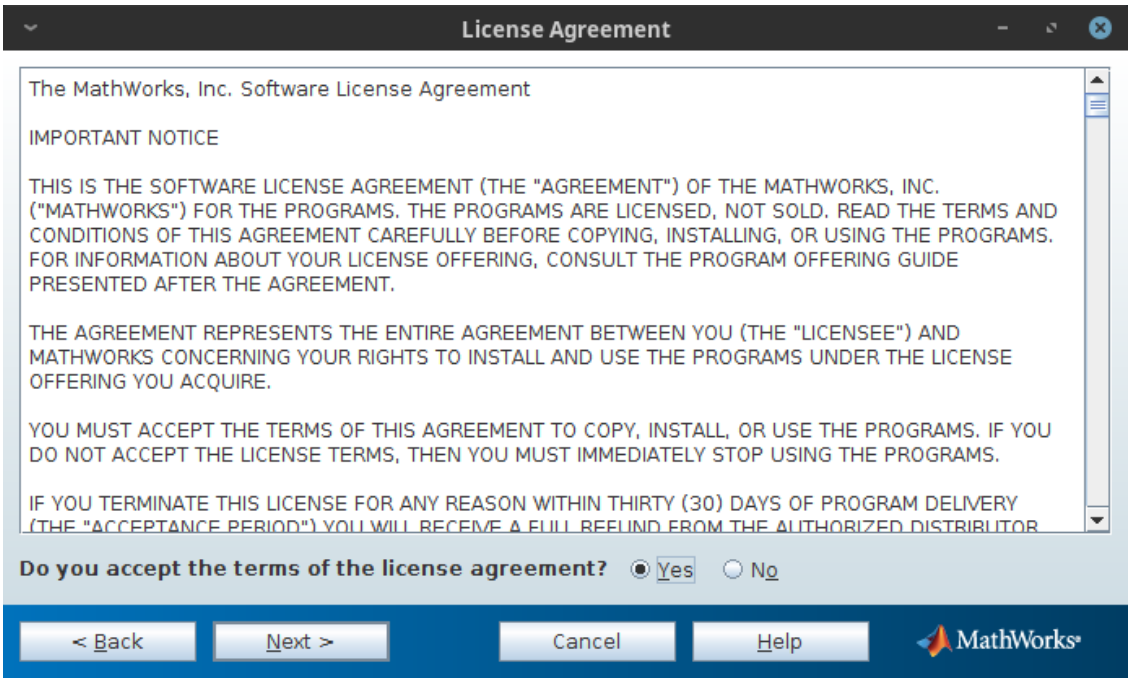
```
./install
```

> **Note:** *Troubleshooting*

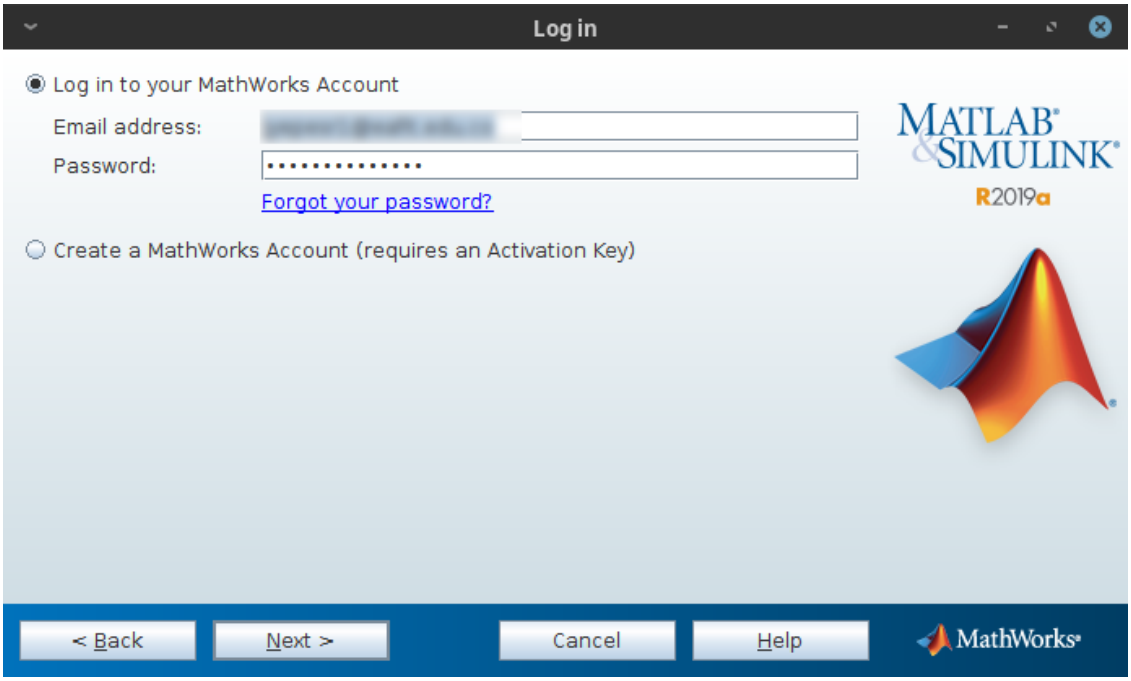
   4. Select the installation method (by MATLAB account).



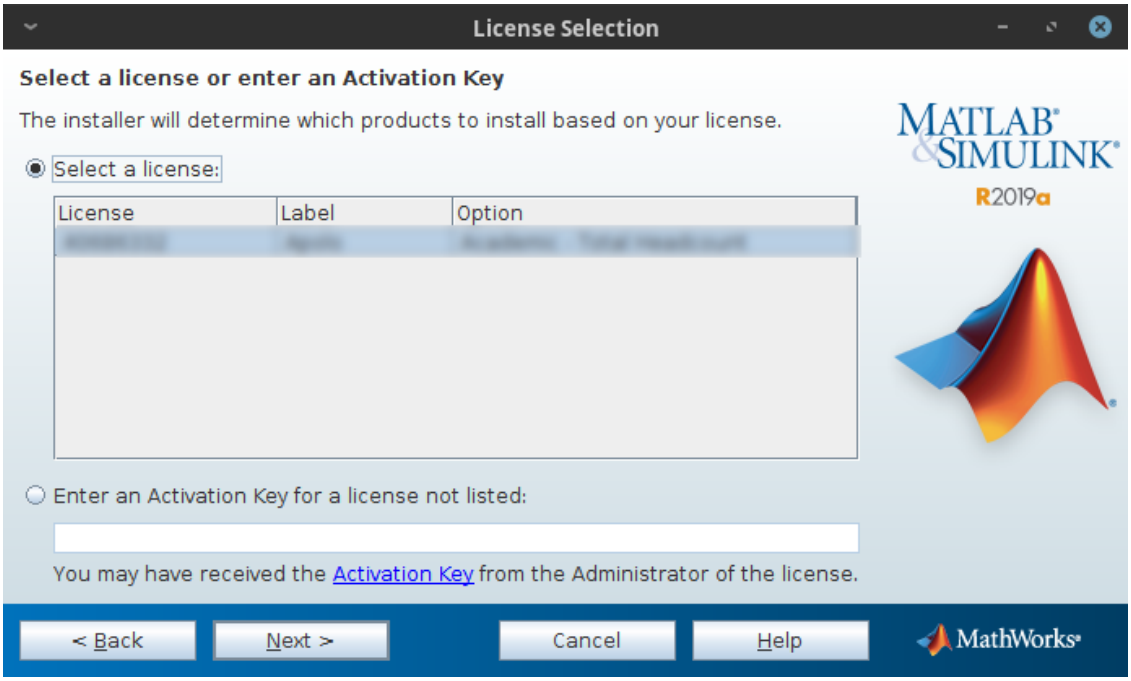
5. Accept license agreement (yes).



6. Login (username and password).

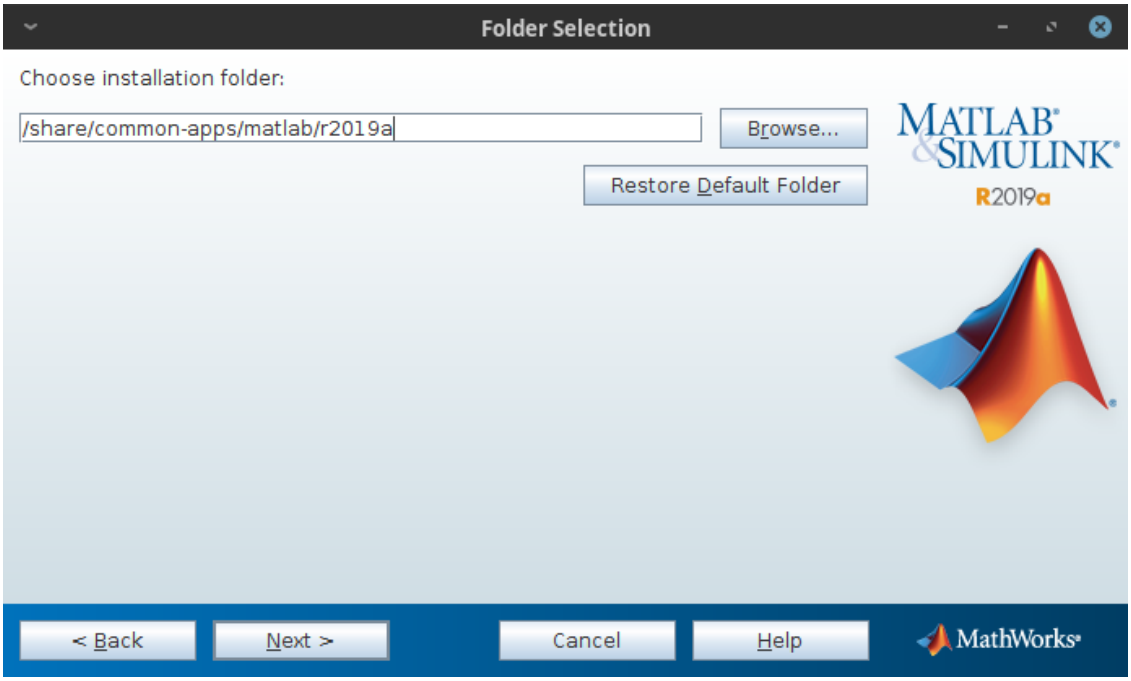


7. Select license (Parallel Server license).



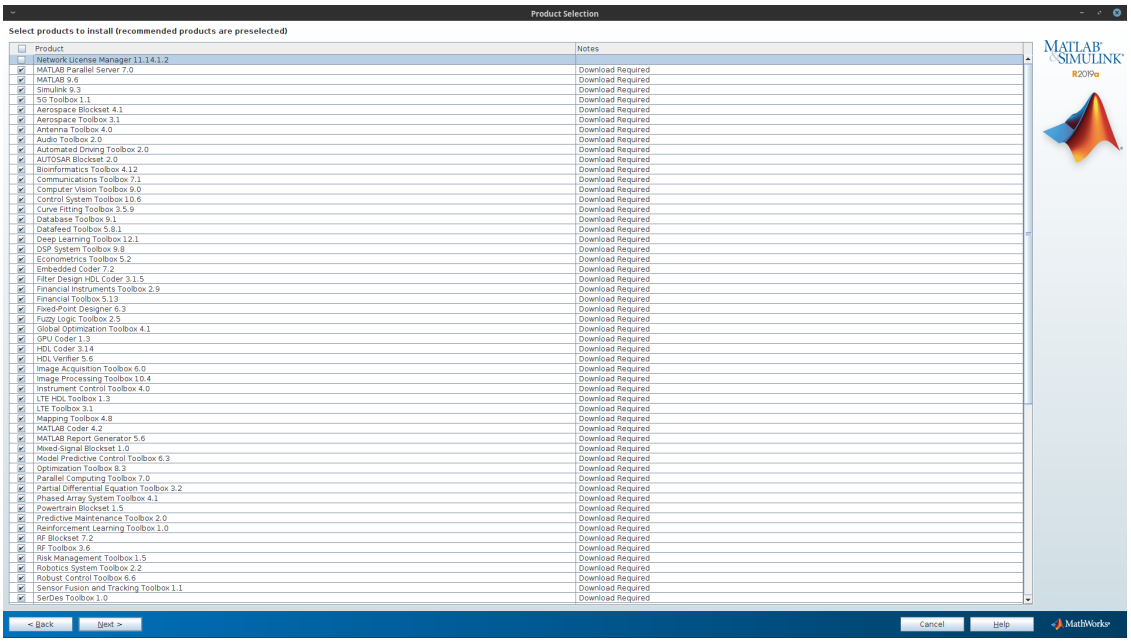
8. Folder selection (`/share/common-apps/matlab/r2019a`).

Note: Use a shared file system to do an unique installation across all the nodes and both clusters.



9. Products selection (All products except License Manager 11.14.1.2).

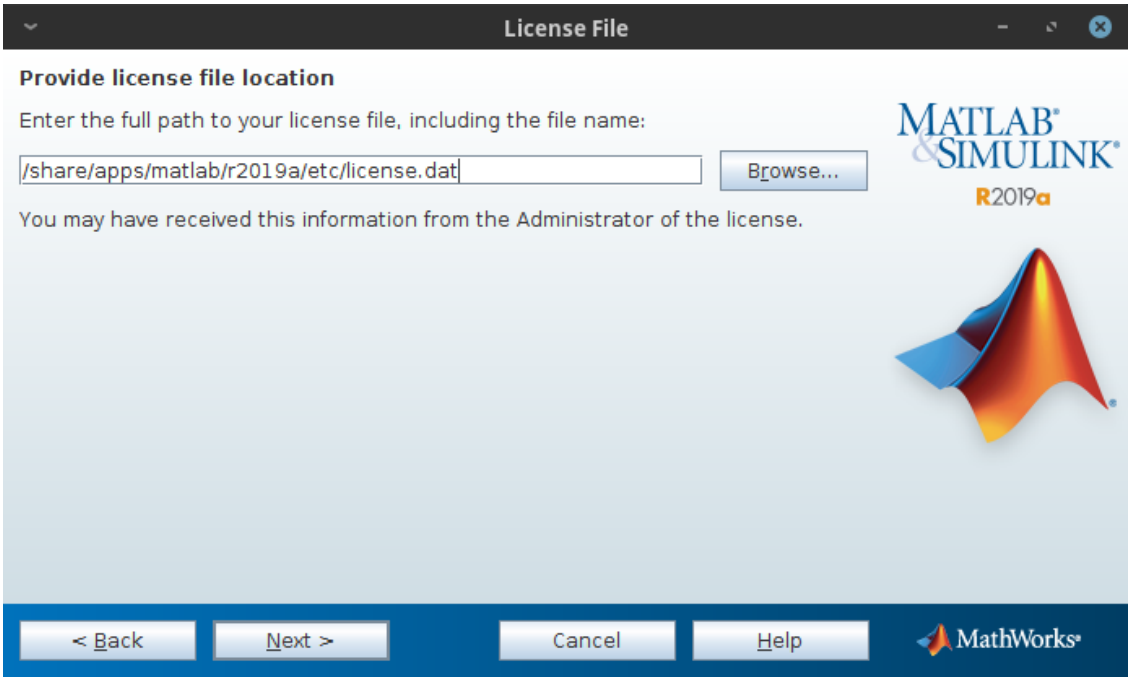
Note: MATLAB recommends install every *Toolbox* available because in this way they can be used by Parallel Server workers.



10. License file (`/share/apps/matlab/r2019a/etc`).

Note: Download and upload the modified `license.dat` file on the *License Manager* server to the `/share/apps/matlab/r2019a/etc` directory on the cluster.

```
mkdir -p /share/apps/matlab/r2019a/etc
cd /share/apps/matlab/r2019a/etc
sftp user@<LICENSE_MANAGER_SERVER>
cd /usr/local/MATLAB/R2019a/etc
mget license.dat
```



11. Wait to finish the installation process.

Intregration with SLURM

To integrate the MATLAB client on the cluster to use SLURM as resource manager you have to follow next steps:

1. Create a folder called `cluster.local` in the following path bash:*/share/common-apps/matlab/r2019a/toolbox/local/*.

```
mkdir /share/common-apps/matlab/r2019a/toolbox/local/cluster.local
```

2. Copy the MATLAB integration scripts into the above mentioned folder (`Scripts`).

```
scp cluster.local.R2019a.tar.gz <user>@<FQDN>:/path/to/file
mv /path/to/file/cluster.local.R2019a.tar.gz /share/common-apps/matlab/
↪r2019a/toolbox/local/cluster.local
cd /share/common-apps/matlab/r2019a/toolbox/local/cluster.local
tar xf cluster.local.R2019a.tar.gz
rm cluster.local.R2019a.tar.gz
```

3. Open the MATLAB client on the cluster to configure the paths for the integration scripts.

Note: Add the module file to be able to load MATLAB for the further steps. Please refer to *Module file* section.

Warning: If MATLAB client is installed in a system directory, we strongly suggest to open it with admin privileges, it is only necessary the first time to configure it.

```
module load matlab/r2019a
matlab -nodisplay
```

```
>> addpath(genpath('/share/common-apps/matlab/r2019a/toolbox/local/cluster.local
↪'))
>> savepath /share/common-apps/matlab/r2019a/toolbox/local/pathdef.m
```

Note: For more info about the last commands please refer to MathWorks documentation.

Configuring Cluster Profiles

1. Open again your MATLAB Client (without admin privileges)

```
module load matlab/r2019a
matlab -nodisplay
```

2. Load the cluster profile and configure it to submit jobs using SLURM via Parallel Server.

```
>> configCluster

   >> % Must set TimeLimit before submitting jobs to
          >> % the cluster.

          >> % i.e. to set the TimeLimit
          >> c = parcluster('cluster R2019a');
          >> c.AdditionalProperties.TimeLimit = '1:00:00';
          >> c.saveProfile
```

Warning: The `configCluster` command should be run only the very first time you configure your MATLAB client. It is NOT necessary to run the command each time.

3. Custom options

   There are some additional properties you can set in order to get the desired resources with Slurm

```
>> c = parcluster()
>> c.AdditionalProperties

ans =

  AdditionalProperties with properties:

              AccountName: ''
    AdditionalSubmitArgs: ''
            EmailAddress: ''
               EmailType: ''
             EnableDebug: 0
```

(continues on next page)

(continued from previous page)

```
                MemUsage: ''
              Partition: ''
           ProcsPerNode: 0
            Reservation: ''
              TimeLimit: ''
                UseSmpd: 0
```

- **TimeLimit (mandatory)** Set a limit on the total run time of the job allocation (more info).
  - e.g. `c.AdditionalProperties.TimeLimit = '3-10:00:00';`
- **AccountName** Change the default user account on Slurm.
  - e.g. `c.AdditionalProperties.AccountName = 'apolo';`
- **ClusterHost** Another way to change the cluster hostname to sumbit jobs.
  - e.g. `c.AdditionalProperties.ClusterHost = 'apolo.eafit.edu.co';`
- **EmailAddress** Get all job notifications by e-mail.
  - e.g. `c.AdditionalProperties.EmailAddress = 'apolo@eafit.edu.co';`
- **EmailType** Get only the desired notifications based on sbatch options.
  - e.g. `c.AdditionalProperties.EmailType = 'END,TIME_LIMIT_50';`
- **MemUsage** Total amount of memory per MATLAB worker (more info).
  - e.g. `c.AdditionalProperties.MemUsage = '5G';`
- **NumGpus** Number of GPUs to use in a job (currently the maximum possible NumGpus value is two, also if you select this option you have to use the *'accel'* partition on *Apolo II*).
  - e.g. `c.AdditionalProperties.NumGpus = '2';`
- **Partition** Select the desire partition to submit jobs (by default *longjobs* partition will be used)
  - e.g. `c.AdditionalProperties.Partition = 'bigmem';`
- **Reservation** Submit a job into a reservation (more info).
  - e.g. `c.AdditionalProperties.Reservation = 'reservationName';`
- **AdditionalSubmitArgs** Any valid sbatch parameter (raw) (more info)
  - e.g. `c.AdditionalProperties.AdditionalSubmitArgs = '-no-requeue -exclusive';`

Troubleshooting

1. When you ran the MATLAB installer with the command `./install`, it prints:

```
Preparing installation files ...
Installing ...
```

Then a small MATLAB window appears and after a while it closes and prints on prompt:

```
Finished
```

To solve this problem, you have to find the root cause modifying `$MATLABINSTALLERPATH/bin/glnxa64/install_unix` script to look the `stderror` and understand what is happening.

- At line *918* change this statement `eval "$java_cmd 2> /dev/null"` to `eval "$java_cmd"`, by this way you can see the related errors launching the MATLAB installer.
  - e.g. missing library *libXtst.so.6*

Module file

Listing 15: `Module file`

```
#%Module1.0#####################################################################
##
## module load matlab/r2019a
##
## /share/apps/modules/matlab/r2019a
## Written by Johan Yepes Rios
##

proc ModulesHelp {} {
     global version modroot
     puts stderr "Sets the environment for using Matlab R2019a\
                  \nin the shared directory /share/common-apps/matlab/r2019a."
}

module-whatis "(Name________) matlab"
module-whatis "(Version_____) r2019a"
module-whatis "(Compilers___) "
module-whatis "(System______) x86_64-redhat-linux"
module-whatis "(Libraries___) "

# for Tcl script use only
set         topdir        /share/common-apps/matlab/r2019a
set         version       r2019a
set         sys           x86_64-redhat-linux

conflict matlab


prepend-path        PATH                        $topdir/bin
```

Usage

This subsection describes three integrations methods to submit jobs to the cluster and in this way use the computational resources through the available licenses in Apolo for MATLAB.

Contents

MDCS using a local MATLAB client

To submit jobs through a local MATLAB client in Apolo II or Cronos using SLURM follow next steps to got the integration:

Integration scripts

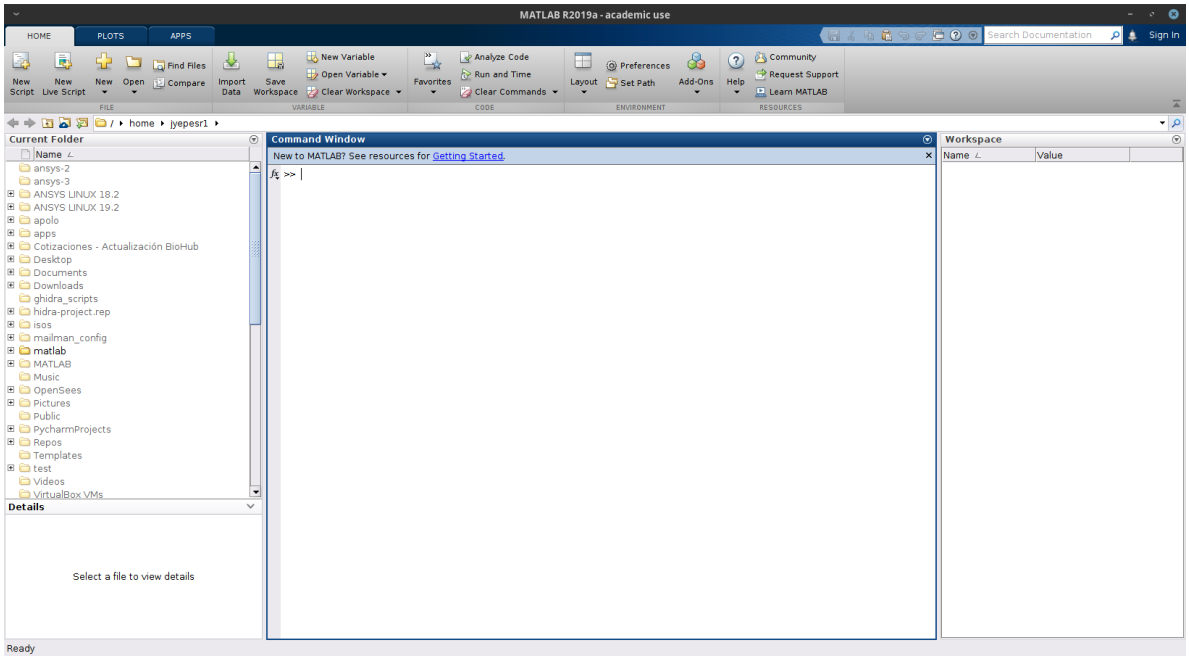
1. Add the MATLAB integration scripts to your MATLAB PATH by placing the integration scripts into a directory in your PC. (`matlab-apolo.zip`).
   - e.g. `$HOME/Documents/matlab-integration`

Linux

```
mkdir $HOME/Documents/matlab-integration
mv path-to-file/matlab-apolo-R2019a.zip $HOME/matlab-integration/
cd $HOME/Documents/matlab-integration
unzip matlab-apolo-R2019a.zip
rm matlab-apolo-R2019a.zip
```

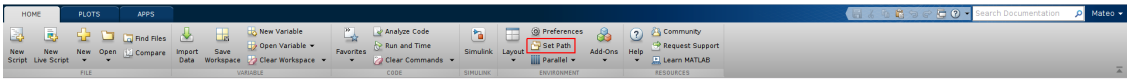
2. Open your MATLAB client to configure it.

Note: MATLAB client is installed in a system directory, we strongly suggest to open it with admin privileges, it is only necessary the first time to configure it.

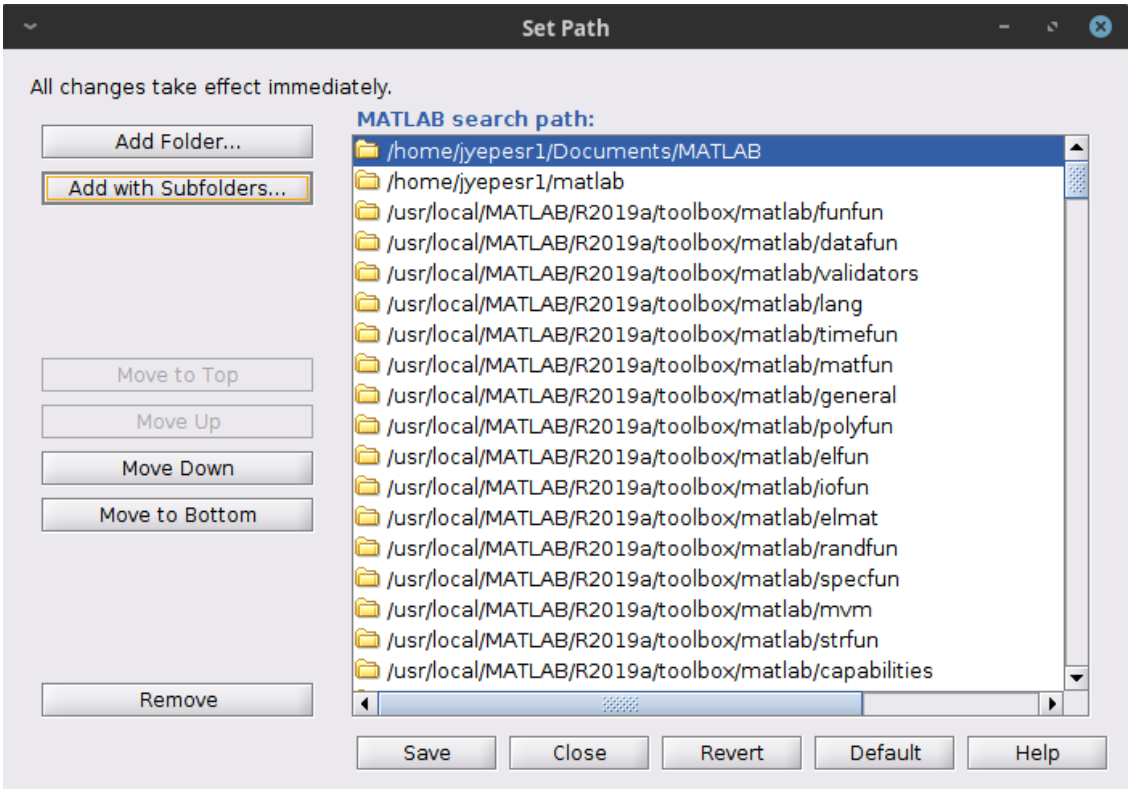


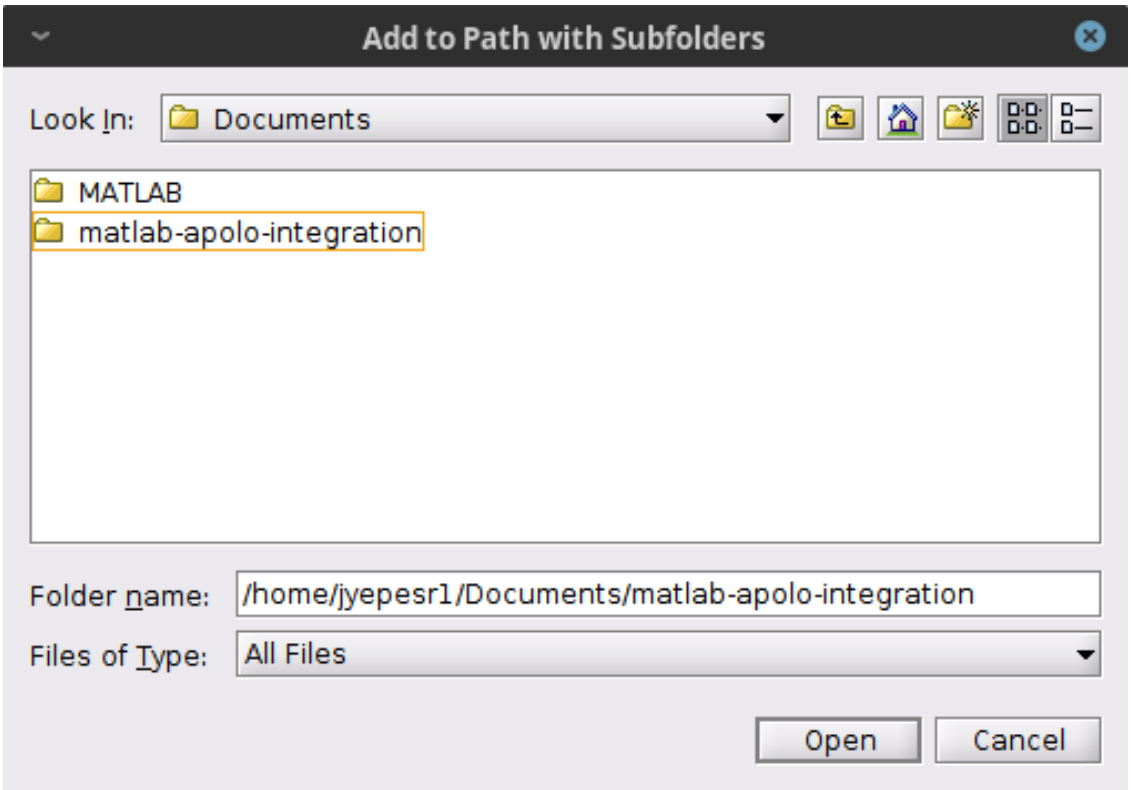
3. Add the integrations scripts to the MATLAB PATH

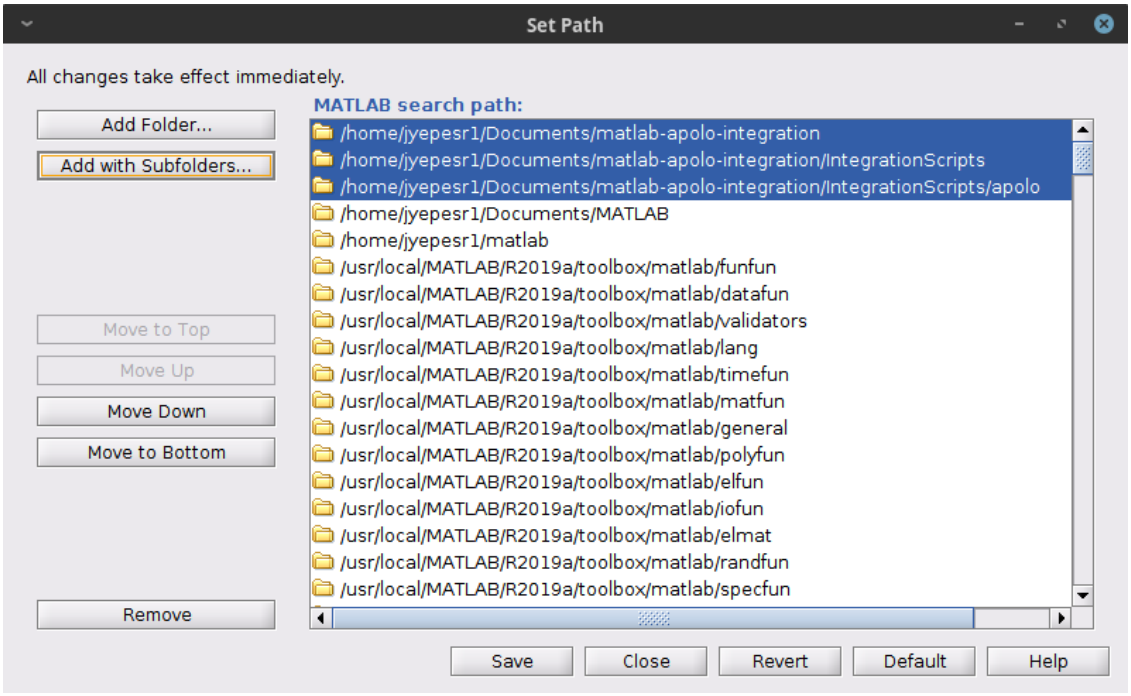
   - Press the **"Set Path"** button



   - Press the **"Add with Subfolders"** button and choose the directories where you unzip the integrations scripts (Apolo II and Cronos) and finally press the **"Save"** button:
     - `$HOME/matlab-integration/EAFIT.nonshared.R2019a`


Set Path
All changes take effect immediately.
MATLAB search path:
Add Folder...
Add with Subfolders...
Move to Top
Move Up
Move Down
Move to Bottom
Remove
/home/jyepesr1/Documents/MATLAB
/home/jyepesr1/matlab
/usr/local/MATLAB/R2019a/toolbox/matlab/funfun
/usr/local/MATLAB/R2019a/toolbox/matlab/datafun
/usr/local/MATLAB/R2019a/toolbox/matlab/validators
/usr/local/MATLAB/R2019a/toolbox/matlab/lang
/usr/local/MATLAB/R2019a/toolbox/matlab/timefun
/usr/local/MATLAB/R2019a/toolbox/matlab/matfun
/usr/local/MATLAB/R2019a/toolbox/matlab/general
/usr/local/MATLAB/R2019a/toolbox/matlab/polyfun
/usr/local/MATLAB/R2019a/toolbox/matlab/elfun
/usr/local/MATLAB/R2019a/toolbox/matlab/iofun
/usr/local/MATLAB/R2019a/toolbox/matlab/elmat
/usr/local/MATLAB/R2019a/toolbox/matlab/randfun
/usr/local/MATLAB/R2019a/toolbox/matlab/specfun
/usr/local/MATLAB/R2019a/toolbox/matlab/mvm
/usr/local/MATLAB/R2019a/toolbox/matlab/strfun
/usr/local/MATLAB/R2019a/toolbox/matlab/capabilities
Save
Close
Revert
Default
Help


Add to Path with Subfolders
Look In:
Documents
MATLAB
matlab-apolo-integration
Folder name:
/home/jyepesr1/Documents/matlab-apolo-integration
Files of Type:
All Files
Open
Cancel



Configuring cluster profiles

1. Configure MATLAB to run parallel jobs on your cluster by calling `configCluster`. It will prompt you for a cluster selection if the user wants to use Apolo or Cronos nodes.

```
>> configCluster
        [1] apolo
        [2] cronos
Select a cluster [1-2]: 1
Username on APOLO (e.g. joe): jyepesr1
```

```
>> % Must set TimeLimit before submitting jobs to apolo.
>> % i.e. to set the TimeLimit to one hour:
>> c = parcluster('apolo R2019a');
>> c.AdditionalProperties.TimeLimit = '1:00:00';
>> c.saveProfile
```

2. Modify the cluster profile through a variable definition `c = parcluster`.

```
>> c = parcluster

c =

 Generic Cluster

    Properties:

                  Profile: apolo R2019a
                 Modified: false
                     Host: odin.apolo.eafit.edu.co
               NumWorkers: 96
               NumThreads: 1

        JobStorageLocation: /home/jyepesr1/MdcsDataLocation/apolo/odin.apolo.local/R2019a/remote
         ClusterMatlabRoot: /share/apps/matlab/r2019a/
           OperatingSystem: unix

   RequiresOnlineLicensing: false
 IntegrationScriptsLocation: /home/jyepesr1/Documents/matlab-apolo-integration/IntegrationScripts/apolo
      AdditionalProperties: List properties

    Associated Jobs:

           Number Pending: 0
            Number Queued: 0
           Number Running: 0
          Number Finished: 0
```

3. Set the custom options as you need.

```
>> c.AdditionalProperties

ans =

  AdditionalProperties with properties:

                AccountName: ''
         AdditionalSubmitArgs: ''
                 ClusterHost: 'apolo.eafit.edu.co'
              EmailAddress: ''
                 EmailType: ''
               EnableDebug: 0
              IdentityFile: ''
                  MemUsage: ''
                   NumGpus: ''
                 Partition: ''
              ProcsPerNode: 0
    RemoteJobStorageLocation: '/home/jyepesr1/MdcsDataLocation/apolo/odin.apolo.local/R2019a/remote'
               Reservation: ''
                 TimeLimit: ''
           UseIdentityFile: 1
                    UseSmpd: 0
          UserNameOnCluster: 'jyepesr1'
```

- `TimeLimit` (**Mandatory**) Set a limit on the total run time of the job allocation (more info).
  - e.g. `c.AdditionalProperties.TimeLimit = '3-10:00:00';`
- `AccountName` Change the default user account on Slurm.
  - e.g. `c.AdditionalProperties.AccountName = 'apolo';`
- `ClusterHost` Another way to change the cluster hostname to submit jobs.
  - e.g. `c.AdditionalProperties.ClusterHost = 'apolo.eafit.edu.co';`
- `EmailAddress` Get all job notifications by e-mail.
  - e.g. `c.AdditionalProperties.EmailAddress = 'apolo@eafit.edu.co';`

- `EmailType` Get only the desired notifications based on sbatch options.
  - e.g. `c.AdditionalProperties.EmailType = 'END,TIME_LIMIT_50';`
- `MemUsage` Total amount of memory **per machine** (more info).
  - e.g. `c.AdditionalProperties.MemUsage = '5G';`
- `NumGpus` Number of GPUs to use in a job.
  - e.g. `c.AdditionalProperties.NumGpus = 2;`

Note: The maximum value for `NumGpus` is four, also if you select this option you should use the **'accel'** partition on *Apolo II*.

- `Partition` Select the desire partition to submit jobs (by default *longjobs* partition will be used)
  - e.g. `c.AdditionalProperties.Partition = 'bigmem';`
- `Reservation` Submit a job into a reservation (more info).
  - e.g. `c.AdditionalProperties.Reservation = 'reservationName';`
- `AdditionalSubmitArgs` Any valid sbatch parameter (raw) (more info)
  - e.g. `c.AdditionalProperties.AdditionalSubmitArgs = '-no-requeue';`

1. Save the current profile to avoid setting the properties each time.

```
>> configCluster
>> c = parcluster;
>> c.AdditionalProperties.TimeLimit = '1:00:00';
>> c.AdditionalProperties.Partition = 'longjobs';
>> c.saveProfile

>> % e.g. to set the NumGpus, TimeLimit and Partition
>> c = parcluster();
>> c.AdditionalProperties.TimeLimit = '1:00:00';
>> c.AdditionalProperties.Partition = 'accel';
>> c.AdditionalProperties.NumGpus = 2;
>> c.saveProfile
```

Submitting jobs

General steps

1. Load *'apolo R2019a'* cluster profile and load the desired properties to submit a job.

```
>> c = parcluster('apolo R2019a');
```

Note: There was no necessary to set the additional properties because they have been already set when the profile was saved the first time with `c.saveProfile`.

---

1. To see the values of the current configuration options, call the specific `AdditionalProperties` method.

```
>> % To view current properties
>> c.AdditionalProperties
```

2. To clear a value, assign the property an empty value (`''`, `[]`, or `false`).

```
>> % Turn off email notifications
>> c.AdditionalProperties.EmailAddress = '';
```

3. If you have to cancel a job (queued or running) type.

```
>> j.cancel
```

4. Delete a job after results are no longer needed.

```
>> j.delete
```

Serial jobs

1. Use the batch command to submit asynchronous jobs to the cluster. The batch command will return a job object which is used to access the output of the submitted job.

   (See the MATLAB documentation for more help on batch.)

Listing 16: `serial_example.m`

```
function t = serial_example(n)

t0 = tic;
A = 500;
a = zeros(n);

for i = 1:n
    a(i) = max(abs(eig(rand(A))));
end

t = toc(t0);

end
```

```
>> % Get a handle to the cluster
>> c = parcluster('apolo R2019a');

>> % Submit job to query where MATLAB is running on the cluster (script)
>> j = c.batch(@serial_example, 1, {1000});

>> % Query job for state
>> j.State

>> % Load results
>> j.fetchOutputs{:}

>> % Delete the job after results are no longer needed
>> j.delete
```

2. To retrieve a list of currently running or completed jobs, call `parcluster` to retrieve the cluster object. The cluster object stores an array of jobs that were run, are running, or are queued to run. This allows us to fetch the results of completed jobs. Retrieve and view the list of jobs as shown below.

```
>> c = parcluster('apolo R2019a');
>> jobs = c.Jobs
```

3. Once we have identified the job we want, we can retrieve the results as we have done previously. `fetchOutputs` is used to retrieve function output arguments; if using batch with a script, use `load` instead.

   Data that has been written to files on the cluster needs be retrieved directly from the file system. To view results of a previously completed job:

```
>> % Get a handle on job with ID 2
>> j2 = c.Jobs(2);
>> j2.fetchOutputs{:}
```

Note: You can view a list of your jobs, as well as their IDs, using the above `c.Jobs` command.

4. Another example using a MATLAB script.

Listing 17: `serial_example_script.m`

```
t0 = tic;
A = 500;
a = zeros(100);
fileID = fopen('/tmp/time.txt','wt');
for i = 1:100
    a(i) = max(abs(eig(rand(A))));
end
t = toc(t0);
fprintf(fileID, '%6.4f\n', t);
fclose(fileID);
```

   - Job submission

```
>> % Get a handle to the cluster
>> c = parcluster('apolo R2019a');

>> % Submit job to query where MATLAB is running on the cluster (script)
>> j = c.batch('serial_example_script');

>> % Query job for state
>> j.State

>> %Load results into the client workspace
>> j.load

>> % Delete the job after results are no longer needed
>> j.delete
```

5. Another example using a MATLAB script that supports GPU.

Listing 18: `gpu_script.m`

```
maxIterations = 500;
gridSize = 1000;
xlim = [-0.748766713922161, -0.748766707771757];
ylim = [ 0.123640844894862,  0.123640851045266];
```

(continues on next page)

(continued from previous page)

```
% Setup
t = tic();
x = gpuArray.linspace( xlim(1), xlim(2), gridSize );
y = gpuArray.linspace( ylim(1), ylim(2), gridSize );
[xGrid,yGrid] = meshgrid( x, y );
z0 = complex( xGrid, yGrid );
count = ones( size(z0), 'gpuArray' );

% Calculate
z = z0;
for n = 0:maxIterations
          z = z.*z + z0;
inside = abs( z )<=2;
count = count + inside;
end
count = log( count );

% Show
count = gather( count ); % Fetch the data back from the GPU
naiveGPUTime = toc( t );
fig = gcf;
fig = figure('visible', 'off');
fig.Position = [200 200 600 600];
imagesc( x, y, count );
colormap( [jet();flipud( jet() );0 0 0] );
axis off;
title( sprintf( '%1.2fsecs (GPU)', naiveGPUTime ) );
saveas(gcf,'/tmp/GPU.png');
```

- Job submission

```
>> % Get a handle to the cluster
>> c = parcluster('apolo R2019a');

>> % Submit job to query where MATLAB is running on the cluster (script)
>> j = c.batch('gpu_script');

>> % Query job for state
>> j.State
```

6. Another example using Simulink via MATLAB.

Listing 19: `parsim_test_script.m`

```
% Example running a Simulink model.

% The Simulink model is called |parsim_test.slx| and it *must be* in
% the cluster.

% Number of simulations
numSims = 10;
W = zeros(1,numSims);

% Changing to the |parsim_test.slx| path
cd ~/tests/simulink

% Create an array of |SimulationInput| objects and specify the argument value
```

(continues on next page)

(continued from previous page)

```
% for each simulation. The variable |x| is the input variable in the Simulink
% model.
for x = 1:numSims
  simIn(x) = Simulink.SimulationInput('parsim_test');
  simIn(x) = setBlockParameter(simIn(x), 'parsim_test/Transfer Fcn', 'Denominator
↪', num2str(x));
end

% Running the simulations.
simOut = parsim(simIn);

% The variable |y| is the output variable in the Simulink model.
for x = 1:numSims
  W(1,x) = max(simOut(x).get('y'));
end

save('~/output_file.mat','W');
```

`parsim_test.slx (Simulink model)`

- Job submission

```
>> % Get a handle to the cluster
>> c = parcluster('apolo remote R2018a');

>> % Submit job to query where MATLAB is running on the cluster (script)
>> j = c.batch('parsim_test_script');

>> % Query job for state
>> j.State

>> % Load data to client workspace
>> j.load
```

Parallel or distributed jobs

Users can also submit parallel or distributed workflows with batch command. Let's use the following example for a parallel job.

Listing 20: `parallel_example.m`

```
function t = parallel_example(n)

t0 = tic;
A = 500;
a = zeros(n);
parfor i = 1:n
  a(i) = max(abs(eig(rand(A))));
end
t = toc(t0);
end
```

1. We will use the batch command again, but since we are running a parallel job, we will also specify a MATLAB pool.

```
>> % Get a handle to the cluster
>> c = parcluster('apolo R2019a');

>> % Submit a batch pool job using 4 workers
>> j = c.batch(@parallel_example, 1, {1000}, 'Pool', 4);

>> % View current job status
>> j.State

>> % Fetch the results after a finished state is retrieved
>> j.fetchOutputs{:}
ans =
   41.7692
```

- The job ran in 41.7692 seconds using 4 workers.

Note: Note that these jobs will always request N+1 CPU cores, since one worker is required to manage the batch job and pool of workers. For example, a job that needs eight workers will consume nine CPU cores.

Note: For some applications, there will be a diminishing return when allocating too many workers, as the overhead may exceed computation time (communication).

2. We will run the same simulation, but increase the pool size. This time, to retrieve the results later, we will keep track of the job ID.

```
>> % Get a handle to the cluster
>> c = parcluster('apolo R2019a');

>> % Submit a batch pool job using 8 workers
>> j = c.batch(@parallel_example, 1, {1000}, 'Pool', 8);

>> % Get the job ID
>> id = j.ID
Id =
4
>> % Clear workspace, as though we quit MATLAB
>> clear
```

3. Once we have a handle to the cluster, we will call the `findJob` method to search for the job with the specified job ID.

```
>> % Get a handle to the cluster
>> c = parcluster('apolo R2019a');

>> % Find the old job
>> j = c.findJob('ID', 4);

>> % Retrieve the state of the job
>> j.State
ans
finished
>> % Fetch the results
```

(continues on next page)

(continued from previous page)

```
>> j.fetchOutputs{:}
ans =
22.2082
```

- The job now runs 22.2082 seconds using 8 workers. Run code with different number of workers to determine the ideal number to use.

4. Another example using a parallel script.

Listing 21: `parallel_example_script.m`

```
n = 1000;
t0 = tic;
A = 500;
a = zeros(n);
fileID = fopen('/tmp/time.txt','wt');

parfor i = 1:n
  a(i) = max(abs(eig(rand(A))));
end

t = toc(t0);
fprintf(fileID, '%6.4f\n', t);
fclose(fileID);
```

```
>> % Get a handle to the cluster
>> c = parcluster('apolo R2019a');

>> % Submit job to query where MATLAB is running on the cluster (script)
>> j = c.batch('parallel_example_script', 'Pool', 8);

>> % Query job for state
>> j.State

>> %Load results
>> j.load

>> % Delete the job after results are no longer needed
>> j.delete
```

Debugging

1. If a serial job produces an error, we can call the `getDebugLog` method to view the error log file using *j.Tasks(1)*. Additionally when submitting independent jobs, with multiple tasks, you will have to specify the task number.

```
>> % If necessary, retrieve output/error log file
>> j.Parent.getDebugLog(j.Tasks(1))
```

2. For pool jobs, do not diference into the job object.

```
>> % If necessary, retrieve output/error log file
>> j.Parent.getDebugLog(j)
```

(continues on next page)

(continued from previous page)

```
>> % or
>> c.getDebugLog(j)
```

3. To get information about the job in SLURM, we can consult the scheduler ID by calling `schedID`.

```
>> schedID(j)
   ans =
      25539
```

MDCS using cluster's MATLAB client

Submitting jobs from within MATLAB client on the cluster

General steps

1. Connect to Apolo II or Cronos via SSH.

```
# Without graphical user interface
ssh username@[cronos,apolo].eafit.edu.co
# or with graphical user interface
ssh -X username@[cronos,apolo].eafit.edu.co
```

2. Load MATLAB modufile.

```
module load matlab/r2019a
```

3. Run MATLAB client

```
matlab
```

4. First time, you have to define the cluster profile running the following command.

```
>> configCluster

  >> % Must set TimeLimit before submitting jobs to
          >> % the cluster.

          >> % i.e. to set the TimeLimit
          >> c = parcluster('cluster R2019a');
          >> c.AdditionalProperties.TimeLimit = '1:00:00';
          >> c.saveProfile
```

Warning: The `configCluster` command should be run only the very first time you configure your MATLAB client. It is NOT necessary to run the command each time.

5. Load the cluster profile to submit a job (MATLAB GUI or command line).

```
>> % Must set TimeLimit before submitting jobs to Apolo II or
>> % Cronos cluster

>> c = parcluster()
```

(continues on next page)

(continued from previous page)

```
 c =

  Generic Cluster

     Properties:

                     Profile: cluster R2019a
                    Modified: false
                        Host: apolo.eafit.edu.co
                  NumWorkers: 96
                  NumThreads: 1

          JobStorageLocation: /home/<user>/MdcsDataLocation/cluster/R2019a/local
           ClusterMatlabRoot: /share/common-apps/matlab/r2019a
             OperatingSystem: unix

     RequiresOnlineLicensing: false
   IntegrationScriptsLocation: /share/common-apps/matlab/r2019a/toolbox/local/
↪cluster.local/IntegrationScripts/cluster
        AdditionalProperties: List properties

     Associated Jobs:

              Number Pending: 0
               Number Queued: 0
              Number Running: 0
             Number Finished: 0
```

6. To see the values of the current configuration options, call the specific `AdditionalProperties` method.

```
>> % To view current properties
>> c.AdditionalProperties

  ans =

    AdditionalProperties with properties:

               AccountName: ''
     AdditionalSubmitArgs: ''
             EmailAddress: ''
                EmailType: ''
              EnableDebug: 0
                 MemUsage: ''
                Partition: ''
             ProcsPerNode: 0
              Reservation: ''
                TimeLimit: ''
                  UseSmpd: 0
```

7. To clear a value, assign the property an empty value (`''`, `[]`, or `false`).

```
>> % Turn off email notifications
>> c.AdditionalProperties.EmailAddress = '';
```

Submitting jobs

Note: Users can submit serial, parallel or distributed jobs with batch command as the previous examples.

Submitting jobs directly through SLURM

MDCS jobs could be submitted directly from the Unix command line through SLURM.

For this, in addition to the MATLAB source, one needs to prepare a MATLAB submission script with the job specifications.

1. An example is shown below:

Listing 22: `matlab_batch.m`

```
%==========================================================
% MATLAB job submission script: matlab_batch.m
%==========================================================

workers = str2num(getenv('SLURM_NTASKS'));
c = parcluster('apolo');
c.AdditionalProperties.TimeLimit = '1:00:00';
c.AdditionalProperties.Partition = 'longjobs';
j = c.batch(@parallel_example_slurm, 1, {1000}, 'pool', workers);
exit;
```

Listing 23: `parallel_example_slurm.m`

```
function t = parallel_example_slurm(n)

t0 = tic;
A = 500;
a = zeros(n);

parfor i = 1:n
  a(i) = max(abs(eig(rand(A))));
end

t = toc(t0);
save prueba.txt t -ascii

end
```

2. It is submitted to the queue with help of the following SLURM batch-job submission script:

Listing 24: `matlab.slurm`

```
#!/bin/bash

#SBATCH -J test_matlab
#SBATCH -o test_matlab-%j.out
#SBATCH -e test_matlab-%j.err
#SBATCH -p longjobs
#SBATCH -n 8
```

(continues on next page)

(continued from previous page)

```
#SBATCH -t 20:00

module load matlab/r2018a

matlab -nosplash -nodesktop -r "matlab_batch"
```

3. Job is submitted as usual with:

```
sbatch matlab.slurm
```

Note: This scheme dispatches 2 jobs - one serial that spawns the actual MDCS parallel jobs, and another, the actual parallel job.

4. Once submitted, the job can be monitored and managed directly through SLURM
   - `squeue` command output

```
 JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
326582  longjobs Gamma_1- jdavidca PD       0:00      1 (Priority)
326602  longjobs hexa298s ltenelan PD       0:00      1 (Priority)
326603  longjobs tetra599 ltenelan PD       0:00      1 (Priority)
326604  longjobs tricli91 ltenelan PD       0:00      1 (Priority)
326618  longjobs    RAxML apolo-jo PD       0:00      1 (Priority)
326619  longjobs  lattice  ohenaoz PD       0:00      1 (Priority)
326620  longjobs bayescan apolo-jo PD       0:00      1 (Priority)
326621  longjobs  lattice  ohenaoz PD       0:00      1 (Priority)
326580  longjobs alpha_1- jdavidca PD       0:00      1 (Resources)
326581  longjobs Beta-fre jdavidca PD       0:00      1 (Priority)
326583     accel edificio apolo-dr  R 1-01:14:26      1 compute-0-5
326578  longjobs    MED3P  maberce  R 1-19:49:23      2 compute-0-[1,3]
326534  longjobs BootInfl larteag7  R 4-19:12:58      2 compute-0-[0,2]
```

5. After the job completes, one can fetch results and delete job object from within MATLAB client on the cluster. If program writes directly to disk fetching is not necessary.

```
>> c = parcluster('apolo');
>> jobs = c.Jobs
>> j = c.Jobs(7);
>> j.fetchOutputs{:};
>> j.delete;
```

MATLAB directly on the cluster

Next steps describes how to use MATLAB and its toolboxes without MDCS (workers) toolbox, but this way has next pros and cons.

- **Pros**
  - No workers limitations
- **Cons**
  - No distributed jobs (Only parallel or serial jobs)

Unattended job

To run unattended jobs on the cluster follow next steps:

1. Connect to Apolo II or Cronos via SSH.

```
ssh username@cronos.eafit.edu.co
```

2. Enter to the matlab directory project.

```
cd ~/test/matlab/slurm
```

3. Create a SLURM batch-job submission script

Listing 25: `slurm.sh`

```
#!/bin/bash

#SBATCH -J test_matlab
#SBATCH -o test_matlab-%j.out
#SBATCH -e test_matlab-%j.err
#SBATCH -p bigmem
#SBATCH -n 8
#SBATCH -t 20:00

module load matlab/r2018a

matlab -nosplash -nodesktop < parallel_example_unattended.m
```

Listing 26: `parallel_example_unattended.m`

```
p = parpool(str2num(getenv('SLURM_NTASKS')));

t0 = tic;
A = 500;
a = zeros(1000);
parfor i = 1:1000
  a(i) = max(abs(eig(rand(A))));
end
t = toc(t0)
exit
```

4. Submit the job.

```
sbatch slurm.sh
```

1. Check the `stdout` file (`test_matlab_xxxx.out`).

```
                      < M A T L A B (R) >
            Copyright 1984-2019 The MathWorks, Inc.
       R2019a Update 3 (9.6.0.1135713) 64-bit (glnxa64)
                         June 5, 2019

To get started, type doc.
For product information, visit www.mathworks.com.

>> Starting parallel pool (parpool) using the 'local' profile ...
```

(continues on next page)

(continued from previous page)

```
connected to 8 workers.

t =

  22.5327
```

Interactive job (No GUI)

If it is necessary the user can run interactive jobs following next steps:

1. Connect to Apolo II or Cronos via SSH.

```
ssh username@apolo.eafit.edu.co
```

2. Submit a interactive request to the resource manager

```
srun -N 1 --ntasks-per-node=2 -t 20:00 -p debug --pty bash
# If resources are available you get immediately a shell in a slave node
# e.g. compute-0-6
module load matlab/r2019a
matlab
```

```
                            < M A T L A B (R) >
                  Copyright 1984-2019 The MathWorks, Inc.
              R2019a Update 3 (9.6.0.1135713) 64-bit (glnxa64)
                                June 5, 2019

To get started, type doc.
For product information, visit www.mathworks.com.

 >> p = parpool(str2num(getenv('SLURM_NTASKS')));
 Starting parallel pool (parpool) using the 'local' profile ...
 >> p.NumWorkers

 ans =

      2
```

Note: At this point you have an interactive MATLAB session through the resource manager (SLURM), giving you the possibility to test and check different MATLAB features.

3. To finish this job, you have to close the MATLAB session and then the bash session granted in the slave node.

References

- Parallel Computing Toolbox
- MATLAB Distributed Computing Server
- "Portions of our documentation contain content originally created by Harvard FAS Research Computing and adapted by us under the Creative Commons Attribution-NonCommercial 4.0 International License. More information: https://rc.fas.harvard.edu/about/attribution/"

Authors

- Mateo Gómez-Zuluaga <mgomezz@eafit.edu.co>

Resources

Here you can find some resources such as documents, videos, tutorials that will support the installation and use of Matlab.

Videos

Use of Matlab with high-performance computing in Apolo. Juan David Pineda Cardenas and Mateo Gómez Zuluaga. APOLO Scientific Computing Center EAFIT University. APOLO Lecture Series. Wednesday, August 1, 2018.

References

3.2.2 Python

Python[1] is an interpreted, high-level, general-purpose programming language. Created by Guido van Rossum and first released in 1991, Python has a design philosophy that emphasizes code readability, notably using significant whitespace. It provides constructs that enable clear programming on both small and large scales.

Conda

Conda[1] is an open source package management system and environment management system that runs on Windows, macOS and Linux. Conda quickly installs, runs and updates packages and their dependencies. Conda easily creates, saves, loads and switches between environments on your local computer. It was created for Python programs, but it can package and distribute software for any language.

Important: Remember to load the python module with miniconda. `module load python/x.x.x_miniconda-x.x.x`

Conda-4.5.x

Conda environments

What is a virtual environment

A virtual environment[1] is a named, isolated, working copy of Python that maintains its own files, directories, and paths so that you can work with specific versions of libraries or Python itself without affecting other Python projects.

[1] Wikipedia contributors. (2019, March 4). Python. In Wikipedia, The Free Encyclopedia. Retrieved 13:29, March 4, 2019, from https://en.wikipedia.org/wiki/Python_(programming_language)

[1] Conda. (2017). Conda. Retrieved March 4, 2019, from https://conda.io/en/latest/

[1] Jekyll. (2014, November 20). Create virtual environments for python with conda. Retrieved from https://uoa-eresearch.github.io/eresearch-cookbook/recipe/2014/11/20/conda/

Creating a virtual environment

To create a virtual environment run:

```
conda create -n <virtual-environment>
```

Then Conda will ask for permission to proceed, press `y`. Remember, the environment usually is created in the user's home directory `.conda/envs/environment-name>`.

Activate a virtual environment

To activate or switch into a virtual environment run:

```
source activate <environment-name>
```

Remember that it's not necessary to deactivate the actual environment to switch into another environment. Also, your shell should change to indicate the name of the current conda environment `(<environment-name>)user@hostname`.

Note: To list all your environments use: `conda env list`

Deactivate a virtual environment

To end a session in the current environment run:

```
source deactivate
```

It is not necessary to specify the environment name, it takes the current environment and deactivates it.

Delete a virtual environment

If the user wants to delete a non-using environment run the following command:

```
coda env remove -n <environment-name>
```

Export an environment

To export an environment file use the following command after activate the environment to export, the file will be create in the actual directory:

```
conda env export > <environment-name>.yml
```

Then to use an environment that was exported run:

```
conda env create -f <environment-name>.yml
```

where `-f` means the environment export file.

Other useful commands

Create a requirements file

The requirements file is a way to get pip to install specific packages to make up an environment[2], also this file list all the packages that are used to documentation. In Conda, you can create this file using:

```
conda list -e > requirements.txt
```

References

Packages

Install a package

To install a package in the current environment use:

```
conda install <package-name>
```

Also, you can specify an environment for the installation:

```
conda install -n <environment-name> <package-name>
```

Additionally, you can install a specific package version:

```
conda install <package-name>=<x.x.x>

conda install scipy=0.15.0
```

Note: You can install multiple packages at once.

```
conda install <package-name> <package-name>

conda install <package-name>=<x.x.x> <package-name>=<x.x.x>
```

Uninstall a package

To remove a list of packages from your current environment use, you can use remove or uninstall (that is an alias for remove):

```
conda remove <package-name>
```

Also, you can specify an environment:

```
conda remove -n <environment-name> <package-name>
```

Note: You can uninstall multiple packages at once.

[2] (2019, June 24). Requirements file. Retrieved from https://pip.readthedocs.io/en/1.1/requirements.html

```
conda remove <package-name> <package-name>
```

Update packages

You can check if a new package update is available, you can choose to install it or not:

```
conda update <package>
```

Others useful commands

- **Clean: Use this command to remove unused packages and caches** `conda clean`
- **List: List all the packages in the current environment** `conda list`
- **Search: Search for packages and display associated information.** `conda search <package-name>`

Note: For more information about Conda or specific Conda commands run `conda -h` or `conda <command> -h`

References

References

3.2.3 R

[1] R is a language and environment for statistical computing and graphics. It is a GNU project which is similar to the S language and environment which was developed at Bell Laboratories (formerly AT&T, now Lucent Technologies) by John Chambers and colleagues. R can be considered as a different implementation of S. There are some important differences, but much code written for S runs unaltered under R.

R provides a wide variety of statistical (linear and nonlinear modelling, classical statistical tests, time-series analysis, classification, clustering, ...) and graphical techniques, and is highly extensible. The S language is often the vehicle of choice for research in statistical methodology, and R provides an Open Source route to participation in that activity.

R - 3.6.0

Basic information

- **Deploy date:** 28 June 2019
- **Official Website:** https://www.r-project.org/
- **License:** https://www.r-project.org/Licenses/
- **Installed on:** *Apolo II*, *Cronos*

[1] R Core Team (2014). R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. URL http://www.R-project.org/.

Dependencies

- zlib >= 1.2.5
- bzip2 >= 1.0.6 compiled with `-fPIC` option
- xz (liblzma) REF https://tukaani.org/xz/
- PCRE >= 8.20 > 10.0 with UTF-8 support
- curl >= 7.22 with https support
- libicu-4.2.1-14.el6.x86_64 and libicu-devel-4.2.1-14.el6.x86_64
- BLAS and LAPACK (Optional) with OpenBlas or MKL
- JDK >= 1.7.0

Installation

After the dependencies have been resolved and configured, the **R** installation can start.

Note: The Intel Compiler will be used to build R.

1. Download the selected version from an official mirror (https://cran.r-project.org/mirrors.html).

```
$ wget https://cran.r-project.org/src/base/R-3/R-3.6.0.tar.gz
$ tar -zxvf R-3.6.0.tar.gz
```

2. Load the corresponding modules to set the building environment.

Note: For BLAS and LAPACK is recommended to use Intel MKL (https://software.intel.com/en-us/articles/build-r-301-with-intel-c-compiler-and-intel-mkl-on-linux)

```
$ module purge # Clean predefined environment
$ module load zlib bzip2 xz pcre curl java mkl intel
```

3. Configure and build the sources.

```
$ ./configure --prefix=/share/apps/r/3.6.0/intel-18.0.2/ --build=x86_64-redhat-linux -
↪-enable-R-shlib \
    --with-blas="-lmkl_rt -liomp5 -lpthread" --with-lapack --enable-BLAS-shlib --
↪without-x --enable-memory-profiling \
     LDFLAGS="$LDFLAGS -qopenmp -L/share/apps/bzip2/1.0.6/gcc-5.5.0/lib/ -L/share/
↪apps/pcre/8.41/gcc-5.5.0/lib/ \
     -L/share/apps/xz/5.2.3/gcc-5.5.0/lib/" CFLAGS="$CFLAGS -fPIC -qopenmp -O3 -ipo -
↪xHost \
     -I/share/apps/bzip2/1.0.6/gcc-5.5.0/include/ -I/share/apps/pcre/8.41/gcc-5.5.0/
↪include/ \
     -I/share/apps/xz/5.2.3/gcc-5.5.0/include" CXXFLAGS="$CXXFLAGS -fPIC -qopenmp -O3␣
↪-ipo -xHost \
     -I/share/apps/bzip2/1.0.6/gcc-5.5.0/include/ -I/share/apps/xz/5.2.3/gcc-5.5.0/
↪include" \
     FFLAGS="$FFLAGS -fPIC -qopenmp -O3 -ipo -xHost" FCFLAGS="$FCFLAGS -fPIC -qopenmp␣
↪-O3 -ipo -xHost"
```

(continues on next page)

(continued from previous page)

```
$ make -j10 && make install
```

Module

The following is the module used for this version.

```
##%Module1.0#####################################################################
##
## modules r/3.6.0_intel-18.0.2_mkl
##
## /share/apps/modules/r/3.6.0_intel-18.0.2_mkl  Written by Johan Yepes
##

proc ModulesHelp { } {
    puts stderr "\tR/3.6.0_intel-18.0.2_mkl - sets the Environment for R in \
    \n\tthe share directory /share/apps/r/3.6.0/intel_mkl/2017_update-1\n"
}

module-whatis "\n\n\tSets the environment for R language \
              \n\tbuilded with Intel Parallel Studio XE Cluster Edition 2018\
              \n\t(Update-1) and Intel MKL 2018 (Update-2)  version\n"

# for Tcl script use only
set       topdir     /share/apps/r/3.6.0/intel-18.0.2
set       version    3.6.0
set       sys        x86_64-redhat-linux

conflict r

module load java/jdk-8_u152 mkl/18.0.2 intel/18.0.2

prepend-path    PATH                    $topdir/bin

prepend-path    LD_LIBRARY_PATH         $topdir/lib64/R/lib
prepend-path    LD_RUN_PATH             $topdir/lib64/R/lib
prepend-path    LIBRARY_PATH            $topdir/lib64/R/lib
prepend-path    LD_LIBRARY_PATH         $topdir/lib64/R/modules
prepend-path    LD_RUN_PATH             $topdir/lib64/R/modules
prepend-path    LIBRARY_PATH            $topdir/lib64/R/modules

prepend-path    C_INCLUDE_PATH          $topdir/lib64/R/include
prepend-path    CXX_INCLUDE_PATH        $topdir/lib64/R/include
prepend-path    CPLUS_INCLUDE_PATH      $topdir/lib64/R/include
prepend-path    C_INCLUDE_PATH          $topdir/lib64/R/include/R_ext
prepend-path    CXX_INCLUDE_PATH        $topdir/lib64/R/include/R_ext
prepend-path    CPLUS_INCLUDE_PATH      $topdir/lib64/R/include/R_ext

prepend-path    PKG_CONFIG_PATH         $topdir/lib64/pkgconfig

prepend-path    MAN_PATH                $topdir/share/man
```

Additional Libraries

Keras GPU

Follow the steps below to set up Keras with GPU support in R.

Note: For the installation, R will be downloaded from Conda because of the Keras library when it is installed does NOT recognize the TensorFlow installation if the R built by us is used.

1. Load the Python 3 module.

```
$ module load python/3.6.8_intel-2019_update-4
```

2. Create the environment in Conda.

```
$ conda create -n r-tensorflow keras-gpu cudatoolkit=9.2
```

Warning: It is mandatory to specify the CUDA Toolkit version because the current driver version supports up to CUDA 9.2. If the version is omitted conda will install the latest (CUDA 10.0).

3. Activate the environment and install R

```
$ source activate r-tensorflow
$ conda install r-essentials r-base
```

4. Log in to the GPU node through Slurm.

```
$ srun -n 1 -t 60 -p accel --gres gpu:4 --pty bash
```

Warning: You only can log in to the GPU node if you have permissions to launch jobs in the "accel" partition. Ask the administrator if you are not sure.

5. Install the Keras library.

```
$ source activate r-tensorflow
$ R
```

```
> install.packages("devtools")
> library(devtools)
> install_github("rstudio/keras")
```

6. Test if GPUs are recognized.

```
> library(keras)
> k=backend()
> sess = k$get_session()
> sess$list_devices()
```

7. Check if the processes are running in the GPUs.

Note: Login to the GPU node using another terminal session. `ssh compute-0-5`.

```
$ watch nvidia-smi
```

```
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 396.26                 Driver Version: 396.26                    |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|===============================+======================+======================|
|   0  Tesla K80           On   | 00000000:05:00.0 Off |                    0 |
| N/A   30C    P0    73W / 149W |     72MiB / 11441MiB |      0%      Default |
+-------------------------------+----------------------+----------------------+
|   1  Tesla K80           On   | 00000000:06:00.0 Off |                    0 |
| N/A   30C    P0    73W / 149W |     72MiB / 11441MiB |      0%      Default |
+-------------------------------+----------------------+----------------------+
|   2  Tesla K80           On   | 00000000:84:00.0 Off |                    0 |
| N/A   34C    P0    57W / 149W |      0MiB / 11441MiB |      0%      Default |
+-------------------------------+----------------------+----------------------+
|   3  Tesla K80           On   | 00000000:85:00.0 Off |                    0 |
| N/A   26C    P8    31W / 149W |      0MiB / 11441MiB |      0%      Default |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                       GPU Memory |
|  GPU       PID   Type   Process name                             Usage      |
|=============================================================================|
|    0     31882      C   ...onda/envs/r-tensorflow/lib/R/bin/exec/R    61MiB |
|    1     31882      C   ...onda/envs/r-tensorflow/lib/R/bin/exec/R    61MiB |
+-----------------------------------------------------------------------------+
```

Authors

- Johan Sebastián Yepes Ríos <jyepesr1@eafit.edu.co>

R - 3.6.1

Basic information

- **Deploy date:** 9 December 2019
- **Official Website:** https://www.r-project.org/
- **License:** https://www.r-project.org/Licenses/
- **Installed on:** *Apolo II*

Dependencies

- zlib >= 1.2.5
- bzip2 >= 1.0.6 compiled with `-fPIC` option
- xz (liblzma) REF https://tukaani.org/xz/
- PCRE >= 8.20 > 10.0 with UTF-8 support

- curl >= 7.22 with https support
- libicu-4.2.1-14.el6.x86_64 and libicu-devel-4.2.1-14.el6.x86_64
- BLAS and LAPACK (Optional) with OpenBlas or MKL
- JDK >= 1.7.0

Installation

After the dependencies have been resolved and configured, the **R** installation can start.

Note: The GCC Compiler will be used to build R.

1. Download the selected version from an official mirror (https://cran.r-project.org/mirrors.html).

```
$ wget https://cran.r-project.org/src/base/R-3/R-3.6.1.tar.gz
$ tar -zxvf R-3.6.1.tar.gz
```

2. Load the corresponding modules to set the building environment.

Note: For BLAS and LAPACK is recommended to use Intel MKL (https://software.intel.com/en-us/articles/build-r-301-with-intel-c-compiler-and-intel-mkl-on-linux)

```
$ module purge # Clean predefined environment
$ module load zlib bzip2 xz pcre curl java mkl gcc/5.4.0
```

3. Configure and build the sources.

```
$ CXX98="gcc -std=c++98" CXX11="gcc -std=c++11" CXX14="gcc -std=c++14" CXX17="gcc -
↪std=c++17" \
    ../configure --enable-lto="bootstrap-lto" --prefix=/share/apps/r/3.6.1/gcc/5.4.0 \
    --enable-java --build=x86_64-redhat-linux --enable-R-shlib \
    --with-blas="-lmkl_blas95_lp64 -liomp5 -lpthread" \
    --with-lapack="-lmkl_scalapack_lp64" --enable-BLAS-shlib \
    --without-x --enable-memory-profiling \
    LDFLAGS="$LDFLAGS \
            -L/share/apps/bzip2/1.0.6/lib \
            -L/share/apps/pcre/8.39/lib \
            -L/share/apps/xz/5.2.2/lib" \
    CFLAGS="$CFLAGS -fPIC -fopenmp -O3 -mavx2 \
            -I/share/apps/bzip2/1.0.6/include \
            -I/share/apps/pcre/8.39/include \
            -I/share/apps/xz/5.2.2/include" \
    CXXFLAGS="$CXXFLAGS -fPIC -fopenmp -O3 -maxv2 \
            -I/share/apps/bzip2/1.0.6/include \
            -I/share/apps/xz/5.2.2/include" \
    FFLAGS="$FFLAGS -fPIC -fopenmp -O3 -mavx2" \
    FCFLAGS="$FCFLAGS -fPIC -fopenmp -O3 -mavx2"

$ make -j8
```

4. Make the tests.

```
$ make check
```

If problems with the test `reg-packages.Rout` arise, ignore it, it seems to be a problem with the NFS, check here.

5. Install.

```
$ sudo mkdir -p /share/apps/r/3.6.1/gcc/5.4.0
$ sudo make install
```

Module

The following is the module used for this version.

```
#%Module1.0#####################################################################
##
## modules r/3.6.1_gcc-5.4.0_mkl
##
## /share/apps/modules/r/3.6.1_gcc-5.4.0_mkl  Written by Johan Yepes
##

proc ModulesHelp { } {
    puts stderr "\tR/3.6.1_gcc-5.4.0_mkl - sets the Environment for R in \
    \n\tthe share directory /share/apps/r/3.6.1/gcc/5.4.0\n"
}

module-whatis "\n\n\tSets the environment for R language \
            \n\tbuilt with GCC 5.4.0 and Intel MKL 2017 (Update-1)version \
            \n\t(Update-1)\n"

# for Tcl script use only
set       topdir     /share/apps/r/3.6.1/gcc/5.4.0
set       version    3.6.1
set       sys        x86_64-redhat-linux

conflict r

module load java/jdk-1.8.0_112 intel/2017_update-1 mkl/2017_update-1 gcc/5.4.0

prepend-path    PATH                    $topdir/bin

prepend-path    LD_LIBRARY_PATH         $topdir/lib64/R/lib
prepend-path    LD_RUN_PATH             $topdir/lib64/R/lib
prepend-path    LIBRARY_PATH            $topdir/lib64/R/lib
prepend-path    LD_LIBRARY_PATH         $topdir/lib64/R/modules
prepend-path    LD_RUN_PATH             $topdir/lib64/R/modules
prepend-path    LIBRARY_PATH            $topdir/lib64/R/modules

prepend-path    C_INCLUDE_PATH          $topdir/lib64/R/include
prepend-path    CXX_INCLUDE_PATH        $topdir/lib64/R/include
prepend-path    CPLUS_INCLUDE_PATH      $topdir/lib64/R/include
prepend-path    C_INCLUDE_PATH          $topdir/lib64/R/include/R_ext
prepend-path    CXX_INCLUDE_PATH        $topdir/lib64/R/include/R_ext
prepend-path    CPLUS_INCLUDE_PATH      $topdir/lib64/R/include/R_ext

prepend-path    PKG_CONFIG_PATH         $topdir/lib64/pkgconfig
```

(continues on next page)

(continued from previous page)

```
prepend-path    MAN_PATH                $topdir/share/man
```

Additional Libraries

Keras GPU

Follow the steps below to set up Keras with GPU support in R.

Note: For the installation, R will be downloaded from Conda because of the Keras library when it is installed does NOT recognize the TensorFlow installation if the R built by us is used.

1. Load the Python 3 module.

```
$ module load python/3.6.8_intel-2019_update-4
```

2. Create the environment in Conda.

```
$ conda create -n r-tensorflow keras-gpu cudatoolkit=9.2
```

Warning: It is mandatory to specify the CUDA Toolkit version because the current driver version supports up to CUDA 9.2. If the version is omitted conda will install the latest (CUDA 10.0).

3. Activate the environment and install R

```
$ source activate r-tensorflow
$ conda install r-essentials r-base
```

4. Log in to the GPU node through Slurm.

```
$ srun -n 1 -t 60 -p accel --gres gpu:4 --pty bash
```

Warning: You only can log in to the GPU node if you have permissions to launch jobs in the "accel" partition. Ask the administrator if you are not sure.

5. Install the Keras library.

```
$ source activate r-tensorflow
$ R
```

```
> install.packages("devtools")
> library(devtools)
> install_github("rstudio/keras")
```

6. Test if GPUs are recognized.

```
> library(keras)
> k=backend()
```

(continues on next page)

(continued from previous page)

```
> sess = k$get_session()
> sess$list_devices()
```

7. Check if the processes are running in the GPUs.

Note: Login to the GPU node using another terminal session. `ssh compute-0-5`.

```
$ watch nvidia-smi
```

```
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 396.26                 Driver Version: 396.26                    |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|===============================+======================+======================|
|   0  Tesla K80           On   | 00000000:05:00.0 Off |                    0 |
| N/A   30C    P0    73W / 149W |     72MiB / 11441MiB |      0%      Default |
+-------------------------------+----------------------+----------------------+
|   1  Tesla K80           On   | 00000000:06:00.0 Off |                    0 |
| N/A   30C    P0    73W / 149W |     72MiB / 11441MiB |      0%      Default |
+-------------------------------+----------------------+----------------------+
|   2  Tesla K80           On   | 00000000:84:00.0 Off |                    0 |
| N/A   34C    P0    57W / 149W |      0MiB / 11441MiB |      0%      Default |
+-------------------------------+----------------------+----------------------+
|   3  Tesla K80           On   | 00000000:85:00.0 Off |                    0 |
| N/A   26C    P8    31W / 149W |      0MiB / 11441MiB |      0%      Default |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                       GPU Memory |
|  GPU       PID   Type   Process name                             Usage      |
|=============================================================================|
|    0     31882      C   ...onda/envs/r-tensorflow/lib/R/bin/exec/R    61MiB |
|    1     31882      C   ...onda/envs/r-tensorflow/lib/R/bin/exec/R    61MiB |
+-----------------------------------------------------------------------------+
```

Authors

- Hamilton Tobon Mosquera <htobonm@eafit.edu.co>
- Johan Sebastián Yepes Ríos <jyepesr1@eafit.edu.co>

R - 4.0.3

Basic information

- **Deploy date:** 29 January 2021
- **Official Website:** https://www.r-project.org/
- **License:** https://www.r-project.org/Licenses/
- **Installed on:** *Apolo II*

Dependencies

- zlib >= 1.2.5
- bzip2 >= 1.0.6 compiled with `-fPIC` option
- xz (liblzma) REF https://tukaani.org/xz/
- PCRE >= 8.20 > 10.0 with UTF-8 support
- curl >= 7.22 with https support
- libicu-4.2.1-14.el6.x86_64 and libicu-devel-4.2.1-14.el6.x86_64
- BLAS and LAPACK (Optional) with OpenBlas or MKL
- JDK >= 1.7.0

Installation

After the dependencies have been resolved and configured, the **R** installation can start.

Note: The GCC Compiler will be used to build R.

1. Download the selected version from an official mirror (https://cran.r-project.org/mirrors.html).

```
$ wget https://cran.r-project.org/src/base/R-4/R-4.0.3.tar.gz
$ tar -zxvf R-4.0.3.tar.gz
```

2. Load the corresponding modules to set the building environment.

Note: For BLAS and LAPACK is recommended to use Intel MKL (https://software.intel.com/en-us/articles/build-r-301-with-intel-c-compiler-and-intel-mkl-on-linux)

```
$ module purge # Clean predefined environment
$ module load zlib/1.2.11_gcc-7.4.0 bzip2/1.0.6 lapack/3.9.0 gcc/7.4.0 pcre/8.39 xz/5.
↪2.2 curl/7.51.0 java/jdk-1.8.0_112 mkl/19.0.4
```

3. Configure and build the sources.

```
$ CXX98="gcc -std=c++98" CXX11="gcc -std=c++11" CXX14="gcc -std=c++14" CXX17="gcc -
↪std=c++17" \
    ../configure --enable-lto="bootstrap-lto" --prefix=/share/apps/r/4.0.3/gcc/7.4.0 \
    --enable-java --build=x86_64-redhat-linux --enable-R-shlib \
    --with-blas="-lmkl_blas95_lp64 -liomp5 -lpthread" \
    --with-lapack="-lmkl_scalapack_lp64" --enable-BLAS-shlib \
    --without-x --enable-memory-profiling \
    LDFLAGS="$LDFLAGS \
            -L/share/apps/bzip2/1.0.6/lib \
            -L/share/apps/pcre/8.39/lib \
            -L/share/apps/xz/5.2.2/lib" \
    CFLAGS="$CFLAGS -fPIC -fopenmp -O3 -mavx2 \
            -I/share/apps/bzip2/1.0.6/include \
            -I/share/apps/pcre/8.39/include \
            -I/share/apps/xz/5.2.2/include" \
```

(continues on next page)

(continued from previous page)

```
    CXXFLAGS="$CXXFLAGS -fPIC -fopenmp -O3 -maxv2 \
            -I/share/apps/bzip2/1.0.6/include \
            -I/share/apps/xz/5.2.2/include" \
    FFLAGS="$FFLAGS -fPIC -fopenmp -O3 -mavx2" \
    FCFLAGS="$FCFLAGS -fPIC -fopenmp -O3 -mavx2" \
    --with-pcre1=/share/apps/pcre/8.39/


$ make -j10
```

4. Make the tests.

```
$ make check
```

If problems with the test `reg-packages.Rout` arise, ignore it, it seems to be a problem with the NFS, check here.

5. Install.

```
$ sudo mkdir -p /share/apps/r/4.0.3/gcc/7.4.0
$ sudo make install
```

Module

The following is the module used for this version.

```
##%Module1.0#####################################################################
##
## modules r/4.0.3_gcc-7.4.0_mkl
##
## /share/apps/modules/r/3.6.1_gcc-5.4.0_mkl  Written by Johan Yepes
##

proc ModulesHelp { } {
    puts stderr "\tR/4.0.3_gcc-7.4.0_mkl - sets the Environment for R in \
    \n\tthe share directory /share/apps/r/4.0.3/gcc/7.4.0\n"
}

module-whatis "\n\n\tSets the environment for R language \
            \n\tbuilt with GCC 7.4.0 and Intel MKL 2017 (Update-1)version \
            \n\t(Update-1)\n"

# for Tcl script use only
set       topdir     /share/apps/r/4.0.3/gcc/7.4.0
set       version    4.0.3
set       sys        x86_64-redhat-linux

conflict r

module load java/jdk-1.8.0_112 intel/2017_update-1 mkl/2017_update-1 gcc/5.4.0

prepend-path    PATH                    $topdir/bin

prepend-path    LD_LIBRARY_PATH         $topdir/lib64/R/lib
prepend-path    LD_RUN_PATH             $topdir/lib64/R/lib
prepend-path    LIBRARY_PATH            $topdir/lib64/R/lib
```

(continues on next page)

(continued from previous page)

```
prepend-path    LD_LIBRARY_PATH         $topdir/lib64/R/modules
prepend-path    LD_RUN_PATH             $topdir/lib64/R/modules
prepend-path    LIBRARY_PATH            $topdir/lib64/R/modules

prepend-path    C_INCLUDE_PATH          $topdir/lib64/R/include
prepend-path    CXX_INCLUDE_PATH        $topdir/lib64/R/include
prepend-path    CPLUS_INCLUDE_PATH      $topdir/lib64/R/include
prepend-path    C_INCLUDE_PATH          $topdir/lib64/R/include/R_ext
prepend-path    CXX_INCLUDE_PATH        $topdir/lib64/R/include/R_ext
prepend-path    CPLUS_INCLUDE_PATH      $topdir/lib64/R/include/R_ext

prepend-path    PKG_CONFIG_PATH         $topdir/lib64/pkgconfig

prepend-path    MAN_PATH                $topdir/share/man
```

Additional Libraries

We recommend for users that need additional libraries in R to use Anaconda. This is because we cannot guarantee that the library will fully work, each library may need different dependencies that we may or may not be able to install and guarantee its functionality.

The following is an example on how to install R in conda with an additional library called `dada2`.

1. Load the Python 3 module.

```
$ module load python/3.7_miniconda-4.8.3
```

2. Create the environment in Conda.

```
$ conda create -n ENVIRONMENT_NAME
```

3. Activate the environment and install R

```
$ conda activate ENVIRONMENT_NAME
$ conda config --add channels bioconda
$ conda config --add channels conda-forge
$ conda install bioconductor-dada2=1.16 r-base r-essentials
```

Note: If the package is not available in conda, please install it using the R version of conda (See the instruction above to install R in conda) inside the R Studio interpreter like this: `install.packages("<package_name>");`.

4. Make sure you activate the environment in the `slurm_file` if you are going to run tasks with this method.

```
#!/bin/bash

#SBATCH --job-name=test_123       # Job name
#SBATCH --mail-type=ALL          # Mail notification
#SBATCH --mail-user=tdnavarrom@eafit.edu.co  # User Email
#SBATCH --output=%x.%j.out # Stdout (%j expands to jobId)
#SBATCH --error=%x.%j.err  # Stderr (%j expands to jobId)
#SBATCH --ntasks=3
#SBATCH --ntasks-per-node=1
#SBATCH --cpus-per-task=6           # Number of tasks (processes)
```

(continues on next page)

(continued from previous page)

```
#SBATCH --time=13-23:01:00            # Walltime
#SBATCH --partition=longjobs         # Partition

## load module
module load python/3.7_miniconda-4.8.3
source activate r-test

## run code
Rscript simple_script.R
conda deactivate
```

Authors

- Tomás David Navarro Múnera <tdnavarrom@eafit.edu.co>

R - 4.1.2

Basic Information

- **Installation Date:** 14/02/2022
- **URL:** https://www.r-project.org/
- **Apolo Version:** Apolo II and later
- **License:** https://www.r-project.org/Licenses/

Dependencies

- OneApi compiler

Installation

1. Load the miniconda module.

```
$ module load miniconda3-4.10.3-oneapi-2022.0.0-2mgeehu
```

2. Change to a conda enviroment different from the base one.

```
$ conda activate ENV_NAME
```

If you have not created a conda enviroment, you can create it with the command

```
$ conda create --name ENV_NAME
```

Note: Creating a conda enviroment or changing between enviroments will not hinder your capabilities in navigating in the console as you normally would do.

3. Install R

```
$ conda install -c conda-forge r-base=4.1.2
```

Note: We recommend using the conda-forge channel for the installation as well as the installation of other R packages, since it has the most recent packages.

4. Check the installation.

```
$ conda list r-base
```

You should be able to see the name of the package, version installed and the channel it was downloaded from.

Installing R packages with conda

1. Move to the conda enviroment you want to install the packages in.

```
$ conda activate ENV_NAME
```

Note: If you don't know in what conda enviroment you currently are, you can check with the command $ conda env list, the enviroment you have activated will be marked with an asterisk.

2. Install the package.

```
$ conda install PACKAGE_NAME
```

Note: Before installing the package, to make sure the one you are installing is the correct one, we recomend that the user looks up beforehand the name of the package in the anaconda repository: https://anaconda.org/search. Here you can also find the command to install the packages if you have any problem installing them.

Note: With this method you can install packages for R without having to use install.packages("package_name") inside R.

Installing R packages with R Studio Interpreter

1. load the conda module

```
$ module load miniconda3-4.10.3-oneapi-2022.0.0-2mgeehu
```

2. activate the enviroment where R is installed

```
$ conda activate ENV_NAME
```

Note: If you don't know in what conda enviroment you currently are, you can check with the command $ conda env list, the enviroment you have activated will be marked with an asterisk.

3. Enter to R Studio interpreter

```
$ R
```

Once you have enter to the R Studio interpreter you can install the packages with the command:

```
install.packages("package_name")
```

Running Example

```
#!/bin/bash

#SBATCH --partition=longjobs                    # Partition
#SBATCH --nodes=1
#SBATCH --ntasks=1                              # Number of tasks (processes)
#SBATCH --time=1:00                            # Walltime
#SBATCH --job-name=test_r                       # Job name
#SBATCH --output=%x_%j.out                      # Stdout (%x-jobName, %j-jobId)
#SBATCH --error=%x_%j.err                       # Stderr (%x-jobName, %j-jobId)
#SBATCH --mail-type=ALL                         # Mail notification
#SBATCH --mail-user=jmonsalve@eafit.edu.co       # User Email


##### ENVIRONMENT CREATION #####
module load miniconda3-4.10.3-oneapi-2022.0.0-2mgeehu
source activate test1

##### JOB COMMANDS ####
Rscript simple_script.r
```

Resources

- https://docs.anaconda.com/anaconda/user-guide/tasks/using-r-language/

  Authors

- Jacobo Monsalve Guzman <jmonsalve@eafit.edu.co>

R - 4.1.3

Basic Information

- **Instalation Date:** 18/03/2022
- **URL:** https://www.r-project.org/
- ** APolo Version** Apolo II
- ** License:** https://www.r-project.org/Licenses/

Dependencies

- zlib >= 1.25
- bzip2 >= 1.0.6

- xz
- pcre2 >= 10.30
- curl >= 7.28.0 with https support
- JDK
- mpi
- mkl
- OneApi compiler
- lapack

Installation

After the dependencies have been resolved and configured, the **R** installation can start.

Note: The intel compiler will be used to build R

1. Download the selected version from an official mirror (https://cran.r-project.org/mirrors.html).

```
$ wget https://cran.r-project.org/src/base/R-4/R-4.1.3.tar.gz
$ tar xvzf R-4.1.3.tar.gz
```

2. Load the corresponding modules to set the building environment.

```
$ module load zlib/1.2.11_Intel_oneAPI-2022_update-1 bzip2/1.1.0_Intel_
↪oneAPI-2022_update-1  xz/5.2.5_Intel_oneAPI-2022_update-1 pcre2/10.39_
↪Intel_oneAPI-2022_update-1 curl/7.82.0_Intel_oneAPI-2022_update1
$ module load java/jdk-17.0.2 intel/2022_oneAPI-update1 mpi/2021.5.1 mkl/
↪2022.0.2 lapack/3.10.0_Intel_oneAPI-2022_update-1
```

3. Configure the resources

```
./configure --prefix=/share/apps/r/4.1.3/Intel_oneAPI-2022_update-1 \
--build=x86_64-redhat-linux --enable-R-shlib \
--with-blas="-L/share/apps/intel/oneAPI/2022_update-1/mkl/2022.0.2/lib/
↪intel64 \
-lmkl_rt -liomp5 -lpthread" --with-lapack --enable-BLAS-shlib \
--without-x --enable-memory-profiling \
LDFLAGS="$LDFLAGS -qopenmp -L/share/apps/bzip2/1.1.0/Intel_oneAPI-2022_
↪update-1/lib64 \
-L/share/apps/pcre2/10.39/Intel_oneAPI-2022_update-1/lib64 \
-L/share/apps/xz/5.2.5/Intel_oneAPI-2022_update-1/lib" \
CFLAGS="$CFLAGS -fPIC -qopenmp -O3 -ipo -xHost \
-I/share/apps/bzip2/1.1.0/Intel_oneAPI-2022_update-1/include \
-I/share/apps/pcre2/10.39/Intel_oneAPI-2022_update-1/include \
-I/share/apps/xz/5.2.5/Intel_oneAPI-2022_update-1/include" \
CXXFLAGS="$CXXFLAGS -fPIC -qopenmp -O3 -ipo -xHost \
-I/share/apps/bzip2/1.1.0/Intel_oneAPI-2022_update-1/include \
-I/share/apps/xz/5.2.5/Intel_oneAPI-2022_update-1/include" \
FFLAGS="$FFLAGS -fPIC -qopenmp -O3 -ipo -xHost" \
FCFLAGS="$FCFLAGS -fPIC -qopenmp -O3 -ipo -xHost" --with-readline=no
```

4. Before doing make, delete or comment the following line on src/include/config.h.in

```
#undef HAVE_MATHERR
```

5. Build the sources

```
$ make -j 16
$ make -j 16 install
```

Module

The following is the module used for this version.

```
##%Module1.0#####################################################################
↪###
##
## modules r/4.1.3_Intel_oneAPI-2022_update-1
##
## /share/apps/r/4.1.3/Intel_oneAPI-2022_update-1  Written by Bryan Lopez␣
↪Parra
##

proc ModulesHelp { } {
    puts stderr "\tR/4.1.3_Intel_oneAPI-2022_update-1 - sets the Environment␣
↪for R in \
    \n\tthe share directory /share/apps/r/4.1.3/Intel_oneAPI-2022_update-1\n"
}

module-whatis "\n\n\tSets the environment for R language \
            \n\tbuilt with  Intel MKL oneAPI 2022 (Update-1)version \
            \n\t(Update-1)\n"

# for Tcl script use only
set       topdir     /share/apps/r/4.1.3/Intel_oneAPI-2022_update-1
set       version    4.1.3
set       sys        x86_64-redhat-linux

conflict r

module load zlib/1.2.11_Intel_oneAPI-2022_update-1 bzip2/1.1.0_Intel_oneAPI-
↪2022_update-1  xz/5.2.5_Intel_oneAPI-2022_update-1 pcre2/10.39_Intel_
↪oneAPI-2022_update-1 curl/7.82.0_Intel_oneAPI-2022_update1
module load java/jdk-17.0.2 intel/2022_oneAPI-update1 mpi/2021.5.1 mkl/2022.
↪0.2 lapack/3.10.0_Intel_oneAPI-2022_update-1

prepend-path    PATH                    $topdir/bin

prepend-path    LD_LIBRARY_PATH         $topdir/lib64/R/lib
prepend-path    LD_RUN_PATH             $topdir/lib64/R/lib
prepend-path    LIBRARY_PATH            $topdir/lib64/R/lib
prepend-path    LD_LIBRARY_PATH         $topdir/lib64/R/modules
prepend-path    LD_RUN_PATH             $topdir/lib64/R/modules
prepend-path    LIBRARY_PATH            $topdir/lib64/R/modules

prepend-path    C_INCLUDE_PATH          $topdir/lib64/R/include
prepend-path    CXX_INCLUDE_PATH        $topdir/lib64/R/include
prepend-path    CPLUS_INCLUDE_PATH      $topdir/lib64/R/include
```

(continues on next page)

(continued from previous page)

```
prepend-path    C_INCLUDE_PATH          $topdir/lib64/R/include/R_ext
prepend-path    CXX_INCLUDE_PATH        $topdir/lib64/R/include/R_ext
prepend-path    CPLUS_INCLUDE_PATH      $topdir/lib64/R/include/R_ext

prepend-path    PKG_CONFIG_PATH         $topdir/lib64/pkgconfig

prepend-path    MAN_PATH                $topdir/share/man
```

Authors

- Jacobo Monsalve Guzmán <jmonsalve@eafit.edu.co>

Resources

Here you can find some resources such as documents, videos, tutorials that will support the use of R.

- R para Principiantes
- R Tutorials
- HOWTO: Install Local R Packages

Packrat: a dependency management system for R

Use packrat[1] to make your R projects more:

- **Isolated:** Installing a new or updated package for one project won't break your other projects, and vice versa. That's because packrat gives each project its own private package library.
- **Portable:** Easily transport your projects from one computer to another, even across different platforms. Packrat makes it easy to install the packages your project depends on.
- **Reproducible**: Packrat records the exact package versions you depend on, and ensures those exact versions are the ones that get installed wherever you go.

When you are using packrat you are no longer in an ordinary R project; you are in a Packrat project. The main difference is that a packrat project has its own private package library. Any packages you install from inside a packrat project are only available to that project; and packages you install outside of the project are not available to the project.[2]

Table of Contents

[1] https://rstudio.github.io/packrat/
[2] https://rstudio.github.io/packrat/walkthrough.html

Basic concepts

Packrat enhances your project directory by storing your package dependencies inside it, rather than relying on your personal R library that is shared across all of your other R sessions. We call this directory your **private package library** (or just **private library**). When you start an R session in a packrat project directory, R will only look for packages in your private library; and anytime you install or remove a package, those changes will be made to your private library.

Unfortunately, private libraries don't travel well; like all R libraries, their contents are compiled for your specific machine architecture, operating system, and R version. Packrat lets you **snapshot** the state of your private library, which saves to your project directory whatever information packrat needs to be able to recreate that same private library on another machine. The process of installing packages to a private library from a snapshot is called **restoring**.[1]

Installation

1. Load the R module that best suits your needs, for example:

```
$ module load r/3.6.3_gcc-7.4.0
```

Note: The necessary dependencies will be loaded automatically once the R module is loaded.

1. start R by typing R in the terminal:

```
$ R

R version 3.6.3 (2020-02-29) -- "Holding the Windsock"
Copyright (C) 2020 The R Foundation for Statistical Computing
Platform: x86_64-pc-linux-gnu (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

  Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.
>
```

2. Once R has started we must install the packrat package.

```
> install.packages("packrat")
```

3. The following warning will appear since you as a user do not have permission to save packages in the default library.

```
    Warning in install.packages("packrat") :
      'lib = "/share/apps/r/3.6.3/gcc/7.4.0/lib64/R/library"' is not writable
    Would you like to use a personal library instead? (yes/No/cancel)
```

(continues on next page)

(continued from previous page)

```
R is going to ask you if you want to create a personal library, it will save the␣
↪packages you install, so type **yes**.
```

```
Would you like to use a personal library instead? (yes/No/cancel) yes
Would you like to create a personal library
'~/R/x86_64-pc-linux-gnu-library/3.6'
to install packages into? (yes/No/cancel) yes
```

4. Now you are going to choose a mirror to download the package, it is recommended to choose the closest to your location for a faster download. In this case we will choose Colombia (21).

```
--- Please select a CRAN mirror for use in this session ---
Secure CRAN mirrors

 1: 0-Cloud [https]
 2: Australia (Canberra) [https]
 3: Australia (Melbourne 1) [https]
 4: Australia (Melbourne 2) [https]
 5: Australia (Perth) [https]
 6: Austria [https]
 7: Belgium (Ghent) [https]
 8: Brazil (BA) [https]
 9: Brazil (PR) [https]
10: Brazil (RJ) [https]
11: Brazil (SP 1) [https]
12: Brazil (SP 2) [https]
13: Bulgaria [https]
14: China (Beijing 1) [https]
15: China (Beijing 2) [https]
16: China (Hong Kong) [https]
17: China (Guangzhou) [https]
18: China (Lanzhou) [https]
19: China (Nanjing) [https]
20: China (Shanghai 1) [https]
21: Colombia (Cali) [https]
...

Selection: 21
```

5. To start your private library just run the following command and from this point on all the packages you install will be in the project's private library.

```
> packrat::init("<project_directory path>")
```

Note: If you are located in the project's directory, you can omit the path.

Commands

Some common commands in packrat.[2]

- To create a packrat project.

```
> packrat::init("<project_directory_path>")
```

- To install a required package.

```
> install.packages("<package_name>")
```

- To snapshot the project to save the changes.

```
> packrat::snapshot()
```

- To see the current status of the project.

```
> packrat::status()
```

- To remove a package.

```
> remove.packages("<package_name>")
```

- To restore the removed packages.

```
> packrat::restore()
```

Note: Remember to save your workspace before exiting. The next time you run R in the project directory, packrat will be automatically activated.

References

Author

- Laura Sánchez Córdoba <lsanchezc@eafit.edu.co>

References

3.2.4 Ruby

[1] Ruby is an interpreted, high-level, general-purpose programming language. It was designed and developed in the mid-1990s by Yukihiro "Matz" Matsumoto in Japan.

Ruby is dynamically typed and uses garbage collection. It supports multiple programming paradigms, inclding procedural, object-oriented, and functional programming. According to the creator, Ruby was influenced by Perl, Smalltalk, Eiffel, Ada, Basic, and Lisp.

Ruby - 2.6.4p104

Basic information

- **Deploy date:** 28 August 2019
- **Official Website:** https://www.ruby-lang.org/en/

[1] Wikipedia contributors. (2019, October 1). Ruby (programming language). In Wikipedia, The Free Encyclopedia. Retrieved 17:25, October 2, 2019, from https://en.wikipedia.org/w/index.php?title=Ruby_(programming_language)&oldid=919041875

- **License:** https://www.ruby-lang.org/en/about/license.txt
- **Installed on:** *Apolo II*, *Cronos*

Installation

Note: The Intel Compiler will be used to build Ruby.

1. Download the selected version from an official mirror (https://www.ruby-lang.org/en/downloads/).

```
$ wget https://cache.ruby-lang.org/pub/ruby/2.6/ruby-2.6.4.tar.gz
$ tar -zxvf ruby-2.6.4.tar.gz
```

2. Load the corresponding modules to set the building environment.

```
$ module purge # Clean predefined environment
$ module load intel/19.0.4
```

3. Configure and build the sources.

```
$ cd ruby-2.6.4
$ mkdir build && cd build
$ debugflags="" CFLAGS="-O3 -xHost" CXXFLAGS="-O3 -xHost" ../configure --prefix=/
↪share/apps/ruby/2.6.4p104/intel/19.0.4
$ make
$ make check
$ sudo su
$ module load intel/19.0.4
$ make install
```

Module

The following is the module used for this version.

```
##%Module1.0#####################################################################
##
## modulefile ruby/2.6.4_intel-19.0.4
##
## Written by Santiago Hidalgo Ocampo and Samuel David Palacios B.
##
proc ModulesHelp { } {
    global version modroot
    puts stderr "\truby - Interpretive, interactive programming language"
}

module-whatis "\n\n\tSets the environment for using ruby 2.6.4p104 binaries \
           \n\tprovided by Intel (2019 update 4)\n"

conflict ruby

# for Tcl script use only
set     topdir       /share/apps/ruby/2.6.4p104/intel/19.0.4
set     version      2.6.4p104
```

(continues on next page)

(continued from previous page)

```
set     sys          x86_64-redhat-linux

module load intel/19.0.4

prepend-path    PATH                            $topdir/bin

prepend-path    LD_LIBRARY_PATH                 $topdir/lib
prepend-path    LIBRARY_PATH                    $topdir/lib
prepend-path    LD_RUN_PATH                     $topdir/lib

prepend-path    C_INCLUDE_PATH                  $topdir/include
prepend-path    CXX_INCLUDE_PATH                $topdir/include
prepend-path    CPLUS_INCLUDE_PATH              $topdir/include

prepend-path    MANPATH                         $topdir/share/man

prepend-path    PKG_CONFIG_PATH                 $topdir/lib/pkgconfig
```

Authors

- Santiago Hidalgo Ocampo <shidalgoo1@eafit.edu.co>
- Samuel David Palacio Bernate <sdpalaciob@eafit.edu.co>

Resources

Here you can find some resources such as documents, videos, tutorials that will support the use of R.

- Ruby documentation

References

3.2.5 java

[1] Java is a programming language and computing platform first released by Sun Microsystems in 1995. It has evolved from humble beginnings to power a large share of today's digital world, by providing the reliable platform upon which many services and applications are built. New, innovative products and digital services designed for the future continue to rely on Java, as well. There are many applications and even some websites that will not function unless you have Java installed.

JDK-17.0.2

Table of Contents

[1] What is java and why do i need it. URL : https://www.java.com/en/download/help/whatis_java.html

Basic information

- **Official Website:** https://www.oracle.com/java/
- **License:** https://www.oracle.com/downloads/licenses/no-fee-license.html
- **Installed on:** Apolo II
- **Instalation date:** 14/03/2022

Prerequisites

Installation

1. Create the folder for java

```
$ mkdir -p /share/apps/java/17.0.2
```

2. Download the software

```
$ wget https://download.oracle.com/java/17/archive/jdk-17.0.2_linux-x64_
↪bin.tar.gz
$ tar xvzf jdk-17.0.2_linux-x64_bin.tar.gz
```

Module

```
##%Module1.0#####################################################################
↪###
##
## modules java/jdk-17.0.2
##
## /share/apps/modules/java/jdk-17.0.2  Written by Jacobo Monsalve
##
##

proc ModulesHelp { } {
    puts stderr "\tThe Java Development Kit (JDK) is an implementation of \
                \n\teither one of the Java Platform.\n"
}

module-whatis "\n\n\tSets the environment for using JDK-17.0.2\n"

# for Tcl script use only
set       topdir     /share/apps/java/jdk-17.0.2
set       version    jdk-17.0.2
set       sys        x86_64-redhat-linux
set       JAVA_HOME  $topdir
```

(continues on next page)

(continued from previous page)

```
conflict java

setenv              JAVA_HOME           $topdir

#setenv             JAVA_JRE            $topdir/jre

prepend-path        PATH                $topdir/bin

prepend-path        LD_LIBRARY_PATH     $topdir/lib
prepend-path        LIBRARY_PATH        $topdir/lib
prepend-path        LD_RUN_PATH         $topdir/lib

prepend-path        C_INCLUDE_PATH      $topdir/include
prepend-path        CXX_INCLUDE_PATH    $topdir/include
prepend-path        CPLUS_INCLUDE_PATH  $topdir/include

append-path         CLASSPATH           "."
#append-path        CLASSPATH           $JAVA_HOME/lib/dt.jar
#append-path        CLASSPATH           $JAVA_HOME/lib/tools.jar

#prepend-path       MANPATH             $topdir/man
```

How to use

```
$ module load java/jdk-17.0.2
```

Authors

- Jacobo Monsalve Guzman

Resources

Here you can find resources that will support the use of java.

- https://dev.java/

References

3.3 Scientific Applications

To find out what is the procedure to build, configure and use the installed scientific applications on our supercomputer (*Apolo II* and *Cronos*) you can review the following entries:

3.3.1 ABySS

ABySS[1] is a de novo sequence assembler intended for short paired-end reads and large genomes.

[1] ABySS. (2019, Oct 4). Retrieved October 25, 2019, from https://github.com/bcgsc/abyss

ABySS 2.2.3

Basic information

- **Deploy date:** 22 October 2019
- **Official Website:** https://github.com/bcgsc/abyss
- **License:** GNU GPL v3
- **Installed on:** *Apolo II*
- **Dependencies:**
  - *Google Sparsehash*
  - *ARCS*
  - *Tigmint*
  - Boost
  - Open MPI or MVAPICH
- **Optional dependencies:**
  - *Pigz*
  - Samtools
  - zsh

Installation

1. Before compiling and installing ABySS install all its dependencies. After installing all the dependencies run:

```
$ module load gcc/5.4.0
$ module load boost/1.67.0_intel-17.0.1
$ module load mvapich2/2.2.3a_gcc-5.4.0
$ module load sparsehash/2.0.3_intel-19.0.4
$ wget https://github.com/bcgsc/abyss/releases/download/2.2.3/abyss-2.2.3.
↪tar.gz
$ tar xvf abyss-2.2.3.tar.gz
$ cd abyss-2.2.3
$ mkdir build && cd build
$ CFLAGS="-O3 -mavx2 -fopenmp" CXXFLAGS="-O3 -mavx2 -fopenmp" ../
↪configure --prefix=/share/apps/abyss/2.2.3/gcc/5.4.0 --enable-mpich --
↪with-mpi=/share/apps/mvapich2/2.2.3a/gcc-5.4.0
$ make -j10
$ make check
```

   Make sure all the tests passed. Then install it:

```
$ sudo make install
```

2. Create and place the needed module file. Create a file with the following content:

Listing 27: 2.2.3_gcc-5.4.0

```
##%Module1.0#####################################################################
↪#####
##
## module load abyss/2.2.3_gcc-5.4.0
##
## /share/apps/modules/abyss/2.2.3_gcc-5.4.0
## Written by Vincent A. Arcila L and Hamilton Tobon Mosquera.
##

proc ModulesHelp {} {
        global version modroot
        puts stderr "Sets the environment for using abyss 2.2.3\
        \nin the shared directory /share/apps/abyss/2.2.3/gcc/5.4.0\
        \nbuilt with GCC 5.4.0."
}

module-whatis "(Name________) abyss"
module-whatis "(Version_____) 2.2.3"
module-whatis "(Compilers___) gcc-5.4.0"
module-whatis "(System______) x86_64-redhat-linux"
module-whatis "(Libraries___) "

# for Tcl script use only
set         topdir        /share/apps/abyss/2.2.3/gcc/5.4.0
set         version       2.2.3
set         sys           x86_64-redhat-linux

conflict abyss
module load mvapich2/2.2.3a_gcc-5.4.0
module load arcs/1.1.0_gcc-7.4.0
module load tigmint/1.1.2_miniconda-4.5.1

prepend-path   PATH            $topdir/bin
prepend-path   MANPATH         $topdir/share/man
```

Create the needed folder and place it:

```
$ sudo mkdir /share/apps/modules/abyss
$ sudo mv 2.2.3_gcc-5.4.0 /share/apps/modules/abyss/
```

Authors

- Hamilton Tobon-Mosquera <htobonm@eafit.edu.co>

3.3.2 ANSYS

Ansys[1] develops and markets finite element analysis software used to simulate engineering problems. The software creates simulated computer models of structures, electronics, or machine components to simulate strength, toughness, elasticity, temperature distribution, electromagnetism, fluid flow, and other attributes. Ansys is used to determine how a product will function with different specifications, without building test products or conducting crash tests.[5] For

[1] Wikipedia contributors. (2019, March 26). Ansys. In Wikipedia, The Free Encyclopedia. Retrieved 18:48, April 9, 2019, from https://en.wikipedia.org/w/index.php?title=Ansys&oldid=889612625

example, Ansys software may simulate how a bridge will hold up after years of traffic, how to best process salmon in a cannery to reduce waste, or how to design a slide that uses less material without sacrificing safety.

ANSYS 19.2

Table of Contents

Basic information

- **Official Website:** https://www.ansys.com/
- **License:** ANSYS Research License
- **Installed on:** *Apolo II* , *Cronos*

Tested on (Requirements)

- **OS base:** CentOS (x86_64) $\geq$ 6.6 (Rocks 6.2)

Installation

The following procedure is the easiest way to install ANSYS in a cluster.

1. Copy the three (3) iso files into a cluster location

```
$ tree ANSYS\ LINUX\ 19.2/
```

```
ANSYS LINUX 19.2/
├── DVD 1
│   └── ANSYS192_LINX64_DISK1.iso
├── DVD 2
│   └── ANSYS192_LINX64_DISK2.iso
└── DVD 3
    └── ANSYS192_LINX64_DISK3.iso
```

2. Mount each iso file into a different folder, you might create a folder structure in your home or `/tmp`.

```
$ mkdir -p ansys-19.2/ansys-{1,2,3}
$ sudo mount -o loop ANSYS192_LINX64_DISK1.iso ansys-19.2/ansys-1
$ sudo mount -o loop ANSYS192_LINX64_DISK2.iso ansys-19.2/ansys-2
$ sudo mount -o loop ANSYS192_LINX64_DISK3.iso ansys-19.2/ansys-3
```

3. After you have mounted all the iso files you have to log into the cluster through the GUI or via ssh using the `-X`.

```
$ ssh <user>@(apolo|cronos).eafit.edu.co -X
```

4. Go to the folder location of the first iso file and execute the `INSTALL` script.

```
$ cd ansys-19.2/ansys-1
$ ./INSTALL
```

It will open an ANSYS 19.2 GUI and just follow the steps. During the installation process, it will ask you for the other two folders where the iso files are mounted.

Note: Ensure to read the "Getting Started - Installation" and "System Requirements" PDFs shown in the GUI to meet the necessary requirements before to try to install the software.

---

5. After the installation is completed you have to create the corresponding module for ANSYS 19.2.

```
##%Module1.0#####################################################################
↪#####
##
## module load ansys/19.2
##
## /share/apps/modules/ansys/19.2
## Written by Johan Sebastian Yepes Rios
##

proc ModulesHelp {} {
     global version modroot
     puts stderr "Sets the environment for using Ansys 19.2\
                  \nin the shared directory /share/common-apps/ansys/19.2"
}

module-whatis "(Name________) Ansys"
module-whatis "(Version_____) 19.2"
module-whatis "(Compilers___) "
module-whatis "(System______) x86_64-redhat-linux"
module-whatis "(Libraries___) "

# for Tcl script use only
set         topdir        /share/common-apps/ansys/19.2/ansys_inc/v192
set         version       19.2
set         sys           x86_64-redhat-linux

conflict ansys

prepend-path              PATH        $topdir/ansys/bin
prepend-path              PATH        $topdir/CFX/bin
prepend-path              PATH        $topdir/autodyn/bin
prepend-path              PATH        $topdir/fluent/bin
prepend-path              PATH        $topdir/Icepak/bin
prepend-path              PATH        $topdir/polyflow/bin
```

(continues on next page)

(continued from previous page)

```
prepend-path                PATH        $topdir/AFD/bin
prepend-path                PATH        $topdir/TurboGrid/bin
```

Running Examples

In this section, there are some examples of how to use the different programs present in ANSYS through the Slurm Workload manager.

CFX5

ANSYS CFX is a high-performance computational fluid dynamics (CFD) software tool that delivers reliable and accurate solutions quickly and robustly across a wide range of CFD and multiphysics applications. CFX is recognized for its outstanding accuracy, robustness, and speed when simulating turbomachinery, such as pumps, fans, compressors and gas and hydraulic turbines.[1]

In the following example, we have decided to use Intel MPI because of the best choice for our current architecture. The example below was adapted from[2]

```
#!/bin/bash

#SBATCH -J jobcfx_test
#SBATCH -e jobcfx_test-%j.err
#SBATCH -o jobcfx_test-%j.out
#SBATCH -t 05:00:00
#SBATCH -n 64

module load ansys/19.2

#Generate lines of names of computational nodes (or hosts).
MYHOSTLIST=$(srun hostname | sort | uniq -c | awk '{print $2 "*" $1}' |␣
↪paste -sd, -)

#Run Ansys CFX.
cfx5solve -def prueba.def \
    -parallel \
    -start-method "Intel MPI Distributed Parallel" \
    -par-dist "$MYHOSTLIST" \
    -batch \
    -stdout-comms
```

Fluent

Fluent software contains the broad, physical modeling capabilities needed to model flow, turbulence, heat transfer and reactions for industrial applications. These range from air flow over an aircraft wing to combustion in a furnace, from bubble columns to oil platforms, from blood flow to semiconductor manufacturing and from clean room design to wastewater treatment plants. Fluent spans an expansive range, including special models, with capabilities to model in-cylinder combustion, aero-acoustics, turbomachinery and multiphase systems.[3]

[1] ANSYS CFX - ANSYS Official website. Retrieved April 10, 2019, from https://www.ansys.com/products/fluids/ansys-cfx

[2] HKHLR-How To - Run ANSYS CFX on an HPC-Cluster. Retrieved April 10, 2019, from https://www.hkhlr.de/sites/default/files/field_download_file/HowTo-ANSYS_CFX.pdf

[3] ANSYS Fluent - ANSYS Official website. Retrieved September 10, 2019, from https://www.ansys.com/products/fluids/ansys-fluent

How to run it in our cluster?. This example was adapted from[4]:

```
#!/bin/bash
#SBATCH --job-name=jobfluent_test
#SBATCH --error=jobfluent_test.err.%j
#SBATCH --output=jobfluent_test.out.%j
#SBATCH --time=00:10:00
#SBATCH --ntasks=16
#SBATCH --nodes=1
#SBATCH --partition=longjobs

# Load Ansys.
module load ansys/19.2

# Generate list of hosts.
MYHOSTLIST="hosts.$SLURM_JOB_ID"
srun hostname | sort > $MYHOSTLIST

# Run AnsysFluent.
fluent 3ddp \
    -g \
    -mpi=intel \
    -t $SLURM_NTASKS \
    -cnf=$MYHOSTLIST \
    -i fluent.jou \
    > fluent.out
```

References

Authors

- Johan Sebastián Yepes Ríos <jyepesr1@eafit.edu.co>

3.3.3 ARCS

ARCS[1] Scaffold genome sequence assemblies using 10x Genomics data.

ARCS 1.1.0

Basic information

- **Deploy date:** 22 October 2019
- **Official Website:** https://github.com/bcgsc/abyss
- **License:** GNU GPL v3
- **Installed on:** *Apolo II*
- **Dependencies:**
  - GCC greather than 4.4.7.

[4] HKHLR-How To - Run ANSYS Fluent on an HPC-Cluster. Retrieved September 10, 2019, from https://www.hkhlr.de/sites/default/files/field_download_file/HKHLR-HowTo-Ansys_Fluent.pdf

[1] ARCS. (2019, Oct 28). Retrieved December 2, 2019, from https://github.com/bcgsc/arcs

- *Google Sparsehash*
- Boost
- *LINKS*
- Autotools

Installation

1. Before compiling and installing ARCS install all its dependencies. After installing all the dependencies run:

```
$ git clone https://github.com/bcgsc/arcs.git
$ cd arcs
$ module load gcc/7.4.0
$ module load boost/1.67.0_intel-17.0.1
$ module load sparsehash/2.0.3_intel-19.0.4
$ ./autogen.sh
$ mkdir build && cd build
$ ../configure CFLAGS="-O3 -mavx2" CXXFLAGS="-O3 -mavx2" --prefix=/share/
↪apps/arcs/1.1.0/gcc/7.4.0
$ make -j4 && make check
```

Make sure all the checks run correctly. Then install it:

```
$ mkdir -p /share/apps/arcs/1.1.0/gcc/7.4.0
$ sudo make install
```

2. Create and place the needed module file. Create a file with the following content:

Listing 28: `1.1.0_gcc-7.4.0`

```
##%Module1.0#####################################################################
↪#####
##
## module load arcs/1.1.0_gcc-7.4.0
##
## /share/apps/modules/arcs/1.1.0_gcc-7.4.0
## Written by Vincent A. Arcila L and Hamilton Tobon Mosquera.
##

proc ModulesHelp {} {
         global version modroot
         puts stderr "Sets the environment for using arcs 1.1.0 \
         \nin the shared directory /share/apps/arcs/1.1.0/gcc/7.4.0 \
         \nbuilt with GCC 7.4.0."
}

module-whatis "(Name________) arcs"
module-whatis "(Version_____) 1.1.0"
module-whatis "(Compilers___) gcc-7.4.0"
module-whatis "(System______) x86_64-redhat-linux"
module-whatis "(Libraries___) "

# for Tcl script use only
set         topdir        /share/apps/arcs/1.1.0/gcc/7.4.0
set         version       1.1.0
set         sys           x86_64-redhat-linux
```

(continues on next page)

(continued from previous page)

```
conflict arcs
module load boost/1.67.0_intel-17.0.1
module load links/1.8.7_gcc-5.4.0_perl-5.26.1
module load sparsehash/2.0.3_intel-19.0.4


prepend-path    PATH            $topdir/bin
```

Create the needed folder and place it:

```
$ sudo mkdir /share/apps/modules/arcs
$ sudo mv 1.1.0_gcc-7.4.0 /share/apps/modules/arcs/
```

Authors

- Hamilton Tobon-Mosquera <htobonm@eafit.edu.co>

3.3.4 AutoDock

AutoDock is an automated procedure for predicting the interaction of ligands with biomacromolecular targets.

For additional information you can open those links:

- **Install** http://autodock.scripps.edu/downloads/faqs-help/faq/how-do-i-install-autodock-on-linux-and-mac-os-x
- **Download** http://autodock.scripps.edu/downloads/autodock-registration/autodock-4-2-download-page/
- **About** http://autodock.scripps.edu/#WHAT

AutoDock 4.2.6

Table of Contents

Basic information

- **Installation Date:** 19/10/2017

- **URL:** http://autodock.scripps.edu/
- **License:** GNU-LPLv2+
- **Installed on:** Apolo II

Installation

Using precompiled files

1. Download the binary for kernel version 2.6.32-504.16.2.el6.x86_64 (obtained by uname -r)

```
wget http://autodock.scripps.edu/downloads/autodock-registration/tars/
↪dist426/autodocksuite-4.2.6-x86_64Linux2.tar
```

2. Unzip the binaries

```
tar -xvf autodocksuite-4.2.6-x86_64Linux2.tar
```

3. We create the installation folder and move the downloaded binaries

```
cd x86_64Linux2
mkdir -p /share/apps/autodock/4.2.6/bin/
mv autodock4 autogrid4 /share/apps/autodock/4.2.6/bin/
```

Module

```
##%Module1.0#####################################################################
↪##
##
## module load autodock/4.2.6
##
## /share/apps/modules/autodock/4.2.6
## Written by Juan David Arcila Moreno
##

proc ModulesHelp {} {
    global version modroot
    puts stderr "Sets the environment for using autodock 4.2.6\
                \nin the shared directory \
                \n/share/apps/autodock/4.2.6/\
                \nbuilded with gcc-5.4.0"
}

module-whatis "(Name________) autodock"
module-whatis "(Version_____) 4.2.6"
module-whatis "(Compilers___) "
module-whatis "(System______) x86_64-redhat-linux"
module-whatis "(Libraries___) "

# for Tcl script use only
set         topdir        /share/apps/autodock/4.2.6
set         version       4.2.6
set         sys           x86_64-redhat-linux
```

(continues on next page)

(continued from previous page)

```
conflict autodock


prepend-path    PATH                    $topdir/bin
```

Mode of Use

```
module load autodock/4.2.6
autodock4
autogrid4
```

References

- http://autodock.scripps.edu/downloads/autodock-registration/autodock-4-2-download-page/%7C

Authors

- Juan David Arcila Moreno

3.3.5 autoDock_Vina

Description

AutoDock Vina is an open source program for molecular coupling. This program was designed byDr. Oleg Trott in the "Molecular Graphics" laboratory at the Scripps Research Institute.

AutoDock Vina

Table of Contents

Basic information

- **Installation date:** 19/09/2016
- **URL:** http://vina.scripps.edu/index.html
- **Apolo version:** Apolo I
- **License:** Apache License Version 2.0

Installation

1. First download the tar from the main page

```
$ wget https://ccpforge.cse.rl.ac.uk/gf/download/frsrelease/255/4726/dl_class_1.9.tar.
↪gz
$ tar -zxvf dl_class_1.9.tar.gz
```

1. configure the makefile

```
cd dl_class_1.9
cp build/MakePAR source/Makefile
cd source
module load openmpi/1.6.5_gcc-4.8.4
make gfortran
```

Module

```
##%Module1.0#####################################################################
##
## modules dl_class/1.9_openmpi-1.6.5_gcc-4.8.4
##
## /share/apps/modules/dl_class/1.9_openmpi-1.6.5_gcc-4.8.4
## Written by Mateo Gomez-Zuluaga
##

proc ModulesHelp { } {
    puts stderr "\tdl_classic/1.9_openmpi-1.6.5_gcc-4.8.4 - sets the
    \n\tEnvironment for DL_POLY Classic in the share directory
    \n\t/share/apps/dl_class/1.9/openmpi-1.6.5/gcc-4.8.4\n"
}

module-whatis "\n\n\tSets the environment for using DL_POLY Classic 1.9 \
               \n\tbuilded with openMPI 1.6.5 and GCC 4.8.4 version\n"

# for Tcl script use only
set       topdir     /share/apps/dl_class/1.9/openmpi-1.6.5/gcc-4.8.4
set       version    1.9
set       sys        x86_64-redhat-linux

module load openmpi/1.6.5_gcc-4.8.4

prepend-path    PATH $topdir/bin
```

Usage mode

```
module load dl_class/1.9_openmpi-1.6.5_gcc-4.8.4
```

References

- Mail from the people of the University of Cartagena
- Manual within the software package

Author

- Mateo Gómez Zuluaga

3.3.6 BayeScan

BayeScan[1] is a command line based open source software used for detecting natural selection from population-based genetic data.

BayeScan, since 2.1 version, uses OpenMP to implement parallel calculation using multicore processing.

BayeScan is distributed as a single package containing all the necessary files, including manual, source code[2] , executables and R scripts.

BayeScan - 2.1

Table of Contents

Basic information

- **Deploy date:** 23 July 2018
- **Official Website:** http://cmpg.unibe.ch/software/BayeScan/index.html
- **License:** GNU GPL v3
- **Installed on:** *Apolo II*, *Cronos*

[1] Matthieu Foll. (2012, January 21). BayeScan Official Page. Retrieved 16:47, August 8, 2018 from http://cmpg.unibe.ch/software/BayeScan/.
[2] Foll M and OE Gaggiotti (2008). A genome scan method to identify selected loci appropriate for both dominant and codominant markers: A Bayesian perspective. Genetics 180: 977-993

- **Available versions:** OpenMP

Installation

This entry covers the entire process performed in the installation and test of **BayeScan** on a cluster with the conditions described below.

Contents

Tested on (Requirements)

- **OS base:** CentOS (x86_64) $\geq$ 6.6 (Rocks 6.2)
- **Compiler:** Intel Parallel Studio XE Cluster Edition $\geq$ 17.0.1

Build process

1. Get the current **BayeScan** version from the **official webpage** and enter into the source directory.

```
wget http://cmpg.unibe.ch/software/BayeScan/files/BayeScan2.1.zip
upzip BayesScan2.1.zip
cd BayeScan/source
```

2. Modifie the `Makefile` in order to use *icpc* (`C++ Intel Compiler`), instead of *g++*.

Listing 29: `Makefile`

```
# BayeScan makefile

CC=icpc

bayescan_2.1: start.o beta.o dirichlet.o RJupdates.o MHupdates.o likelihood.o␣
↪read_write.o anyoption.o
	$(CC) -fopenmp -static -lpthread -o bayescan_2.1 start.o beta.o dirichlet.
↪o RJupdates.o MHupdates.o likelihood.o read_write.o anyoption.o

start.o: start.cpp errors.cpp anyoption.h global_defs.h
	$(CC) -fopenmp -c start.cpp errors.cpp

beta.o: beta.cpp global_defs.h
	$(CC) -fopenmp -c beta.cpp

dirichlet.o: dirichlet.cpp global_defs.h
	$(CC) -fopenmp -c dirichlet.cpp

RJupdates.o: RJupdates.cpp global_defs.h
	$(CC) -fopenmp -c RJupdates.cpp
```

(continues on next page)

(continued from previous page)

```
MHupdates.o: MHupdates.cpp global_defs.h
        $(CC) -fopenmp -c MHupdates.cpp

likelihood.o: likelihood.cpp global_defs.h
        $(CC) -fopenmp -c likelihood.cpp

read_write.o: read_write.cpp errors.cpp global_defs.h
        $(CC) -fopenmp -c read_write.cpp errors.cpp

anyoption.o: anyoption.cpp anyoption.h
        $(CC) -fopenmp -c anyoption.cpp

clean:
        rm *.o bayescan_2.1
```

3. Build **BayeScan**

```
make
```

Note: You must load the necessary modules to build BayeScan (i.e. `Intel Compiler`).

In Apolo II:

```
module load intel/2017_update-1
```

In Cronos:

```
module load intel/intel-18.0.2
```

---

4. Finally, create the installation directory and move the built executable.

Apolo

```
mkdir -p /share/apps/bayescan/2.1/intel-2017_update-1/bin
mv bayescan_2.1 /share/apps/bayescan/2.1/intel-2017_update-1/bin/bayescan
```

Cronos

```
mkdir -p /share/apps/bayescan/2.1/intel-18.0.2/bin
mv bayescan_2.1 /share/apps/bayescan/2.1/intel-18.0.2/bin/bayescan
```

Modulefile

Apolo II

Listing 30: `Module file`

```
#%Module1.0#####################################################################
##
## module load bayescan/2.1_intel-2017_update-1
##
## /share/apps/modules/bayescan/2.1_intel-2017_update-1
```

(continues on next page)

(continued from previous page)

```
## Written by Juan David Arcila Moreno
##

proc ModulesHelp {} {
     global version modroot
     puts stderr "Sets the environment for using bayescan 2.1_intel-2017_update-1\
                  \nin the shared directory \
                  \n/share/apps/bayescan/2.1/intel-2017_update-1\
                  \nbuilded with icpc compiler from Intel/2017_update-1. This␣
↪application uses OpenMP"
}

module-whatis "(Name________) bayescan"
module-whatis "(Version_____) 2.1_intel-2017_update-1"
module-whatis "(Compilers___) intel-2017_update-1"
module-whatis "(System______) x86_64-redhat-linux"
module-whatis "(Libraries___) "

# for Tcl script use only
set         topdir        /share/apps/bayescan/2.1/intel-2017_update-1
set         version       2.1_intel-2017_update-1
set         sys           x86_64-redhat-linux

conflict bayescan


prepend-path        PATH                        $topdir/bin
```

Cronos

Listing 31: `Module file`

```
##%Module1.0#####################################################################
##
## module load bayescan/2.1_intel-18.0.2
##
## /share/apps/modules/bayescan/2.1_intel-18.0.2
## Written by Juan David Arcila-Moreno
##

proc ModulesHelp {} {
     global version modroot
     puts stderr "Sets the environment for using bayescan 2.1_intel-18.0.2\
                  \nin the shared directory \
                  \n/share/apps/bayescan/2.1/intel-18.0.2\
                  \nbuilded with icpc from intel-18.0.2. This application uses OpenMP"
}

module-whatis "(Name________) bayescan"
module-whatis "(Version_____) 2.1_intel-18.0.2"
module-whatis "(Compilers___) icpc - intel-18.0.2"
module-whatis "(System______) x86_64-redhat-linux"
module-whatis "(Libraries___) "

# for Tcl script use only
set         topdir        /share/apps/bayescan/2.1/intel-18.0.2
set         version       2.1_intel-18.0.2
```

(continues on next page)

(continued from previous page)

```
set         sys           x86_64-redhat-linux

conflict bayescan


prepend-path        PATH                        $topdir/bin
```

Usage

In the following example we are going to run a test included in the source code of **BayeSacan**. Note that we are using one single compute-node, with 16 threads.

Listing 32: `slurm.sh`

```
#!/bin/bash
#SBATCH --partition=longjobs
#SBATCH --job-name=bayescan_test
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=16
#SBATCH --time=30:00
#SBATCH --output=bayescan_test%j.slurm.log

# Load module for default version
module load bayescan

bayscan -snp ~/bayescan/BayeScan2.1/input_examples/test_genotype_SNP.txt -all_trace -
↪out_prilot pilot
```

To run the previous example.

```
sbatch bayescan_run.sh
```

Note: In this example we are using the default version of **BayeScan** module. We recommend to specify the version. To use the version of this entry, pleace load the module as follow:

In Apolo II:

```
module load bayescan/2.1_intel-2017_update-1
```

In Cronos:

```
module load bayescan/2.1_intel-18.0.2
```

Performance Tests

The following test was performed in *Cronos*. We were comparing the build-in version and our builded version described in this entry. As input, we use one of the included tests of **BayeScan** source.

sbatch script for build-in version test:

```
#!/bin/bash
#SBATCH --partition=longjobs
#SBATCH --job-name=bayescan_test
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=16
#SBATCH --time=30:00
#SBATCH --output=bayescan_test%j.slurm.log

echo -n "Started at:"
date

# Run with build-in binaries
time ~/BayeScan2.1/binaries/BayeScan2.1_linux64bits -snp ~/bayescan/BayeScan2.1/input_
↪examples/test_genotype_SNP.txt -all_trace -out_prilot pilot
```

The test for our **builded version** was executed by the example provided in *Usage* section of this entry.

Note: As you can see in the sbatch script, the time was calculated using `time` command.

Results

Compiled version	Time		
	Real	User	Sys
Build-in	7m3.124s	46m39.444s	6m46.893s
Intel (icmp)	2m8.054s	33m26.063s	0m42.262s
change	99.970%	28.341%	89.613%

Authors

- Juan David Arcila-Moreno <jarcil13@eafit.edu.co>

3.3.7 BEAST2

BEAST is a cross-platform program for Bayesian inference using MCMC of molecular sequences. It is entirely orientated towards rooted, time-measured phylogenies inferred using strict or relaxed molecular clock models. It can be used as a method of reconstructing phylogenies but is also a framework for testing evolutionary hypotheses without conditioning on a single tree topology. BEAST uses MCMC to average over tree space, so that each tree is weighted proportional to its posterior probability. We include a simple to use user-interface program for setting up standard analyses and a suit of programs for analysing the results.

BEAST 2.4.5

Table of Contents

Basic information

- **Official Website:** http://www.beast2.org/
- **License:** GNU LESSER GENERAL PUBLIC LICENSE Version 2.1
- **Installed on:** Apolo II and Cronos
- **Installation date:** 07/02/2017

Tested on (Requirements)

- **OS base:** CentOS (x86_64) $\geq$ 6.6 (Rocks 6.2)
- **Dependencies:**
  - Java JDK - 1.8.0 u112

Installation

1. Download the desired version of the software (Source code - tar.gz)

```
cd /home/mgomezzul/apps/bpp/intel
wget https://github.com/CompEvol/beast2/releases/download/v2.4.5/BEAST.v2.
↪4.5.Linux.tgz
tar -zxvf BEAST.v2.4.5.Linux.tgz
```

2. After unpacking blast, continue with the following steps to finish the installation:

```
cd beast
sudo mkdir -p /share/apps/beast2/2.4.5
sudo cp -r bin/ lib/ templates/ examples/ images/ /share/apps/beast2/2.4.5
```

Module

```
#%Module1.0#####################################################################
↪###
##
## module beast2/2.4.5
##
## /share/apps/modules/beast2/2.4.5 Written by Juan David Pineda-Cárdenas
##

proc ModulesHelp { } {
```

(continues on next page)

(continued from previous page)

```
    puts stderr "\tbeast2/2.4.5 - sets the Environment for BEAST2 \
    \n\tin the share directory /share/apps/beast2/2.4.5\n"
}

module-whatis "\n\n\tSets the environment for using beast2\n"

# for Tcl script use only
set     topdir          /share/apps/beast2/2.4.5
set     version         2.4.5
set     sys             linux-x86_64

conflict beast2

module load java/jdk-1.8.0_112

prepend-path PATH               $topdir/bin

prepend-path CLASSPATH          $topdir/lib/beast.jar
prepend-path CLASSPATH          $topdir/lib/beast.src.jar
prepend-path CLASSPATH          $topdir/lib/DensiTree.jar
prepend-path CLASSPATH          $topdir/lib/launcher.jar
```

Use

```
module load beast2/2.4.5
```

Resources

- http://www.beast2.org/

Author

- Mateo Gómez Zuluaga

BEAST 2.6.3

Table of Contents

- *Resources*

Basic information

- **Official Website:** http://www.beast2.org/
- **License:** GNU LESSER GENERAL PUBLIC LICENSE Version 2.1
- **Installed on:** Apolo II
- **Installation date:** 23/02/2021

Tested on (Requirements)

- **OS base:** CentOS (x86_64) $\geq$ 6.6 (Rocks 6.2)
- **Dependencies:**
  - Java JDK - 1.8.0 u112

Installation

1. Download the desired version of the software (Source code - tar.gz)

```
cd /home/blopezp/Desktop/BEAST
wget https://github.com/CompEvol/beast2/releases/download/v2.6.3/BEAST.v2.
↪6.3.Linux.tgz
tar -zxvf BEAST.v2.6.3.Linux.tgz
2. After unpacking blast, continue with the following steps to finish the␣
↪installation:
```

2. After unpacking blast, continue with the following steps to finish the installation:

```
cd beast
sudo mkdir -p /share/apps/beast2/2.6.3
sudo cp -r bin/ lib/ templates / examples/ images/ /sahre/apps/beast2/2.6.
↪3
ls -p /share/apps/beast2/2.6.3
```

Module

```
##%Module1.0#####################################################################
↪###
##
## module beast2/2.6.3
##
## /share/apps/modules/beast2/2.6.3 Written by Bryan López Parra
##

proc ModulesHelp { } {
    puts stderr "\tbeast2/2.6.3 - sets the Environment for BEAST2 \
    \n\tin the share directory /share/apps/beast2/2.6.3\n"
```

(continues on next page)

(continued from previous page)

```
}

module-whatis "\n\n\tSets the environment for using beast2\n"

# for Tcl script use only
set     topdir          /share/apps/beast2/2.6.3
set     version         2.6.3
set     sys             linux-x86_64

conflict beast2

module load java/jdk-1.8.0_112

prepend-path PATH               $topdir/bin

prepend-path CLASSPATH          $topdir/lib/beast.jar
prepend-path CLASSPATH          $topdir/lib/beast.src.jar
prepend-path CLASSPATH          $topdir/lib/DensiTree.jar
prepend-path CLASSPATH          $topdir/lib/launcher.jar
```

Use

```
module load beast2/2.6.3
```

Slurm template

```
#!/bin/bash
#SBATCH --job-name=BEAST2-2.6.0-case
#SBATCH --partition=batch
#SBATCH --nodes=1
#SBATCH --ntasks=8
#SBATCH --time=1:00:00

module load beast2/2.6.3

beast -threads $SLURM_NTASKS testStarBeast.xml
```

Resources

- http://www.beast2.org/

Author

- Bryan López Parra <blopezp@eafit.edu.co>

BEAST 2.6.4

Table of Contents

Basic information

- **Official Website:** http://www.beast2.org/
- **License:** GNU LESSER GENERAL PUBLIC LICENSE Version 2.1
- **Installed on:** *Apolo II*
- **Installation date:** 08/06/2021

Tested on (Requirements)

- **Dependencies:**
  - Java JDK - 1.8.0 u112

Installation

1. Download the desired version of the software (Source code - tar.gz)

```
cd /home/jpossaz/sources/beast2
wget -O beast2.6.4.tgz "https://github-releases.githubusercontent.com/
↪15949777/faf69900-ae6f-11eb-8247-ca2b5a96b6dd?X-Amz-Algorithm=AWS4-HMAC-
↪SHA256&X-Amz-Credential=AKIAIWNJYAX4CSVEH53A%2F20210608%2Fus-east-1%2Fs3
↪%2Faws4_request&X-Amz-Date=20210608T204641Z&X-Amz-Expires=300&X-Amz-
↪Signature=4edf797e065ec87baa8a21dbd0cd5938f85a56d0f03e25535125646c65cfdbc2&
↪X-Amz-SignedHeaders=host&actor_id=13303029&key_id=0&repo_id=15949777&
↪response-content-disposition=attachment%3B%20filename%3DBEAST.v2.6.4.
↪Linux.tgz&response-content-type=application%2Foctet-stream"
tar -zxvf beast2.6.4.tgz
```

2. After unpacking, copy the files to the corresponding apps directory:

```
cd beast
sudo mkdir -p /share/apps/beast2/2.6.4
sudo cp -r bin/ lib/ templates/ examples/ images/ /share/apps/beast2/2.6.4
```

Module

```
#%Module1.0#####################################################################
↪###
##
## module /share/apps/modules/beast2/2.6.4
##
## /share/apps/beast2/2.6.4 Written by Juan Pablo Ossa Zapata
##

proc ModulesHelp { } {
    puts stderr "\tbeast2/2.6.4 - sets the Environment for BEAST2 \
    \n\tin the share directory /share/apps/beast2/2.6.4\n"
}

module-whatis "\n\n\tSets the environment for using beast2\n"

# for Tcl script use only
set     topdir          /share/apps/beast2/2.6.4
set     version         2.6.4
set     sys             linux-x86_64

conflict beast2

module load java/jdk-1.8.0_112

prepend-path PATH               $topdir/bin

prepend-path CLASSPATH          $topdir/lib/beast.jar
prepend-path CLASSPATH          $topdir/lib/beast.src.jar
prepend-path CLASSPATH          $topdir/lib/DensiTree.jar
prepend-path CLASSPATH          $topdir/lib/launcher.jar
```

Use

```
module load beast2/2.6.4
```

Example slurm job file:

```
#!/bin/sh
#SBATCH --partition=longjobs
#SBATCH --nodes=1
#SBATCH --cpus-per-task=4
#SBATCH --ntasks=1
#SBATCH --time=2:00:00
#SBATCH --job-name=testBEAST2
#SBATCH -o %x_%j.out      # File to which STDOUT will be written
#SBATCH -e %x_%j.err      # File to which STDERR will be written
#SBATCH --mail-type=ALL
#SBATCH --mail-user=jpossaz@eafit.edu.co

module load beast2/2.6.4
export OMP_NUM_THREADS=4
DISPLAY="" beast -threads 4 testRNA.xml
```

Make sure that your xml file can and will use all of the threads that you assign to the job.

Resources

- http://www.beast2.org/

Author

- Juan Pablo Ossa Zapata

3.3.8 BLAST

The Basic Local Alignment Search Tool (BLAST) finds regions of local similarity between sequences. The program compares nucleotide or protein sequences to sequence databases and calculates the statistical significance of matches. BLAST can be used to infer functional and evolutionary relationships between sequences as well as help identify members of gene families.

BLAST 2.6.0

Table of Contents

Basic information

- **Official Website:** https://blast.ncbi.nlm.nih.gov/Blast.cgi
- **License:** GNU LESSER GENERAL PUBLIC LICENSE Version 2.1
- **Installed on:** Apolo II and Cronos
- **Installation date:** 21/02/2017

Tested on (Requirements)

- **OS base:** CentOS (x86_64) $\geq$ 6.6 (Rocks 6.2)
- **Dependencies:**
  - python >= 2.7.11
  - libm (by default)

Installation

1. Download the desired version of the software (Source code - tar.gz)

```
cd /home/mgomezzul/apps/vsearch/gcc
wget ftp://ftp.ncbi.nlm.nih.gov/blast/executables/blast+/2.6.0/ncbi-blast-
↪2.6.0+-x64-linux.tar.gz
tar -zxvf ncbi-blast-2.6.0+-x64-linux.tar.gz
```

2. After unpacking blast, continue with the following steps to finish the installation:

```
cd ncbi-blast-2.6.0+
rm ChangeLog LICENSE ncbi_package_info README
sudo mkdir -p /share/apps/ncbi-blast/2.6.0
sudo mv bin /share/apps/ncbi-blast/2.6.0/
```

Module

```
#%Module1.0#####################################################################
↪###
##
## modules ncbi-blast+/2.6.0
##
## /share/apps/modules/ncbi-blast+/2.6.0_x86_64  Written by Mateo Gomez-
↪Zuluaga
##

proc ModulesHelp { } {
    puts stderr "\n\tThe NCBI Basic Local Alignment Search Tool (BLAST)␣
↪finds \
           \n\tregions of local similarity between sequences in the \
        \n\tshare directory /share/apps/ncbi-blast+/2.6.0\n"
}

module-whatis "\n\tSets the Environment for NCBI-BLAST 2.6.0\n"

# for Tcl script use only
set     topdir     /share/apps/ncbi-blast+/2.6.0
set     version    2.6.0
set     sys        x86_64-redhat-linux

conflict ncbi-blast+
module load intel/2017_update-1
module load mkl/2017_update-1
module load python/2.7.12_intel-2017_update-1

prepend-path PATH $topdir/bin
```

Slurm template

```
#!/bin/bash
#SBATCH --partition=longjobs
#SBATCH --nodes=1
```

(continues on next page)

(continued from previous page)

```
#SBATCH --ntasks-per-node=32
#SBATCH --time=1:00:00
#SBATCH --job-name=vsearch
#SBATCH -o result_%N_%j.out
#SBATCH -e result_%N_%j.err

export SBATCH_EXPORT=NONE
export OMP_NUM_THREADS=???

module load ncbi-blast/2.6.0_x86_64

xxx
```

Resources

- https://blast.ncbi.nlm.nih.gov/Blast.cgi

Author

- Mateo Gómez Zuluaga

3.3.9 BLAT

BLAT[1] on DNA is designed to quickly find sequences of 95% and greater similarity of length 25 bases or more. It may miss more divergent or shorter sequence alignments. It will find perfect sequence matches of 20 bases. BLAT on proteins finds sequences of 80% and greater similarity of length 20 amino acids or more. In practice DNA BLAT works well on primates, and protein BLAT on land vertebrates.

BLAT 36

Basic information

- **Deploy date:** 22 October 2019
- **Official Website:** https://genome.ucsc.edu/cgi-bin/hgBlat
- **License:** Custom license
- **Installed on:** *Apolo II*

Installation

1. To download run:

```
$ module load intel/19.0.4
$ wget https://genome-test.gi.ucsc.edu/~kent/src/blatSrc.zip
$ unzip batSrc.zip && cd blatSrc
```

To use the Intel's compiler modify the file `inc/common.mk`: like this:

[1] BLAT. (2014, Oct 11). Retrieved December 3, 2019, from https://genome.ucsc.edu/cgi-bin/hgBlat

```
CC=icc
COPT=-O3 -xHost
```

Create the makefile's needed directory `~/bin/x86_64`.

```
mkdir -p ~/bin/x86_64
```

Compile:

```
make -j6
```

Install:

```
$ sudo mkdir -p /share/apps/blat/36/intel/19.0.4/bin

# Make sure that the directory ~/bin/x86_64 has only the BLAT's␣
↪installation
$ sudo cp -r ~/bin/x86_64/* /share/apps/blat/36/intel/19.0.4/bin
```

2. Create and place the needed module file. Create a file with the following content:

Listing 33: `36_intel-19.0.4`

```
##%Module1.0#####################################################################
↪######
##
## modulefile blat/36_intel-19.0.4
##
## /share/apps/modules/blat/36_intel-19.0.4
## Written by Hamilton Tobon Mosquera.
proc ModulesHelp { } {
     global version modroot
     puts stderr "Sets the environment for using blat 36\
     \nin the shared directory /share/apps/blat/36/intel/19.0.4\
     \nbuilt with Intel 19.0.4."
}

module-whatis "(Name________) blat"
module-whatis "(Version_____) 36"
module-whatis "(Compilers___) Intel 19.0.4"
module-whatis "(System______) x86_64-redhat-linux"
module-whatis "(Libraries___) "

conflict blat

# for Tcl script use only
set     topdir       /share/apps/blat/36/intel/19.0.4
set     version      36
set     sys          x86_64-redhat-linux

prepend-path   PATH            $topdir/bin
```

Create the needed folder and place it:

```
$ sudo mkdir /share/apps/modules/blat
$ sudo mv 36_intel-19.0.4 /share/apps/modules/blat/
```

Authors

- Hamilton Tobon-Mosquera <htobonm@eafit.edu.co>

3.3.10 bowtie2

Bowtie 2 is an ultrafast and memory-efficient tool for aligning sequencing reads to long reference sequences. It is particularly good at aligning reads of about 50 up to 100s or 1,000s of characters, and particularly good at aligning to relatively long (e.g. mammalian) genomes. Bowtie 2 indexes the genome with an FM Index to keep its memory footprint small: for the human genome, its memory footprint is typically around 3.2 GB. Bowtie 2 supports gapped, local, and paired-end alignment modes.

bowtie2 2.3.0

Table of Contents

Basic information

- **Official Website:** http://bowtie-bio.sourceforge.net/bowtie2/index.shtml
- **License:** GNU GENERAL PUBLIC LICENSE Version 3
- **Installed on:** Apolo II and Cronos
- **Installation date:** 27/02/2017

Tested on (Requirements)

- **OS base:** CentOS (x86_64) $\geq$ 6.6 (Rocks 6.2)
- **Dependencies:**
  - Intel Parallel Studio XE Cluster Edition 2017 Update 1

Installation

1. Download the desired version of the software (Source code - tar.gz)

```
cd /home/mgomezzul/apps/bowtie2/
wget https://sourceforge.net/projects/bowtie-bio/files/bowtie2/2.3.0/
↪bowtie2-2.3.0-source.zip/download
unzip bowtie2-2.3.0-source.zip
```

2. Do the following changes in the Makefile

```
cd bowtie2-2.3.0
module load intel/2017_update-1
emacs Makefile
...
prefix = /share/apps/bowtie2/2.3.0/intel/2017_update-1
bindir = $(prefix)/bin

INC =
GCC_PREFIX = $(shell dirname `which icc`)
GCC_SUFFIX =
CC ?= $(GCC_PREFIX)/icc$(GCC_SUFFIX)
CPP ?= $(GCC_PREFIX)/icpc$(GCC_SUFFIX)
CXX ?= $(CPP)
HEADERS = $(wildcard *.h)
BOWTIE_MM = 1
BOWTIE_SHARED_MEM = 0
...
make 2>&1 | tee bowtie2-make.log
```

3. After compiling bowtie2, continue with the following steps:

```
sudo mkdir -p /share/apps/bowtie2/2.3.0/intel/2017_update-1
sudo chown -R mgomezzul.apolo /share/apps/bowtie2/2.3.0/intel/2017_update-
↪1
make install 2>&1 | tee bowtie2-make-install.log
sudo chown -R root.root /share/apps/bowtie2/2.3.0/intel/2017_update-1
```

Module

```
##%Module1.0#####################################################################
↪###
##
## module bowtie2/2.3.0_intel-2017_update-1
##
## /share/apps/modules/bowtie2/2.3.0_intel-2017_update-1     Written by␣
↪Mateo Gomez-Zuluaga
##

proc ModulesHelp { } {
    puts stderr "\tbowtie2/2.3.0_intel-2017_update-1 - sets the Environment␣
↪for Bowtie2 in \
    \n\tthe share directory /share/apps/bowtie2/2.3.0/intel/2017_update-1\n"
}

module-whatis "\n\n\tSets the environment for using Bowtie2 2.3.0 \
            \n\tbuilded with Intel Parallel Studio XE 2017\n"

# for Tcl script use only
```

(continues on next page)

(continued from previous page)

```
set       topdir     /share/apps/bowtie2/2.3.0/intel/2017_update-1
set       version    2.3.0
set       sys        x86_64-redhat-linux

module load intel/2017_update-1

prepend-path PATH    $topdir/bin
```

Usage mode

```
module load bowtie2/2.3.0_intel-2017_update-1
```

Resources

- http://bowtie-bio.sourceforge.net/bowtie2/index.shtml

Author

- Mateo Gómez Zuluaga

3.3.11 Bpp

BP&P: Bayesian analysis of genomic sequence data under the multispecies coalescent model

The BP&P program implements a series of Bayesian inference methods under the multispecies coalescent model. The analyses may include estimation of population size (theta's) and species divergence times (tau's), species tree estimation and species delimitation.

BP&P 3.3.0

Table of Contents

Basic information

- **Official Website:** http://abacus.gene.ucl.ac.uk/software/
- **License:** GNU GENERAL PUBLIC LICENSE Version 3
- **Installed on:** Apolo II and Cronos
- **Installation date:** 07/02/2017

Tested on (Requirements)

- **OS base:** CentOS (x86_64) $\geq$ 6.6 (Rocks 6.2)
- **Dependencies:**
  - Intel Parallel Studio XE 2017 Update 1 (Intel C Compiler)

Installation

1. Download the desired version of the software (Source code - tar.gz)

```
cd /home/mgomezzul/apps/bpp/intel
wget http://abacus.gene.ucl.ac.uk/software/bpp3.3.tgz
tar -zxvf bpp3.3.tgz
```

2. Manual compilation procedure.

```
cd bpp3.3/src
module load intel/2017_update-1
echo manual compilation
icc -o bpp -fast bpp.c tools.c -lm
icc -o bpp_sse -fast -DUSE_SSE -msse3 bpp.c tools.c -lm
icc -o bpp_avx -fast -DUSE_AVX -mavx bpp.c tools.c -lm
icc -o MCcoal -DSIMULATION -fast bpp.c tools.c -lm
```

3. After compiling BP&P, continue with the following steps:

```
sudo mkdir -p /share/apps/bpp/3.3/intel/2017_update-1/bin
sudo cp bpp bpp_sse bpp_avx MCcoal /share/apps/bpp/3.3/intel/2017_update-
↪1/bin
```

Module

```
##%Module1.0#####################################################################
↪###
##
## module bpp/3.3_intel-2017_update-1
##
## /share/apps/modules/bpp/3.3_intel-2017_update-1     Written by Mateo␣
↪Gomez-Zuluaga
##

proc ModulesHelp { } {
```

(continues on next page)

(continued from previous page)

```
    puts stderr "\tbpp/3.3_intel-2017_update-1 - sets the Environment for␣
↪Bpp in \
    \n\tthe share directory /share/apps/bpp/3.3/intel/2017_update-1\n"
}

module-whatis "\n\n\tSets the environment for using Bpp 3.3 \
            \n\tbuilded with Intel Parallel Studio XE 2017\n"

# for Tcl script use only
set       topdir     /share/apps/bpp/3.3/intel/2017_update-1
set       version    3.3
set       sys        x86_64-redhat-linux

module load intel/2017_update-1

prepend-path PATH    $topdir/bin
```

Usage mode

```
module load bpp/3.3_intel-2017_update-1
```

Resources

- http://abacus.gene.ucl.ac.uk/software/

Author

- Mateo Gómez Zuluaga

BP&P 4.3.8

Table of Contents

Basic information

- **Official Website:** https://github.com/bpp/bpp/

- **License:** GNU GENERAL PUBLIC LICENSE Version 3
- **Installed on:** Apolo II and Cronos
- **Installation date:** 07/02/2021

Tested on (Requirements)

- **OS base:** CentOS (x86_64) $\geq$ 6.6 (Rocks 6.2)
- **Dependencies:**
  - Intel Parallel Studio XE 2019 (Intel C Compiler)

Installation

1. Download the desired version of the software (Source code - tar.gz)

```
cd /home/tdnavarrom/apps/bpp/intel
wget https://github.com/bpp/bpp/releases/download/v4.3.8/bpp-4.3.8-linux-
↪x86_64.tar.gz
tar zxf v4.3.8.tar.gz
```

2. Manual compilation procedure.

```
cd bpp-4.3.8/src/
module load intel/19.0.4
make
```

3. After compiling BP&P, continue with the following steps:

```
sudo mkdir /share/apps/bpp/4.3.8/intel/19.0.4/bin -p
sudo cp src/bpp /share/apps/bpp/4.3.8/intel/19.0.4/bin/
```

Module

```
#%Module1.0#####################################################################
↪###
##
## module bpp/4.3.8_intel-19.0.4
##
## /share/apps/modules/bpp/4.3.8_intel-19.0.4     Written by Tomas Navarro
##

proc ModulesHelp { } {
    puts stderr "\tbpp/4.3.8_intel-19.0.4 - sets the Environment for Bpp in \
    \n\tthe share directory /share/apps/bpp/4.3.8/intel/19.0.4\n"
}

module-whatis "\n\n\tSets the environment for using Bpp 4.3.8 \
            \n\tbuilded with Intel Parallel Studio XE 2019\n"

# for Tcl script use only
set       topdir     /share/apps/bpp/4.3.8/intel/19.0.4
```

(continues on next page)

(continued from previous page)

```
set       version    4.3.8
set       sys        x86_64-redhat-linux

module load intel/19.0.4

prepend-path PATH    $topdir/bin
```

Usage mode

```
module load bpp/3.3_intel-2017_update-1
```

Resources

- https://github.com/bpp/bpp/

Authors

- Tomas Navarro <tdnavarrom@eafit.edu.co>

BP&P 4.4.1

Table of Contents

Basic information

- **Official Website:** https://github.com/bpp/bpp/
- **License:** GNU GENERAL PUBLIC LICENSE Version 3
- **Installed on:** Apolo II
- **Installation date:** 15/03/2022

Tested on (Requirements)

- **OS base:** Rocky Linux 8.5 (x86_64) $\geq$
- **Dependencies:**

- Intel oneAPI 2022 update 1 (Intel C Compiler)

Installation

1. Download the desired version of the software.

```
cd /home/blopezp/Modulos
git clone https://github.com/bpp/bpp.git
```

2. Manual compilation procedure.

```
cd bpp-4.4.1/src/
module load intel/2022_oneAPI-update1
make
```

3. After compiling BP&P, continue with the following steps:

```
sudo mkdir /share/apps/bpp/4.4.1/Intel_oneAPI-2022_update-1/bin -p
sudo cp src/bpp /share/apps/bpp/4.4.1/Intel_oneAPI-2022_update-1/bin
```

Module

```
##%Module1.0#####################################################################
↪###
##
## module bpp/4.4.1_Intel_oneAPI-2022_update-1
##
## /share/apps/bpp/4.4.1/Intel_oneAPI-2022_update-1     Written by Bryan␣
↪Lopez Parra
##

proc ModulesHelp { } {
    puts stderr "\tbpp/4.4.1_intel-2022_update-1 sets the Environment for␣
↪Bpp in \
    \n\tthe share directory /share/apps/bpp/4.4.1/Intel_oneAPI-2022_update-
↪1\n"
}

module-whatis "\n\n\tSets the environment for using Bpp 4.4.1 \
            \n\tbuilded with Intel oneAPI 2022 update 1\n"

# for Tcl script use only
set       topdir     /share/apps/bpp/4.4.1/Intel_oneAPI-2022_update-1
set       version    4.4.1
set       sys        x86_64-redhat-linux

module load intel/2022_oneAPI-update1

prepend-path PATH    $topdir/bin
```

Slurm template

```
#!/bin/bash
#SBATCH --partition=longjobs
#SBATCH --nodes=1
#SBATCH --ntasks=1
#SBATCH --time=14-00:00:00
#SBATCH --job-name=A10_All_L-D
#SBATCH -o result_%x_%j.out      # File to which STDOUT will be written
#SBATCH -e result_%x_%j.err      # File to which STDERR will be written
#SBATCH --mail-type=ALL
#SBATCH --mail-user=blopezp@eafit.edu.co

module load bpp/4.4.1_Intel_oneAPI-2022_update-1

bpp --cfile A10_All_L-D.ctl
```

Resources

- https://github.com/bpp/bpp/

Authors

- Bryan López Parra <blopezp@eafit.edu.co>

3.3.12 BWA

Description

BWA is a software package for mapping low-divergent sequences against a large reference genome, such as the human genome. It consists of three algorithms: BWA-backtrack, BWA-SW and BWA-MEM. The first algorithm is designed for Illumina sequence reads up to 100bp, while the rest two for longer sequences ranged from 70bp to 1Mbp. BWA-MEM and BWA-SW share similar features such as long-read support and split alignment, but BWA-MEM, which is the latest, is generally recommended for high-quality queries as it is faster and more accurate. BWA-MEM also has better performance than BWA-backtrack for 70-100bp Illumina reads.s.

BWA 0.7.15

Table of contents

– *Author*

Basic information

- **Installation date:** 27/02/2017
- **URL:** http://bio-bwa.sourceforge.net/
- **Apolo version:** Apolo II
- **License:** GPL

Dependencies

- Intel Parallel Studio XE Cluster Edition 2017 Update 1

Installation

After solving the previously mentioned dependencies, you can proceed with the installation of **BWA.**

1. Download the latest version of the software (Source code - bz2) (https://sourceforge.net/projects/bio-bwa/files/):

```
cd /home/$USER/apps/bwa/src
wget https://downloads.sourceforge.net/project/bio-bwa/bwa-0.7.15.tar.bz2?r=https%3A
↪%2F%2Fsourceforge.net%2Fprojects%2Fbio-bwa%2Ffiles%2F&ts=1488235843&use_mirror=ufpr
tar xf bwa-0.7.15.tar.bz2
```

2. Compilation (Makefile), do the following steps:

```
cd bwa-0.7.15
module load intel/2017_update-1
emacs Makefile
...
CC=                     icc
...
make 2>&1 | tee bwa-make.log
```

3. After compiling **BWA**, we continue with the following steps:

```
sudo mkdir -p /share/apps/bwa/0.7.15/intel/2017_update-1/bin
sudo cp bwa /share/apps/bwa/0.7.15/intel/2017_update-1/bin
```

Module

```
#%Module1.0#####################################################################
##
## module bwa/0.7.15_intel-2017_update-1
##
## /share/apps/modules/bwa/0.7.15_intel-2017_update-1     Written by Mateo Gomez-
↪Zuluaga
##
```

(continues on next page)

(continued from previous page)

```
proc ModulesHelp { } {
    puts stderr "\tbwa/0.7.15_intel-2017_update-1 - sets the Environment for Bwa in \
    \n\tthe share directory /share/apps/bwa/0.7.15/intel/2017_update-1\n"
}

module-whatis "\n\n\tSets the environment for using Bwa 0.7.15 \
               \n\tbuilded with Intel Parallel Studio XE 2017\n"

# for Tcl script use only
set       topdir     /share/apps/bwa/0.7.15/intel/2017_update-1
set       version    0.7.15
set       sys        x86_64-redhat-linux

module load intel/2017_update-1

prepend-path PATH    $topdir/bin
```

Usage mode

Load the necessary environment through the **module:**

```
module load bwa/0.7.15_intel-2017_update-1
```

TO-DO

References

- Makefile (inside bz2)

Author

- Mateo Gómez Zuluaga

3.3.13 CDO

Description

The Climate Data Operators (CDO) software is a collection of many operators for standard processing of climate and forecast model data. The operators include simple statistical and arithmetic functions, data selection and subsampling tools, and spatial interpolation. CDO was developed to have the same set of processing functions for GRIB [GRIB] and NetCDF [NetCDF] datasets in one package.

For additional information you can open those links:

CDO Official page , CDO wiki , Download

CDO 1.9.2

- **Installation date:** 22/02/2017

- **URL:** https://code.mpimet.mpg.de/projects/cdo/
- **Apolo version:** Apolo II
- **License:** GNU v2

Table of Contents

Dependencies

- STD C++11
- GCC/5.4.0

Dependencies not mandatory

- NetCDF
- Hdf5
- Hdf5 szip
- proj.4
- ECMWF ecCodes
- Magic

Installation

The installation specified here does not include any of the non-mandatory dependencies previously listed.

1. We obtain the source of **CDO (Climate Data Operations)** from the official website

```
wget https://code.mpimet.mpg.de/attachments/download/16035/cdo-1.9.2.tar.gz
```

2. Unzip, open the package and enter the directory

```
mkdir build
mkdir -p /share/apps/cdo/1.9.2/
```

3. We carry out the compilation process

```
cd build
../configure --prefix=/share/apps/cdo/1.9.2 --build=x86_64-redhat-linux-gnu
make
```

4. We confirm that the compilation has been correct and then we install

```
make check
make install
```

Module file

```
##%Module1.0#####################################################################
##
## module load cdo/1.9.2
##
## /share/apps/modules/cdo/1.9.2
## Written by Juan David Arcila Moreno
##

proc ModulesHelp {} {
    global version modroot
    puts stderr "Sets the environment for using cdo 1.9.2\
                        \nin the shared directory \
                        \n/share/apps/cdo/1.9.2\
                        \nbuilded with gcc-5.4.0"
}

module-whatis "(Name________) cdo"
module-whatis "(Version_____) 1.9.2"
module-whatis "(Compilers___) gcc-5.4.0"
module-whatis "(System______) x86_64-redhat-linux"
module-whatis "(Libraries___) "

# for Tcl script use only
set         topdir        /share/apps/cdo/1.9.2
set         version       1.9.2
set         sys           x86_64-redhat-linux

conflict cdo


prepend-path        PATH                    $topdir/bin
```

Usage mode

```
module load cdo/1.9.2
cdo -h [operator]
```

References

CDO Official page

Author

- Juan David Arcila-Moreno

3.3.14 CHARMM

(Chemistry at HARvard Macromolecular Mechanics)

A molecular simulation program with broad application to many-particle systems with a comprehensive set of energy functions, a variety of enhanced sampling methods, and support for multi-scale techniques including QM/MM, MM/CG, and a range of implicit solvent models.

- CHARMM primarily targets biological systems including peptides, proteins, prosthetic groups, small molecule ligands, nucleic acids, lipids, and carbohydrates, as they occur in solution, crystals, and membrane environments. CHARMM also finds broad applications for inorganic materials with applications in materials design.
- CHARMM contains a comprehensive set of analysis and model builiding tools.
- CHARMM achieves high performance on a variety of platforms including parallel clusters and GPUs and can be obtained here.
- CHARMM is actively maintained by a large group of developers led by Martin Karplus.
- charmm, with all of the functionality of CHARMM except its performance enhancements, is distributed at no cost to academic users. It can be downloaded directly here.

CHARMM c41b2

- **Installation date:** 5/06/2017
- **URL:** https://www.charmm.org/charmm
- **Apolo version:** Apolo II
- **License:** Payment (Special Contract)

Table of contents

Dependencies

Apolo

- MPICH 3.2 (This must be compiled without using the PMI interface, as there is a bug with SLURM which makes it impossible to run distributed jobs)
- GCC-5.4.0
- Source code GAMESS 2016 R1
- Cmake >= 3.7.1

Cronos

- Gnu GCC >= 5.5.0
- Mpich2 >= 3.2.1
- FFTW3 >= 3.3.7
- CMake >= 3.10.2
- Source code GAMESS 2016 R1 (gamess-current.tar.gz)

Installation

Apolo

After resolving to load the previously mentioned dependencies, you can proceed with the installation of CHARMM-GAMESS.

1. Prepare the environment before compilation:

```
$ cd /apps/charmm/src/charmm-gammess
$ tar xf c41b2.tar.gz
$ cd charmm
$ wget http://ala.cmm.ki.si/gamess/20160818.1/gamess-20160818.1-charmm.tar.gz
$ tar xf gamess-20160818.1-charmm.tar.gz
$ rm gamess-20160818.1-charmm.tar.gz
$ mkdir source/gamint/gamess
$ cp ~/apps/gamess/src/gamess/source/* source/gamint/gamess/
$ rm source/gamint/gamess/vector.src
$ rm source/gamint/gamess/ga.src
$ module unload slurm/16.05.6
$ module load mpich2/3.2_gcc-5.4.0 cmake/3.7.1
$ patch -p1 < gamess/gamess2016-for-charmm.patch
$ patch -p1 < gamess/c41b2-for-gamess.patch
$ chmod 755 tool/gmscomp
```

For the moment ignore this step.

We comment on the following line in the **tool/gmscomp** file:

```
$ emacs /home/mgomezzul/apps/charmm/src/charmm-gammess/charmm/tool/gmscomp
$ ...
$ #set EXTRAOPT="$EXTRAOPT -w -fno-aggressive-loop-optimizations -fno-sanitize=all"
$ ...
```

2. After preparing the compilation environment we proceed with the compilation:

```
$ mkdir -p ~/charmm/build
$ cd ~/charmm/build
$ ~/apps/charmm/src/charmm-gammess/charmm/configure --gamess
$ make -j 8
```

3. At the end of the execution of the **make -j 8**, several errors related to functions that are not defined will result, this happens because the CHARMM and GAMESS compilation methods are different, making cmake unable to produce the dependencies for the parallel compilation of the program. This is why it is necessary to apply a new patch and recompile again.

```
$ cd ~/apps/charmm/src/charmm-gammess/charmm
$ patch -p1 < gamess/c41b2-phase2.patch
$ cd ~/charmm/build
$ make -j 8
```

4. After finishing with the last parallel make, the **charmm** binary should have been produced, we can try it as follows:

```
$ ./charmm
...
1
                Chemistry at HARvard Macromolecular Mechanics
        (CHARMM) - Developmental Version 41b2   February 15, 2017
                           Revision unknown
    Copyright(c) 1984-2014  President and Fellows of Harvard College
                           All Rights Reserved
  Current operating system: Linux-2.6.32-504.16.2.el6.x86_64(x86_64)@deb
              Created on  6/6/17 at  9:36:54 by user: mgomezzul

         Maximum number of ATOMS:    360720, and RESidues:      120240
...
```

Cronos

After resolving to load the previously mentioned dependencies, you can proceed with the installation of **CHARMM-GAMESS.**

1. Prepare the compilation environment:

```
$ cd /home/mgomezzul/apps/charmm-gammes/src/gcc-5.5.0_mpich2-3.2.1
$ tar xf c41b2.tar.gz
$ cd charmm
We download the patch that allows you to compile charmm with support for gamess
$ wget http://ala.cmm.ki.si/gamess/20160818.1/gamess-20160818.1-charmm.tar.gz
$ tar xf gamess-20160818.1-charmm.tar.gz
$ rm gamess-20160818.1-charmm.tar.gz
$ mkdir source/gamint/gamess
$ cp  ~/apps/gammess/src/gamess/source/* source/gamint/gamess/
```

(continues on next page)

(continued from previous page)

```
$ rm source/gamint/gamess/vector.src
$ rm source/gamint/gamess/ga.src
$ module purge
$ module load fftw/3.3.7_gcc-5.5.0_mpich2-3.2.1 cmake/3.10.2
$ patch -p1 < gamess/gamess2016-for-charmm.patch
$ patch -p1 < gamess/c41b2-for-gamess.patch
$ chmod 755 tool/gmscomp
```

2. The following line is commented in the **tool/gmscomp** file:

```
$ emacs /home/mgomezzul/apps/charmm-gammes/src/gcc-5.5.0_mpich2-3.2.1/charmm/tool/
↪gmscomp
$ ...
$ #set EXTRAOPT="$EXTRAOPT -w -fno-aggressive-loop-optimizations -fno-sanitize=all"
$ ...
```

3. After preparing the compilation environment we proceed with it:

```
$ mkdir -p ~/apps/charmm-gammes/build
$ cd ~/apps/charmm-gammes/build
$ FFTWDIR=/share/apps/fftw/3.3.7/gcc-5.5.0_mpich2-3.2.1 FFTW_HOME=/share/apps/fftw/3.
↪3.7/gcc-5.5.0_mpich2-3.2.1 ~/apps/charmm-gammes/src/gcc-5.5.0_mpich2-3.2.1/charmm/
↪configure --prefix=/share/apps/charmm/c41b2/gcc-5.5.0_mpich2-3.2.1 --gamess --with-
↪gcc
$ make -j 16 | tee make-charmm.log
```

4. At the end of the execution of the **make -j 16**, several errors related to functions that are not defined will result, this happens because the CHARMM and GAMESS compilation methods are different, making cmake unable to produce the dependencies for the parallel compilation of the program. This is why it is necessary to apply a new patch and recompile again.

```
$ cd /home/$USER/apps/charmm-gammes/src/gcc-5.5.0_mpich2-3.2.1
$ patch -p1 < gamess/c41b2-phase2.patch
$ cd /home/$USER/apps/charmm-gammes/build
$ make -j 16 2>&1 | tee make-charmm.log
$ sudo mkdir -p /share/apps/charmm/c41b2/gcc-5.5.0_mpich2-3.2.1
$ sudo chown -R $USER.apolo /share/apps/charmm/c41b2/gcc-5.5.0_mpich2-3.2.1
$ sudo make install 2>&1 | tee make-charmm.log
$ sudo chown -R root.root /share/apps/charmm/c41b2/gcc-5.5.0_mpich2-3.2.1
```

5. After finishing with the last parallel make, the charmm binary should have been produced, we can try it as follows:

```
$ ./charmm
...
1
               Chemistry at HARvard Macromolecular Mechanics
         (CHARMM) - Developmental Version 41b2   February 15, 2017
                              Revision unknown
    Copyright(c) 1984-2014  President and Fellows of Harvard College
                            All Rights Reserved
  Current operating system: Linux-2.6.32-504.16.2.el6.x86_64(x86_64)@deb
               Created on  6/6/17 at  9:36:54 by user: mgomezzul

          aximum number of ATOMS:    360720, and RESidues:      120240
...
```

Module

```
##%Module1.0#####################################################################
##
## module load charmm/c41b2
##
## /share/apps/modules/charmm/c41b2
## Written by Mateo Gomez-Zuluaga
##

proc ModulesHelp {} {
    global version modroot
    puts stderr "Sets the environment for using CHARMM c41b2\
        \nin the shared directory\
        \n/share/apps/charmm/c41b2\
        \nbuilded with GAMESS 2016 R1, MPICH2 3.2 and GCC-5.4.0"
}

module-whatis "(Name_________) charmm"
module-whatis "(Version______) c41b2"
module-whatis "(Compilers____) mpich2-3.2_gcc-5.4.0"
module-whatis "(System_______) x86_64-redhat-linux"
module-whatis "(Libraries____) "

# for Tcl script use only
set         topdir        /share/apps/charmm/c41b2
set         version       c41b2
set         sys           x86_64-redhat-linux

conflict charmm

module load mpich2/3.2_gcc-5.4.0


prepend-path        PATH            $topdir/bin
```

Mode of Use

1. Have available an example to run charmm-gamess, in our case the example is available in **/home/$USER/test/charm/ilans**

```
cd /home/$USER/test/charmm-gamess/test_ilans
```

SLURM Template

```
#!/bin/sh

#!/bin/sh

#SBATCH --partition=longjobs
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=16
#SBATCH --time=250:00:00
```

(continues on next page)

(continued from previous page)

```
#SBATCH --job-name=mdrun-npt
#SBATCH -o result_%N_%j.out
#SBATCH -e result_%N_%j.err

# Don't share environment variables
export SBATCH_EXPORT=NONE
export OMP_NUM_THREADS=1

module load charmm/c41b2_gcc-5.5.0_mpich2-3.2.1

srun --mpi=pmi2 charmm < min1.inp  > pot.out
```

References

- https://www.charmm.org/charmm/documentation/by-version/c41b1/
- http://ala.cmm.ki.si/gamess/20160818.1/Compile-CHARMM-with-GAMESS.txt
- https://www.charmmtutorial.org/index.php/Installation
- https://github.com/kgullikson88/Telluric-Fitter/issues/5

Author

- Mateo Gómez Zuluaga

3.3.15 Clustalw

[1] Clustal W is a general purpose multiple alignment program for DNA or proteins.

Clustalw 2.1

Table of Contents

Basic information

- **Official Website:** http://www.clustal.org/clustal2/
- **Downloads page:** http://www.clustal.org/download/current/

[1] http://www.clustal.org/download/clustalw_help.txt

- **Installed on:** APOLO II

Tested on (Requirements)

- **OS base:** CentOS (x86_64) $\geq$ 6
- **Compiler** Intel 19.0.4

Installation

1. First of all, we need to load the following module for the compilation

```
$ module load intel/19.0.4
```

2. After that download the tar.gz file and unpack it

```
$ wget http://www.clustal.org/download/current/clustalw-2.1.tar.gz
$ tar -xzvf clustalw-2.1.tar.gz
$ cd clustalw-2.1
```

3. Then we can continue with the installation

```
$ mkdir $HOME/clustalw
$ ./configure --prefix=$HOME/clustalw --exec-prefix=$HOME/clustalw CFLAGS=
↪"-xHost -O3" CPPFLAGS="-xHost -O3" CXXFLAGS="-xHost -O3"
$ make -j
$ make install
```

4. If the installation was successful then you should:

```
$ cd $HOME/clustalw/bin/
$ ./clustalw2
```

Authors

- Laura Sanchez Cordoba
- Samuel Palacios Bernate

3.3.16 crystal

Description

A powerful and scalable computational tool for solid state chemistry and physics. CRYSTAL is a general-purpose program for the study of periodic systems. The code may be used to perform consistent studies of the physical and chemical properties of crystalline solids, surfaces, polymers, nanotubes and molecules. The CRYSTAL program computes the electronic structure, structural features, vibrational, magnetic, dielectric and elastic properties of periodic systems, density functional or various hybrid approximations.

The Bloch functions of the periodic systems are expanded as linear combinations of atom centred Gaussian functions. Powerful screening techniques are used to exploit real space locality and space symmetry is fully exploited.

A unique feature is the extensive exploitation of symmetry to achieve computational efficiency: 230 space groups, 80 two-sided plane groups, 99 rod groups and helical symmetry, 32 crystallographic point groups and molecular point group symmetry. The Bloch functions of the periodic systems are expanded as linear combinations of atom centred Gaussian functions. Powerful screening techniques are used to exploit real space locality. Restricted and unrestricted calculations can be performed with all-electron and valence-only basis sets with effective core pseudo-potentials.

Extensions:

- **MPP** - An enhanced massive parallel version of the code to allow users to reach an improved scaling in parallel executing on supercomputing resources.
- **CRYSCOR** - local second order Møller-Plesset Perturbation Theory - LMP2 for 1D-, 2D- and 3D-periodic non-conducting systems.

Crystal 1.0.4

- **Installation date:** 13/02/2017
- **URL:** http://www.crystalsolutions.eu/crystal.html
- **Apolo version:** Apolo II
- **License:** Academic license

Table of Contents

Dependencies

-Intel Parallel Studio XE 2017 Update 1

Installation

1. First download the tar from the main page https://www.crystalsolutions.eu/order.html?uid=2156&oid=649

```
mkdir CRYSTAL14
cd CRYSTAL14
tar -zxvf crystal14_v1_0_4_Linux-ifort_emt64_Pdistrib.tar.gz
cd build/Xmakes
```

1. Edit the folowwing lines in Linux-ifort_XE_impi_emt64.inc

```
cp Linux-ifort_XE_openmpi_emt64.inc Linux-ifort_XE_impi_emt64.inc
emacs Linux-ifort_XE_impi_emt64.inc
...
MPIBIN  = /share/apps/intel/ps_xe/2017_update-1/compilers_and_libraries_2017.1.132/
↪linux/mpi/intel64/bin
F90     = $(MPIBIN)/mpiifort
...
PLD     = $(MPIBIN)/mpiifort
F90FLAGS = -O3 -align -static-intel -cxxlib
...
MKL=/export/apps/intel/ps_xe/2017_update-1/compilers_and_libraries_2017.1.132/linux/
↪mkl/lib/intel64_lin
```

1. Edit the makefile

```
cd ..
emacs Makefile
...
ARCH = Linux-ifort_XE_impi_emt64
VERSION = v1.0.4
```

1. Compilation process

```
module load impi/2017_update-1
make all
```

Module

```
##%Module1.0#####################################################################
##
## modules 1.0.4_intel_impi-2017_update-1
##
## /share/apps/modules/crystal14/1.0.4_intel_impi-2017_update-1  Written by Mateo␣
↪Gomez-Zuluaga
##

proc ModulesHelp { } {
    puts stderr "\tcrystal14/v$version - sets the Environment for crystal14 in \
    \n\tthe share directory /share/apps/crystal14/bin/Linux-ifort_XE_impi_emt64/v1.0.
↪4\n"
}

module-whatis "\n\n\tSets the environment for using crystal14 v1.0.4 builded with \
          \n\tIntel Parallel Studio XE 2017 Update 1 (ifort and mpiifort)\n"

# for Tcl script use only
set     topdir          /share/apps/crystal14
set     version         1.0.4
set     sys          x86_64-redhat-linux

# Module use
set         user         [exec bash -c "echo \$USER"]
set         CRY14_ROOT   $topdir
set         CRY14_BIN    bin
```

(continues on next page)

(continued from previous page)

```
set          CRY14_ARCH   Linux-ifort_XE_impi_emt64
set          VERSION      v1.0.4
set          CRY14_EXEDIR $CRY14_ROOT/$CRY14_BIN/$CRY14_ARCH
set          CRY14_UTILS  $CRY14_ROOT/utils14


conflict crystal

module load impi/2017_update-1

setenv       CRY14_ROOT   $topdir
setenv       CRY14_BIN    bin
setenv       CRY14_ARCH   Linux-ifort_XE_impi_emt64
setenv       VERSION      v1.0.4
setenv       CRY14_SCRDIR /scratch-local/$user/crystal14
setenv       CRY14_EXEDIR $CRY14_ROOT/$CRY14_BIN/$CRY14_ARCH
setenv       CRY14_UTILS  $CRY14_ROOT/utils14
setenv       CRY14_TEST   $CRY14_ROOT/test_cases/inputs

prepend-path PATH    $CRY14_EXEDIR/$VERSION
prepend-path PATH    $CRY14_UTILS
```

Usage mode

```
rocks run host "hostname; mkdir -p /scratch-local/mmarins/crystal14; chown mmarins.
↪gema -R /scratch-local/mmarins/crystal14"
module load crystal14/1.0.4_intel_impi-2017_update-1
mkdir example_crystal
cd example_crystal
```

Slurm template

```
#!/bin/sh
#SBATCH --partition=longjobs
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=32
#SBATCH --time=2:00:00
#SBATCH --job-name=crystal14
#SBATCH -o result_%N_%j.out
#SBATCH -e result_%N_%j.err
#SBATCH --mail-type=END,FAIL
#SBATCH --mail-user=mmarins@eafit.edu.co

# Don't share environment variables
export SBATCH_EXPORT=NONE
export OMP_NUM_THREADS=1

module load crystal14/1.0.4_intel_impi-2017_update-1

# Store machine's names in nodes.par and machines.LINUX files
NODES="nodes.par"
MACHINES="machines.LINUX"
```

(continues on next page)

(continued from previous page)

```
srun hostname | sort > $NODES
srun hostname | sort > $MACHINES

INPUT_FILE='m_3a'

# Execute the Pcrystal
runmpi14  $SLURM_NTASKS $INPUT_FILE

# Removes the nodes.par and machines.LINUX files
rm nodes.par
rm machines.LINUX

# Delete temporal files
TMP_DIRS=`srun hostname | sort -u`

for i in ${TMP_DIRS[@]}; do
    rocks run host ${i} "rm -rf `cat temp_folder`"
done
```

References

- **Documents on the zip**
  - howtoinstall.txt
  - howtoinstall_from_objects.txt

Author

- Mateo Gómez Zuluaga

Crystal17

Description

CRYSTAL is a general-purpose program for the study of periodic systems. The code may be used to perform consistent studies of the physical and chemical properties of crystalline solids, surfaces, polymers, nanotubes and molecules. The CRYSTAL program computes the electronic structure, structural features, vibrational, magnetic, dielectric (linear and non-linear electric susceptibilities up to forth-order), elastic, piezoelectric, photoelastic, thermodynamic (Quasi-Harmonic Approximation) and transport properties of periodic systems, within Hartree Fock, density functional or various hybrid approximations (global and range-separated hybrids).

The Bloch functions of the periodic systems are expanded as linear combinations of atom centred Gaussian functions. Powerful screening techniques are used to exploit real space locality. Restricted and unrestricted calculations can be performed with all-electron and valence-only basis sets with effective core pseudo-potentials. A unique feature is the extensive exploitation of symmetry to achieve computational efficiency: 230 space groups, 80 two-sided plane groups, 99 rod groups and helical symmetry, 32 crystallographic point groups and molecular point group symmetry.

Extensions

- **MPP** - An enhanced massive parallel version of the code to allow users to reach an improved scaling in parallel executing on supercomputing resources.
- **CRYSCOR** - (for CRYSTAL14/09 only) Local second order Møller-Plesset Perturbation Theory - LMP2 for 1D-, 2D- and 3D-periodic non-conducting systems.

Crystal17 1.0.2

- **Installation date:** 07/03/2022
- **URL:** http://www.crystalsolutions.eu/crystal.html
- **Apolo version:** Apolo II
- **License:** Academic license

Table of Contents

Dependencies

- Intel oneAPI 2022 Update 1

Installation

1. First download the tar from the main page http://www.crystalsolutions.eu/try-it.html

```
module load intel/2022_oneAPI-update1
mkdir CRYSTAL17
cd CRYSTAL17
tar -zxvf crystal17_v1_0_2_demo.tar.gz
```

2. It will generate some tarball files, uncompress the Linux one

```
tar -zxvf crystal17_v1_0_2_Linux-ifort17_emt64_demo.tar.gz
```

3. Copy the bin directorie to the Root directorie of crystal

```
sudo mkdir /share/apps/crystal17/1_0_2/Intel_oneAPI-2022_update-1
sudo cp -r bin /share/apps/crystal17/1_0_2/Intel_oneAPI-2022_update-1
```

Module

```
##%Module1.0#####################################################################
##
## modules 1.0.4_intel_impi-2017_update-1
##
## /share/apps/crystal17/1_0_2/Intel_oneAPI-2022_update-1  Written by Santiago Alzate␣
↪Cardona
##

proc ModulesHelp { } {
    puts stderr "\tcrystal17/v$version - sets the Environment for crystal17 in \
    \n\tthe share directory /share/apps/crystal17/1_0_2/Intel_oneAPI-2022_update-1\n"
}

module-whatis "\n\n\tSets the environment for using crystal17 v1.0.2 builded with \
          \n\tIntel_oneAPI-2022_update-1 (ifort and mpiifort)\n"

# for Tcl script use only
set     topdir           /share/apps/crystal17/1_0_2/Intel_oneAPI-2022_update-1
set     version          1.0.4
set     sys          x86_64-redhat-linux

# Module use
set         user         [exec bash -c "echo \$USER"]
set         CRY17_ROOT   $topdir
set         CRY17_BIN    bin
set         CRY17_ARCH   Linux-ifort14_XE_emt64/std
set         VERSION      v1.0.2
set         CRY17_EXEDIR $CRY17_ROOT/$CRY17_BIN/$CRY17_ARCH


conflict crystal

module load intel

setenv      CRY17_ROOT   $topdir
setenv      CRY17_BIN    bin
setenv      CRY17_ARCH   Linux-ifort14_XE_emt64/std
setenv      VERSION      v1.0.2
setenv      CRY17_EXEDIR $CRY17_ROOT/$CRY17_BIN/$CRY17_ARCH

prepend-path PATH    $CRY17_EXEDIR
```

Usage mode

```
cd example_crystal
module load module load crystal17/1_2_0_Intel_oneAPI-2022_update-1
crystal
```

References

- https://www.crystal.unito.it/documentation.php

- https://www.crystal.unito.it/Manuals/crystal17_P.pdf

Author

- Santiago Alzate Cardona

3.3.17 Curl

Curl[1] is used in command lines or scripts to transfer data. curl is also used in cars, television sets, routers, printers, audio equipment, mobile phones, tablets, settop boxes, media playersand is the Internet transfer engine for thousands of software applications in over ten billion installations.

Curl 7.82.0

Table of Contents

Basic information

- **Instalation date:** 03/03/2022
- **Official Website:** https://github.com/curl/curl
- **Supercomputer:** Apolo II

Installation

1. Load the necessary modules for compilation

```
$ module load gcc/11.2.0
```

2. Download the desired version of the software (Source code - tar.bz2)[1]

```
$ cd /home/blopezp
$ wget https://curl.se/download/curl-7.82.0.tar.bz2
$ tar -xvf curl-7.82.0.tar.bz2
```

3. After unzipping **Curl**, continue with the following steps for configuration and compilation:

[1] https://curl.se/
[1] https://curl.se/

```
$ cd curl-7.82.0

$ ./configure --prefix=/share/apps/curl/7.82.0/gcc-11.2.0 --disable-
↪static --with-openssl --enable-threaded-resolver

$ make -j 10 2>&1 | tee curl-make.log
$ make -j 10 check 2>&1 | tee curl-make-check.log
$ sudo mkdir -p /share/apps/curl/7.82.0
$ sudo make install 2>&1 | tee curl-make-install.log
```

Module

```
##%Module1.0#####################################################################
↪###
##
## module curl/7.82.0_gcc-11.2.0
##
## /share/apps/curl/7.82.0/gcc-11.2.0     Written by Bryan Lopez Parra
##

proc ModulesHelp { } {
    puts stderr "\tcurl/7.82.0 - sets the Environment for CURL in \
    \n\tthe share directory /share/apps/curl/7.82.0/gcc-11.2.0\n"
}

module-whatis "\n\n\tSets the environment for using CURL-7.82.0 \
               \n\tbuilded with gcc 11.2.0 version\n"

# for Tcl script use only
set       topdir     /share/apps/curl/7.82.0/gcc-11.2.0
set       version    7.82.0
set       sys        x86_64-redhat-linux

module load gcc/11.2.0

prepend-path PATH               $topdir/bin

prepend-path LD_LIBRARY_PATH    $topdir/lib
prepend-path LIBRARY_PATH       $topdir/lib
prepend-path LD_RUN_PATH        $topdir/lib

prepend-path C_INCLUDE_PATH     $topdir/include
prepend-path CXX_INCLUDE_PATH   $topdir/include
prepend-path CPLUS_INCLUDE_PATH $topdir/include

prepend-path PKG_CONFIG_PATH    $topdir/lib/pkgconfig
```

Mode of use

```
$ module load curl/7.82.0_gcc-11.2.0
```

References

Author

- Bryan López Parra <blopezp@eafit.edu.co>

3.3.18 Cufflinks

Description

Cufflinks assembles transcripts, estimates their abundances, and tests for differential expression and regulation in RNA-Seq samples. It accepts aligned RNA-Seq reads and assembles the alignments into a parsimonious set of transcripts. Cufflinks then estimates the relative abundances of these transcripts based on how many reads support each one, taking into account biases in library preparation protocols.

Cufflinks was originally developed as part of a collaborative effort between the Laboratory for Mathematical and Computational Biology, led by Lior Pachter at UC Berkeley, Steven Salzberg's computational genomics group at the Institute of Genetic Medicine at Johns Hopkins University, and Barbara Wold's lab at Caltech. The project is now maintained by Cole Trapnell's lab at the University of Washington.

Cufflinks is provided under the OSI-approved Boost License

Cufflinks 2.2.1

- **Installation date:** 28/02/2017
- **URL:** http://cole-trapnell-lab.github.io/cufflinks/
- **Apolo version:** Apolo II
- **License:** Boost Software License, Version 1.0

Table of Contents

Installation

These are the steps to install **Cufflinks:**

1. Download the latest version of the software (Binaries - bz2) (http://cole-trapnell-lab.github.io/cufflinks/install/):

```
cd /home/$USER/apps/cufflinks/src
wget http://cole-trapnell-lab.github.io/cufflinks/assets/downloads/cufflinks-2.2.1.
↪Linux_x86_64.tar.gz
tar xf cufflinks-2.2.1.Linux_x86_64.tar.gz
```

2. For installation, the following steps must be done:

..code-block:: bash

    cd cufflinks-2.2.1.Linux_x86_64 sudo mkdir -p /share/apps/cufflinks/2.2.1/bin sudo cp cuff* g* /share/apps/cufflinks/2.2.1/bin

Module

```
##%Module1.0#####################################################################
##
## module cufflinks/2.2.1
##
## /share/apps/modules/cufflinks/2.2.1     Written by Mateo Gomez-Zuluaga
##

proc ModulesHelp { } {
    puts stderr "\tcufflinks/2.2.1_intel-2017_update-1 - sets the Environment for␣
↪Cufflinks in \
    \n\tthe share directory /share/apps/cufflinks/2.2.1\n"
}

module-whatis "\n\n\tSets the environment for using Cufflinks 2.2.1 \n"

# for Tcl script use only
set       topdir     /share/apps/cufflinks/2.2.1
set       version    2.2.1
set       sys        x86_64-redhat-linux

repend-path PATH    $topdir/bin
```

Mode of use

Load the necessary environment through the **module:**

```
module load cufflinks/2.2.1
```

TO-DO

References

- https://github.com/cole-trapnell-lab/cufflinks

Author

- Mateo Gómez Zuluaga

3.3.19 delft3D

Description

Delft3D is a world leading 3D modeling suite to investigate hydrodynamics, sediment transport and morphology and water quality for fluvial, estuarine and coastal environments. As per 1 January 2011, the Delft3D flow (FLOW), morphology (MOR) and waves (WAVE) modules are available in open source.

The software is used and has proven his capabilities on many places around the world, like the Netherlands, USA, Hong Kong, Singapore, Australia, Venice, etc. The software is continuously improved and developed with innovating advanced modelling techniques as consequence of the research work of our institute and to stay world leading.

The FLOW module is the heart of Delft3D and is a multi-dimensional (2D or 3D) hydrodynamic (and transport) simulation programme which calculates non-steady flow and transport phenomena resulting from tidal and meteorological forcing on a curvilinear, boundary fitted grid or sperical coordinates. In 3D simulations, the vertical grid is defined following the so-called sigma coordinate approach or Z-layer approach. The MOR module computes sediment transport (both suspended and bed total load) and morphological changes for an arbitrary number of cohesive and non-cohesive fractions. Both currents and waves act as driving forces and a wide variety of transport formulae have been incorporated. For the suspended load this module connects to the 2D or 3D advection-diffusion solver of the FLOW module; density effects may be taken into account. An essential feature of the MOR module is the dynamic feedback with the FLOW and WAVE modules, which allow the flows and waves to adjust themselves to the local bathymetry and allows for simulations on any time scale from days (storm impact) to centuries (system dynamics). It can keep track of the bed composition to build up a stratigraphic record. The MOR module may be extended to include extensive features to simulate dredging and dumping scenarios.

For over 30 years Deltares has been in the forefront of these types of combined morphological simulation techniques.

r7792

- **Installation date:** 14/11/2017
- **URL:** https://oss.deltares.nl/web/delft3d
- **Apolo version:** Apolo II and Cronos
- **License:** GNU GENERAL PUBLIC LICENSE 3.0

Table of Contents

Dependencies

- Intel Parallel Studio XE 2017 Update 1 Cluster Editon >= 17.0.1

- mpich2 >= 3.2
- NetCDF-C >= 4.5.0
- NetCDF Fortran
- hdf5 >= 1.8.19
- zlib >= 1.2.11
- szip >= 2.1
- libtool >= 2.4.6
- autconf >= 2.69

Installation

Load the needed modules or add the to the path

1. Download the repository

```
$ cd /home/mgomezzul/apps/delft3d/intel
$ svn checkout https://svn.oss.deltares.nl/repos/delft3d/tags/8799 delft3d_8850
```

2. For installation, the following steps must be done:

```
$ cd delft3d_8850/src
$ ./build.sh -intel18 -64bit -c "--build=x86_64-redhat-linux"
```

Module

```
#%Module1.0#####################################################################
##
## module load delft3d/8799_intel-18.0.2
##
## /share/apps/modules/delft3d/8799_intel-18.0.2
##
## Written by Mateo Gómez-Zuluaga
##

proc ModulesHelp {} {
 global version modroot
 puts stderr "Sets the environment for using Delft3D 8799\
              \nin the shared directory /share/apps/delft3d/8799/intel-18.0.2\
              \nbuilded with Intel Compiler 18.0.2, MPICH2-3.2.1, NetCDF 4.6.1, \
              \nNetCDF 4.4.4 and HDF5-1.8.20."
}

module-whatis "(Name________) delft3d"
module-whatis "(Version_____) 8799"
module-whatis "(Compilers___) intel-18.0.2"
module-whatis "(System______) x86_64-redhat-linux"
module-whatis "(Libraries___) netcdf-4.6.1, netcdf-fortran-4.4.4, hdf5-1.8.20"
# for Tcl script use only
set         topdir        /share/apps/delft3d/8799/intel-18.0.2
set         version       8799
```

(continues on next page)

(continued from previous page)

```
set         sys           x86_64-redhat-linux
conflict delft3d
module load netcdf/4.6.1_intel-18.0.2
module load mpich2/3.2.1_intel-18.0.2
setenv              D3D_HOME                $topdir
setenv              ARCH                    lnx64
prepend-path        PATH                    $topdir/bin
prepend-path        LD_LIBRARY_PATH         $topdir/lib
prepend-path        LIBRARY_PATH            $topdir/lib
prepend-path        LD_RUN_PATH             $topdir/lib
prepend-path    LD_LIBRARY_PATH         $topdir/lnx64/dimr/bin
prepend-path    LIBRARY_PATH            $topdir/lnx64/dimr/bin
prepend-path    LD_RUN_PATH             $topdir/lnx64/dimr/bin
prepend-path        PATH                    $topdir/lnx64/flow2d3d/bin
prepend-path        LD_LIBRARY_PATH         $topdir/lnx64/flow2d3d/bin
prepend-path        LIBRARY_PATH            $topdir/lnx64/flow2d3d/bin
prepend-path        LD_RUN_PATH             $topdir/lnx64/flow2d3d/bin
prepend-path        PATH                    $topdir/lnx64/part/bin
prepend-path        LD_LIBRARY_PATH         $topdir/lnx64/part/bin
prepend-path        LIBRARY_PATH            $topdir/lnx64/part/bin
prepend-path        LD_RUN_PATH             $topdir/lnx64/part/bin
prepend-path        LD_LIBRARY_PATH         $topdir/lnx64/plugins/bin
prepend-path        LIBRARY_PATH            $topdir/lnx64/plugins/bin
prepend-path        LD_RUN_PATH             $topdir/lnx64/plugins/bin
prepend-path        PATH                    $topdir/lnx64/scripts
prepend-path        LD_LIBRARY_PATH         $topdir/lnx64/shared
prepend-path        LIBRARY_PATH            $topdir/lnx64/shared
prepend-path        LD_RUN_PATH             $topdir/lnx64/shared
prepend-path        PATH                    $topdir/lnx64/swan/bin
prepend-path        PATH                    $topdir/lnx64/swan/scripts
prepend-path        LD_LIBRARY_PATH         $topdir/lnx64/swan/bin
prepend-path        LIBRARY_PATH            $topdir/lnx64/swan/bin
prepend-path        LD_RUN_PATH             $topdir/lnx64/swan/bin
prepend-path        PATH                    $topdir/lnx64/waq/bin
prepend-path        LD_LIBRARY_PATH         $topdir/lnx64/waq/bin
prepend-path        LIBRARY_PATH            $topdir/lnx64/waq/bin
prepend-path        LD_RUN_PATH             $topdir/lnx64/waq/bin
prepend-path        PATH                    $topdir/lnx64/wave/bin
prepend-path        LD_LIBRARY_PATH         $topdir/lnx64/wave/bin
prepend-path        LIBRARY_PATH            $topdir/lnx64/wave/bin
prepend-path        LD_RUN_PATH             $topdir/lnx64/wave/bin
```

Use mode

```
#!/bin/bash

#SBATCH --partition=longjobs
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=16
#SBATCH --time=4:00:00
#SBATCH --job-name=delft3d_271
#SBATCH -o test_%N_%j.out      # File to which STDOUT will be written
#SBATCH -e test_%N_%j.err      # File to which STDERR will be written
```

(continues on next page)

(continued from previous page)

```
export SBATCH_EXPORT=NONE
export OMP_NUM_THREADS=16
export OMP_NUM_THREADS_SWAN=$SLURM_NTASKS
#export NHOSTS=$SLURM_JOB_NUM_NODES
#export NSLOTS=$SLURM_NTASKS

module load delft3d/8799_intel-18.0.2

# Specify the config file to be used here
argfile=config_d_hydro.xml
mdwfile=271.mdw

# Run in parallel Flow
mpirun -np 1 d_hydro.exe $argfile &

# Run in combination Wave
wave.exe $mdwfile 1
```

References

- http://oss.deltares.nl/web/delft3d/source-code
- http://oss.deltares.nl/web/delft3d/faq

Author

- Mateo Gómez Zuluaga

r7799

- **Installation date:** 29/05/2018
- **URL:** https://oss.deltares.nl/web/delft3d
- **Apolo version:** Apolo II and Cronos
- **License:** GNU GENERAL PUBLIC LICENSE 3.0

Table of Contents

Dependencies

- Intel Parallel Studio XE 2017 Update 1 Cluster Editon >= 17.0.1
- mpich2 >= 3.2
- NetCDF-C >= 4.5.0
- NetCDF Fortran
- hdf5 >= 1.8.19
- zlib >= 1.2.11
- szip >= 2.1
- libtool >= 2.4.6
- autconf >= 2.69

Installation

Load the needed modules or add the to the path

1. Download the repository

```
$ cd cd /home/mgomezzul/apps/delft3d/intel
$ svn checkout https://svn.oss.deltares.nl/repos/delft3d/tags/8799 delft3d_8850
```

2. For installation, the following steps must be done:

```
$ cd delft3d_8850/src
$ ./build.sh -intel18 -64bit -c "--build=x86_64-redhat-linux"
```

Module

```
    ##%Module1.0#####################################################################
##
## module load delft3d/8799_intel-18.0.2
##
## /share/apps/modules/delft3d/8799_intel-18.0.2
##
## Written by Mateo Gómez-Zuluaga
##

proc ModulesHelp {} {
     global version modroot
     puts stderr "Sets the environment for using Delft3D 8799\
          \nin the shared directory /share/apps/delft3d/8799/intel-18.0.2\
          \nbuilded with Intel Compiler 18.0.2, MPICH2-3.2.1, NetCDF 4.6.1, \
          \nNetCDF 4.4.4 and HDF5-1.8.20."
}

module-whatis "(Name________) delft3d"
module-whatis "(Version_____) 8799"
module-whatis "(Compilers___) intel-18.0.2"
module-whatis "(System______) x86_64-redhat-linux"
```

(continues on next page)

(continued from previous page)

```
module-whatis "(Libraries___) netcdf-4.6.1, netcdf-fortran-4.4.4, hdf5-1.8.20"

# for Tcl script use only
set         topdir        /share/apps/delft3d/8799/intel-18.0.2
set         version       8799
set         sys           x86_64-redhat-linux

conflict delft3d
module load netcdf/4.6.1_intel-18.0.2
module load mpich2/3.2.1_intel-18.0.2

setenv              D3D_HOME                $topdir
setenv              ARCH                    lnx64

prepend-path        PATH                    $topdir/bin

prepend-path        LD_LIBRARY_PATH         $topdir/lib
prepend-path        LIBRARY_PATH            $topdir/lib
prepend-path        LD_RUN_PATH             $topdir/lib

prepend-path    LD_LIBRARY_PATH         $topdir/lnx64/dimr/bin
prepend-path    LIBRARY_PATH            $topdir/lnx64/dimr/bin
prepend-path    LD_RUN_PATH             $topdir/lnx64/dimr/bin

prepend-path        PATH                    $topdir/lnx64/flow2d3d/bin
prepend-path        LD_LIBRARY_PATH         $topdir/lnx64/flow2d3d/bin
prepend-path        LIBRARY_PATH            $topdir/lnx64/flow2d3d/bin
prepend-path        LD_RUN_PATH             $topdir/lnx64/flow2d3d/bin

prepend-path        PATH                    $topdir/lnx64/part/bin
prepend-path        LD_LIBRARY_PATH         $topdir/lnx64/part/bin
prepend-path        LIBRARY_PATH            $topdir/lnx64/part/bin
prepend-path        LD_RUN_PATH             $topdir/lnx64/part/bin

prepend-path        LD_LIBRARY_PATH         $topdir/lnx64/plugins/bin
prepend-path        LIBRARY_PATH            $topdir/lnx64/plugins/bin
prepend-path        LD_RUN_PATH             $topdir/lnx64/plugins/bin

prepend-path        PATH                    $topdir/lnx64/scripts

prepend-path        LD_LIBRARY_PATH         $topdir/lnx64/shared
prepend-path        LIBRARY_PATH            $topdir/lnx64/shared
prepend-path        LD_RUN_PATH             $topdir/lnx64/shared

prepend-path        PATH                    $topdir/lnx64/swan/bin
prepend-path        PATH                    $topdir/lnx64/swan/scripts
prepend-path        LD_LIBRARY_PATH         $topdir/lnx64/swan/bin
prepend-path        LIBRARY_PATH            $topdir/lnx64/swan/bin
prepend-path        LD_RUN_PATH             $topdir/lnx64/swan/bin

prepend-path        PATH                    $topdir/lnx64/waq/bin
prepend-path        LD_LIBRARY_PATH         $topdir/lnx64/waq/bin
prepend-path        LIBRARY_PATH            $topdir/lnx64/waq/bin
prepend-path        LD_RUN_PATH             $topdir/lnx64/waq/bin

prepend-path        PATH                    $topdir/lnx64/wave/bin
prepend-path        LD_LIBRARY_PATH         $topdir/lnx64/wave/bin
```

(continues on next page)

(continued from previous page)

```
prepend-path        LIBRARY_PATH            $topdir/lnx64/wave/bin
prepend-path        LD_RUN_PATH             $topdir/lnx64/wave/bin
```

Use mode

```
#!/bin/bash

#SBATCH --partition=longjobs
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=16
#SBATCH --time=4:00:00
#SBATCH --job-name=delft3d_271
#SBATCH -o test_%N_%j.out      # File to which STDOUT will be written
#SBATCH -e test_%N_%j.err      # File to which STDERR will be written

export SBATCH_EXPORT=NONE
export OMP_NUM_THREADS=16
export OMP_NUM_THREADS_SWAN=$SLURM_NTASKS
#export NHOSTS=$SLURM_JOB_NUM_NODES
#export NSLOTS=$SLURM_NTASKS

module load delft3d/8799_intel-18.0.2

# Specify the config file to be used here
argfile=config_d_hydro.xml
mdwfile=271.mdw

# Run in parallel Flow
mpirun -np 1 d_hydro.exe $argfile &

# Run in combination Wave
wave.exe $mdwfile 1
```

References

- http://oss.deltares.nl/web/delft3d/source-code
- http://oss.deltares.nl/web/delft3d/faq

Author

- Mateo Gómez Zuluaga

3.3.20 DL_POLY_Classic

Description

DL_POLY_Classic is a general-purpose (parallel and serial) molecular dynamics simulation package derived from the package formally known as DL_POLY_2 (version 20), originally written by W. Smith and TR Forester at Daresbury

Laboratory to support the "Computer Simulation of Condensed Phases "in CCP5. The terms under which this package is available are described in the DL_POLY_2 source code.

DL_POLY_Classic is different from (DL_POLY_3) DL_POLY_4 packages, which are available from Daresbury Laboratory under different license terms.

POLY

- **Installation date:** 19/09/2016
- **URL:** http://www.ccp5.ac.uk/DL_POLY_CLASSIC/
- **Apolo version:** Apolo I
- **License:** BSD licence

Table of Contents

Dependencies

- GNU GCC - gcc, gfortran >= 4.8.4
- MPI - openMPI (mpif90) >= 1.6.5

Installation

1. First download the tar from the main page from https://gitlab.com/DL_POLY_Classic/dl_poly, then:

```
$ cd /home/mgomezzul/apps/dl_poly_classic/dl_class
$ wget https://gitlab.com/DL_POLY_Classic/dl_poly/-/archive/RELEASE-1-10/dl_poly-
↪RELEASE-1-10.tar.gz
$ tar -zxvf dl_poly-RELEASE-1-10.tar.gz
```

2. configure the makefile

```
$ cd dl_class
$ cp build/MakePAR
$ cd source
$ module load openmpi/1.6.5_gcc-4.8.4
$ make gfortran
```

Note: This process was made with version 1.9, but these resources were removed so URLs refers to version 1.10, somethings could change in that version

Module

```
#%Module1.0#####################################################################
↪###
##
## modules dl_class/1.9_openmpi-1.6.5_gcc-4.8.4
##
## /share/apps/modules/dl_class/1.9_openmpi-1.6.5_gcc-4.8.4
## Written by Mateo Gomez-Zuluaga
##

proc ModulesHelp { } {
    puts stderr "\tdl_classic/1.9_openmpi-1.6.5_gcc-4.8.4 - sets the
    \n\tEnvironment for DL_POLY Classic in the share directory
    \n\t/share/apps/dl_class/1.9/openmpi-1.6.5/gcc-4.8.4\n"
}

module-whatis "\n\n\tSets the environment for using DL_POLY Classic 1.9 \
               \n\tbuilded with openMPI 1.6.5 and GCC 4.8.4 version\n"

# for Tcl script use only
set       topdir     /share/apps/dl_class/1.9/openmpi-1.6.5/gcc-4.8.4
set       version    1.9
set       sys        x86_64-redhat-linux

module load openmpi/1.6.5_gcc-4.8.4

prepend-path    PATH $topdir/bin
```

Use mode

```
#!/bin/sh
#SBATCH --partition=bigmem
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=2
#SBATCH --time=20:00
#SBATCH --job-name=gamess
#SBATCH -o result_%N_%j.out
#SBATCH -e result_%N_%j.err

# Don't share environment variables
export SBATCH_EXPORT=NONE
export OMP_NUM_THREADS=1

module load intel/2017_update-1 impi/2017_update-1 mkl/2017_update-1

# Execution line
rungms c2h6O2 00 $SLURM_NTASKS $SLURM_NTASKS_PER_NODE
```

Usage mode

```
$ module load dl_class/1.9_openmpi-1.6.5_gcc-4.8.4
```

References

- Mail from the people of the University of Cartagena
- Manual within the software package

Author

- Mateo Gómez Zuluaga

3.3.21 EDGE-pro

EDGE-pro[1] (Estimated Degree of Gene Expression in PROkaryotes) is an efficient software system to estimate gene expression levels in prokaryotic genomes from RNA-seq data.

EDGE-pro uses **Bowtie2** for alignment and then estimates expression directly from the alignment results.

EDGE-pro is distributed as a single package containing all the necessary files, including manual, source code[2] , executables and Perl scripts.

EDGE-pro - 1.3.1

Table of Contents

Basic information

- **Deploy date:** 15 December 2018
- **Official Website:** http://ccb.jhu.edu/software/EDGE-pro/
- **License:** GNU GPL v3
- **Installed on:** *Apolo II*, *Cronos*

[1] T. Magoc, D. Wood, and S.L. Salzberg. (2013, May 31). EDGE-pro Official Page. Retrieved 09:52, January 16, 2019 from http://ccb.jhu.edu/software/EDGE-pro/ .

[2] T. Magoc, D. Wood, and S.L. Salzberg. EDGE-pro: Estimated Degree of Gene Expression in Prokaryotic Genomes. Evolutionary Bioinformatics vol.9, pp.127-136, 2013

- **Dependencies:** Bowtie2
- **Available versions:** Intel Compiled

Installation

This entry covers the entire process performed in the installation and test of **EDGE-pro** on a cluster with the conditions described below. It is worth mentioning that this application does not meet the Linux Directory Hierarchy Standard, so we change the directory structure to make it work with `modules`, `Slurm`, and other subsystems.

As a side note, the performance of **EDGE-pro** depends directly on the **bowtie**'s performance, so we suggest making a different tuning compilation of it.

Contents

Tested on (Requirements)

- **OS base:** CentOS (x86_64) $\geq$ 6.6 (Rocks 6.2)
- **Compiler:** Intel Parallel Studio XE Cluster Edition $\geq$ 17.0.1

Build process

1. Get the current **EDGE-pro** version from the **official web page** and enter into the source directory.

```
wget http://ccb.jhu.edu/software/EDGE-pro/EDGE_pro_v1.3.1.tar.gz
tar -xvf EDGE_pro_v1.3.1.tar.gz
cd EDGE_pro_v1.3.1
```

2. You will notice that it just contains a simple source file. Build it. `make` will use the compiler you have loaded in your environment.

```
make
```

Note: You must load the necessary modules to build (i.e. `Intel Compiler`).

In Apolo II:

```
module load intel/2017_update-1
```

In Cronos:

```
module load intel/intel-18.0.2
```

---

3. Create the installation directories. Move the built executable and other necessary scripts. The main script of **EDGE_pro** is `edge.pl`, so is the only script we need to put in the `PATH`.

Apolo

```
mkdir -p /share/apps/edge_pro/1.3.1/intel-2017_update-1/bin
mkdir -p /share/apps/edge_pro/1.3.1/intel-2017_update-1/scripts
mv count additionalScripts /share/apps/edge_pro/1.3.1/intel-2017_update-1/
↪scripts
mv edge.pl /share/apps/edge_pro/1.3.1/intel-2017_update-1/bin
```

Cronos

```
mkdir -p /share/apps/edge_pro/1.3.1/intel-18.0.2/bin
mkdir -p /share/apps/edge_pro/1.3.1/intel-18.0.2/scripts
mv count additionalScripts /share/apps/edge_pro/1.3.1/intel-18.0.2/scripts
mv edge.pl /share/apps/edge_pro/1.3.1/intel-18.0.2/bin
```

4. Finally, we need to put the **bowtie** binaries in the `scripts` directory. Otherwise **EDGE-pro** will not find it, even if is load in the `PATH`. To do this we use symbolic links:

Apolo

```
cd /share/apps/edge_pro/1.3.1/intel-2017_update-1/scripts
ln -s /share/apps/bowtie2/2.3.4.1/2017_update-1/bin/bowtie2-align-l bowtie2-
↪align
ln -s /share/apps/bowtie2/2.3.4.1/2017_update-1/bin/bowtie2-build bowtie2-
↪build
ln -s /share/apps/bowtie2/2.3.4.1/2017_update-1/bin/bowtie2 bowtie
```

Cronos

```
cd /share/apps/edge_pro/1.3.1/intel-18.0.2/scripts
ln -s /share/apps/bowtie2/2.3.4.1/intel-18.0.2/bin/bowtie2-align-l bowtie2-
↪align
ln -s /share/apps/bowtie2/2.3.4.1/intel-18.0.2/bin/bowtie2-build bowtie2-
↪build
ln -s /share/apps/bowtie2/2.3.4.1/intel-18.0.2/bin/bowtie2 bowtie2
```

Modulefile

Apolo II

Listing 34: `Module file`

```
##Module1.0#####################################################################
##
## module load edge_pro/1.3.1_intel-2017_update-1
##
## /share/apps/modules/edge_pro/1.3.1_intel-2017_update-1
## Written by Juan David Arcila-Moreno
##

proc ModulesHelp {} {
     global version modroot
     puts stderr "Sets the environment for using edge_pro 1.3.1\
                  \nin the shared directory \
                  \n/share/apps/edge_pro/1.3.1/intel-2017_update-1\
                  \ncount script builded with intel-2017_update-1"
```

(continues on next page)

(continued from previous page)

```
}

module-whatis "(Name_________) edge_pro"
module-whatis "(Version______) 1.3.1"
module-whatis "(Compilers____) intel-2017_update-1"
module-whatis "(System_______) x86_64-redhat-linux"
module-whatis "(Libraries____) "

# for Tcl script use only
set         topdir        /share/apps/edge_pro/1.3.1/intel-2017_update-1
set         version       1.3.1
set         sys           x86_64-redhat-linux

conflict edge_pro
module load intel/2017_update-1
module load bowtie2/2.3.0_intel-2017_update-1


prepend-path        PATH                        $topdir/bin
```

Cronos

Listing 35: `Module file`

```
##%Module1.0#####################################################################
##
## module load edge_pro/1.3.1_intel-18.0.2
##
## /share/apps/modules/edge_pro/1.3.1_intel-18.0.2
## Written by Juan David Arcila-Moreno
##

proc ModulesHelp {} {
     global version modroot
     puts stderr "Sets the environment for using edge_pro 1.3.1\
                  \nin the shared directory \
                  \n/share/apps/edge_pro/1.3.1/intel-18.0.2\
                  \ncount binary builded with intel-18.0.2"
}

module-whatis "(Name_________) edge_pro"
module-whatis "(Version______) 1.3.1"
module-whatis "(Compilers____) intel-18.0.2"
module-whatis "(System_______) x86_64-redhat-linux"
module-whatis "(Libraries____) "

# for Tcl script use only
set         topdir        /share/apps/edge_pro/1.3.1/intel-18.0.2
set         version       1.3.1
set         sys           x86_64-redhat-linux

conflict edge_pro
module load intel/18.0.2
module load bowtie2/2.3.4.1_intel-18.0.2


prepend-path        PATH                        $topdir/bin
```

Usage

In the following example we are going to run a test included in the source code of **EDGE-pro**. Note that we are using one single compute-node. Bowtie will be using `OpenMP` with 16 and 32 threads, in Cronos and Apolo, respectively.

Apolo

Listing 36: `slurm_edge_pro_apolo.sh`

```
#!/bin/bash

#SBATCH --job-name=edge_pro_test
#SBATCH --output=edge_pro_%A_%a.out
#SBATCH --error=edge_pro_%A_%a.err
#SBATCH --time=2:00
#SBATCH --partition=longjobs
#SBATCH --cpus-per-task=32

######################
# Begin work section #
######################

# Load module in default version
module load edge_pro

# Export number of threads
OMP_NUM_THREADS=32

edge.pl -g Cjejuni.fa -p Cjejuni.ptt -r Cjejuni.rnt -u wild1.fastq -s /share/apps/
↪edge_pro/1.3.1/intel-2017_update-1/scripts -t 32
```

Cronos

Listing 37: `slurm_edge_pro_cronos.sh`

```
#!/bin/bash

#SBATCH --job-name=test_arrayjobs
#SBATCH --output=edge_pro_%A_%a.out
#SBATCH --error=edge_pro_%A_%a.err
#SBATCH --time=2:00
#SBATCH --partition=longjobs
#SBATCH --cpus-per-task=16

######################
# Begin work section #
######################

# Load module in default version
module load edge_pro

# Export number of threads
OMP_NUM_THREADS=16

edge.pl -g Cjejuni.fa -p Cjejuni.ptt -r Cjejuni.rnt -u wild1.fastq -s /share/apps/
↪edge_pro/1.3.1/intel-18.0.2/scripts -t 16
```

To run the previous example.

```
sbatch slurm_edge_pro_[apolo|cronos].sh
```

Note: As you can notice in the previous examples we are using the flag `-s`. This is because we must specify to **EDGE_pro** where is the directory that contains: the built executable, the binaries of `bowtie2` and additional scripts.

Note: In this example we are using the default version of **EDGE-pro** module. We recommend to specify the version. To use the version of this entry, pleace load the module as follow:

In Apolo II:

```
module load edge_pro/1.3.1_intel-2017_update-1
```

In Cronos:

```
module load edge_pro/1.3.1_intel-18.0.2
```

Authors

- Juan David Arcila-Moreno <jarcil13@eafit.edu.co>

References

3.3.22 elmer

Elmer is a finite element software for numerical solution of partial differential equations. Elmer is capable of handling any number of equations and is therefore ideally suited for the simulation of multiphysical problems. It includes models, for example, of structural mechanics, fluid dynamics, heat transfer and electromagnetics. Users can also write their own equations that can be dynamically linked with the main program.

Elmer consists of several parts. The most important ones are ElmerSolver, the finite element solver, ElmerGUI, the graphical user interface, and ElmerGrid, the mesh creation and manipulation tool. Also a visualization tool, ElmerPost, is included in the package but it is no longer developed.

Elmer software is licensed under GPL except for the ElmerSolver library which is licensed under LGPL license.

Elmer is mainly developed at CSC - IT Center for Science, Finland. However, there have been numerous contributions from other organizations and developers as well, and the project is open for new contributions.

8.2

- **Installation date:** 06/03/2017
- **URL:** https://www.csc.fi/web/elmer
- **Apolo version:** Apolo II
- **License:** LGPL license

Table of contents

Pre requirements

- Intel Parell Studio XE Cluster Edition 2017 - Update 1 (Compilers, MPI, Math Libraries)
- Hypre >= 2.10.1
- Cmake >= 3.7.1

Installation

1. Download the latest version of the software (Github) (https://github.com/ElmerCSC/elmerfem):

```
cd /home/mgomezzul/apps/elmer/src/intel
mkdir elmer
cd elmer
git clone git://www.github.com/ElmerCSC/elmerfem+
mkdir build
cd build
```

2. To continue with the installation we must continue with the following configuration and compilation steps using **cmake**:

```
sudo mkdir -p /share/apps/elmer/8.2/intel_impi/2017_update-1
sudo chown -R $USER.apolo /share/apps/elmer/8.2/intel_impi/2017_update-1
cmake .. -DCMAKE_INSTALL_PREFIX=/share/apps/elmer/8.2/intel_impi/2017_update-1 -DWITH_
↪MPI:BOOL=TRUE  -DWITH_ELMERGUI:BOOL=FALSE -DWITH_OpenMP:BOOL=TRUE -DCMAKE_BUILD_
↪TYPE=Release -DMPI_TEST_MAXPROC=16 -DWITH_ElmerIce:BOOL=TRUE -DWITH_Hypre:BOOL=TRUE␣
↪-DWITH_MKL:BOOL=TRUE  -DHypre_LIBRARIES=/share/apps/hypre/2.10.1/intel_impi/2017_
↪update-1/lib/libHYPRE.so  -DHypre_INCLUDE_DIR=/share/apps/hypre/2.10.1/intel_impi/
↪2017_update-1/include 2>&1 | tee elmer-cmake.log
make 2>&1 | tee elmer-make.log
make install 2>&1 | tee elmer-make-install.log
sudo chown -R root.root /share/apps/elmer/8.2/intel_impi/2017_update-1
```

Module

```
##%Module1.0#####################################################################
##
## modules load elmer/8.2_intel_impi_mkl-2017_update-1
##
## /share/apps/modules/elmer/8.2_intel_impi_mkl-2017_update-1  Written by Mateo Gomez-
↪Zulauga
##

proc ModulesHelp { } {2
        puts stderr "\tSets the environment for Elmer 8.2 in the following \
                    \n\tdirectory /share/apps/elmer/8.2/intel_impi/2017_update-1\n"
}

module-whatis "\n\n\tSets the environment for using Elmer 8.2 \
            \n\tbuilded with Intel Parallel Studio XE Cluster Edition 2017 Update 1\n"


# for Tcl script use only
set         topdir     /share/apps/elmer/8.2/intel_impi/2017_update-1/
set         version    8.2
set     sys        x86_64-redhat-linux

conflict elmer
module load hypre/2.10.1_intel_impi_2017_update-1

prepend-path PATH           $topdir/bin

prepend-path LD_LIBRARY_PATH    $topdir/lib
prepend-path LIBRARY_PATH       $topdir/lib
prepend-path LD_RUN_PATH        $topdir/lib

prepend-path C_INCLUDE_PATH     $topdir/include
prepend-path CXX_INCLUDE_PATH   $topdir/include
prepend-path CPLUS_INCLUDE_PATH $topdir/include

setenv           ELMER_HOME         $topdir
setenv           ELMER_LIB          $topdir/lib
```

Mode of Use

Load the necessary environment through the **module**:

```
module load elmer/8.2_intel_impi_mkl-2017_update-1
```

Slurm template

```
#!/bin/sh
#SBATCH --partition=longjobs
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=32
#SBATCH --time=1-00
#SBATCH --job-name=elmer_test
#SBATCH -o result_%N_%j.out
```

(continues on next page)

(continued from previous page)

```
#SBATCH -e result_%N_%j.err
#SBATCH --mail-type=END,FAIL
#SBATCH --mail-user=dtobone@eafit.edu.co


# Don't share environment variables
export SBATCH_EXPORT=NONE
export OMP_NUM_THREADS=1

module load elmer/8.2_intel_impi_mkl-2017_update-1

ElmerGrid 2 2 mesh -metis $SLURM_NTASKS 4
srun ElmerSolver_mpi
```

References

- https://www.csc.fi/web/elmer/sources-and-compilation
- http://www.elmerfem.org/elmerwiki/index.php?title=Compilation_of_Elmer_on_Linux_using_Cmake
- http://www.elmerfem.org/elmerwiki/index.php
- https://github.com/ElmerCSC/elmerfem/releases
- http://www.elmerfem.org/forum/viewtopic.php

Author

- Mateo Gómez Zuluaga

3.3.23 fastQC

A Quality Control application for FastQ files.

Most high throughput sequencers generate output in FastQ format. This format combines the base calls for the sequence which was generated with an encoded quality value for each base which says how confident the sequencer was that the base call generated was correct.

Before proceeding with the analysis of a sequence data set it is a good idea to do some basic quality control checks on the raw data to ensure that there are no hidden problems which might be more difficult to detect at a later stage.

FastQC is an application which takes a FastQ file and runs a series of tests on it to generate a comprehensive QC report. This will tell you if there is anything unusual about your sequence. Each test is flagged as a pass, warning or fail depending on how far it departs from what you'd expect from a normal large dataset with no significant biases. It's important to stress that warnings or even failures do not necessarily mean that there is a problem with your data, only that it is unusual. It is possible that the biological nature of your sample means that you would expect this particular bias in your results.

FastQC can be run either as an interactive graphical application which allows you to view results for multiple files in a single application. Alternatively you can run the program in a non interactive way (say as part of a pipeline) which will generate an HTML report for each file you process.

FastQC is a cross-platform application, written in java. In theory it should run on any platform which has a suitable java runtime environment. Having said that we've only tested in on Windows, MacOSX and Linux running the Oracle v1.6 to 1.8 JREs. Please let us know what happened if you try running it on other platforms / JREs. Please see the

detailed instructions in the INSTALL.txt document to tell you how to get a suitable java version to run FastQC on your system.

0.11.5

- **Installation date:** 01/03/2017
- **URL:** http://www.bioinformatics.babraham.ac.uk/projects/fastqc/
- **Apolo version:** Apolo II
- **License:** GNU GENERAL PUBLIC LICENSE, Version 3

Table of Contents

Installation

These are the steps to install **FastQC:**

1. Download the latest version of the software (Binaries - zip) (http://www.bioinformatics.babraham.ac.uk/projects/download.html#fastqc):

```
cd /home/$USER/apps/fastqc/src
wget http://www.bioinformatics.babraham.ac.uk/projects/fastqc/fastqc_v0.11.5.zip
unzip fastqc_v0.11.5.zip
```

2. For installation, the following steps must be done:

```
cd FastQC
sudo mkdir -p /share/apps/fastqc/0.11.5/bin
sudo mkdir -p /share/apps/fastqc/0.11.5/lib
sudo cp *.jar /share/apps/fastqc/0.11.5/lib/
sudo cp -r uk/ org/ net/ fastqc /share/apps/fastqc/0.11.5/bin/
```

Module

```
##%Module1.0#####################################################################
##
## module fastqc/0.11.5
##
## /share/apps/modules/fastqc/0.11.5     Written by Mateo Gomez-Zuluaga
##
```

(continues on next page)

(continued from previous page)

```
proc ModulesHelp { } {
    puts stderr "\tfastqc/0.11.5 - sets the Environment for FastQC in \
    \n\tthe share directory /share/apps/fastqc/0.11.5\n"
}

module-whatis "\n\n\tSets the environment for using FastQC 0.11.5 \
               \n\tprecompiled\n"

# for Tcl script use only
set       topdir     /share/apps/fastqc/0.11.5
set       version    0.11.5
set       sys        x86_64-redhat-linux

conflict fastqc

module load java/jdk-1.8.0_112

prepend-path PATH                   $topdir/bin

prepend-path CLASSPATH                      $topdir/lib/cisd-jhdf5.jar
prepend-path CLASSPATH                      $topdir/lib/jbzip2-0.9.jar
prepend-path CLASSPATH                      $topdir/lib/sam-1.103.jar
```

Mode of use

Load the necessary environment through the **module:**

```
module load fastqc/0.11.5
```

TO-DO

References

- INSTALL.txt (file inside .zip)

Author

- Mateo Gómez Zuluaga

3.3.24 gamess

Description

GAMESS is a program to work with ab initio groups of molecular quantum chemistry. Briefly GAMESS can compute Honda SCF functions within the range of RHF, ROHF, UHF, GVB, and MCSCF.

- http://www.msg.ameslab.gov/GAMESS/capabilities.html

GAMESS 2014

- **Installation date:** 02/09/2016
- **URL:** http://www.msg.ameslab.gov/GAMESS/
- **Apolo version:** Apolo I
- **License:** ACADEMIC LICENSE

Table of Contents

Dependencies

- FORTRAN compiler

Installation

1. First download the tar from the main page, then:

```
$ tar -zxvf gamess-current.tar.gz
$ cd gamess/2014-R1
```

1. Gamess configuration

```
$ ./config
```

1. The above command is interactive, so it will ask several questions in the following

```
GAMESS can compile on the following 32 bit or 64 bit machines? linux64
GAMESS directory? [/share/apps/gamess/2014-R1]
GAMESS build directory? [/share/apps/gamess/2014-R1]
Version? [00]
Please enter your choice of FORTRAN? ifort
Version? 15
Enter your choice of 'mkl' or 'atlas' or 'acml' or 'none'? mkl
MKL pathname? /share/apps/intel/15/mkl
MKL version (or 'skip')? skip
Communication library ('sockets' or 'mpi')? sockets (Esta elección obedece a que la␣
↪versión de MPI solo puede usarse si se cuenta con Infiniband)
Do you want to try LIBCCHEM? (yes/no)? no
```

then

```
$ cd ddi
$ ./compddi 2>&1 | tee compddi.log
$ mv ddikick.x ..
$ cd ..
$ ./compall 2>&1 | tee compall.log
$ ./lked gamess 00 2>&1 | tee lked.log
```

Module

```
#%Module1.0#####################################################################
 ##
 ## modules gamess/dec-5-2014-R1
 ##
 ## /share/apps/modules/gamess/dec-5-2014-R1  Written by Mateo Gomez-Zuluaga
 ##

 proc ModulesHelp { } {
     puts stderr "\tgamess/dec-5-2014-R1 - sets the Environment for gamess in \
     \n\tthe share directory /share/apps/gamess/dec-5-2014-R1/intel-15/mkl-15\n"
 }

 module-whatis "\n\n\tSets the environment for gamess assembler \
               \n\tbuilded with Intel 15 and MKL 15 version\n"

 # for Tcl script use only
 set       topdir     /share/apps/gamess/dec-5-2014-R1/intel-15/mkl-15
 set       version    1.2.10
 set       sys        x86_64-redhat-linux

 conflict gamess

 module load intel/15.1.10.380555
 module load mkl/15.1.10.380555

 prepend-path    PATH                        $topdir
```

References

- http://www.mothur.org/
- https://github.com/mothur/mothur
- https://github.com/mothur/mothur/releases
- http://www.mothur.org/wiki/Makefile_options
- http://www.mothur.org/wiki/Main_Page

Author

- Mateo Gómez Zuluaga

GAMESS 2016

- **Installation date:** 13/03/2017
- **URL:** http://www.msg.ameslab.gov/GAMESS/
- **Apolo version:** Apolo II
- **License:** ACADEMIC LICENSE

Table of Contents

Dependencies

- Intel Parallel Studio XE Cluster Edition 2017 (Update 1)
- Intel Fortran and C compilers
- Intel MKL
- Intel MPI

Installation

1. First download the tar from the main page, then:

```
$ tar -zxvf gamess-current.tar.gz
$ cd gamess/
```

1. Gamess configuration

```
$ ./config
```

1. The above command is interactive, so it will ask several questions in the following

```
Enter
linux64
/share/apps/gamess/2016-r1/intel_impi_mkl/2017_update-1
/share/apps/gamess/2016-r1/intel_impi_mkl/2017_update-1
00
ifort
17
Enter
```

(continues on next page)

(continued from previous page)

```
mkl
/share/apps/intel/ps_xe/2017_update-1/mkl
proceed
 Enter
 Enter
mpi
impi
/share/apps/intel/ps_xe/2017_update-1/impi/2017.1.132
no
```

Module

```
#%Module1.0#####################################################################
 ##
 ## module load 2016-r1_intel_mkl_impi-2017_update-1
 ##
 ## /share/apps/modules/gamess/2016-r1_intel_mkl_impi-2017_update-1
 ##
 ## Written by Mateo Gomez-Zuluaga
 ##

 proc ModulesHelp { } {
     puts stderr "\tgamess/2016-r1_intel_mkl_impi-2017_update-1 - sets the␣
↪Environment for GAMESS 2016-R1 in \
     \n\tthe shared directory /share/apps/gamess/2016-r1/intel_impi_mkl/2017_update-
↪1\n"
 }

 module-whatis "\n\n\tSets the environment for GAMESS 2016-R1 \
               \n\tbuilded with Intel Parallel Studio XE Cluster Edition \n"

 # for Tcl script use only
 set       topdir     /share/apps/gamess/2016-r1/intel_impi_mkl/2017_update-1
 set       version    2016-r1
 set       sys        x86_64-redhat-linux

 conflict gamess

 module load intel/2017_update-1
 module load mkl/2017_update-1
 module load impi/2017_update-1

 prepend-path    PATH                        $topdir
```

Use mode

```
#!/bin/sh
#SBATCH --partition=bigmem
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=2
#SBATCH --time=20:00
#SBATCH --job-name=gamess
```

(continues on next page)

(continued from previous page)

```
#SBATCH -o result_%N_%j.out
#SBATCH -e result_%N_%j.err

# Don't share environment variables
export SBATCH_EXPORT=NONE
export OMP_NUM_THREADS=1

module load intel/2017_update-1 impi/2017_update-1 mkl/2017_update-1

# Execution line
rungms c2h6O2 00 $SLURM_NTASKS $SLURM_NTASKS_PER_NODE
```

References

- https://software.intel.com/en-us/articles/building-gamess-with-intel-compilers-intel-mkl-and-intel-mpi-on-linux
- https://linuxcluster.wordpress.com/2010/05/18/building-the-gamess-with-intel%C2%AE-compilers-intel%C2%AE-mkl-and-openmpi-on-linux/
- https://www.webmo.net/support/gamess_linux.html
- http://myweb.liu.edu/~nmatsuna/gamess/tests

Author

- Mateo Gómez Zuluaga

3.3.25 garli

Description

GARLI, Genetic Algorithm for Rapid Likelihood Inference is a program for inferring phylogenetic trees. Using an approach similar to a classical genetic algorithm, it rapidly searches the space of evolutionary trees and model parameters to find the solution maximizing the likelihood score. It implements nucleotide, amino acid and codon-based models of sequence evolution, and runs on all platforms. The latest version adds support for partitioned models and morphology-like datatypes. It is written and maintained by Derrick Zwickl.

Garli 2.01

- **Installation date:** 06/02/2017
- **URL:** http://phylo.bio.ku.edu/slides/GarliDemo/garliExercise.Evolution2014.html
- **Apolo version:** Apolo II
- **License:** GNU GENERAL PUBLIC LICENSE Version 3

Table of Contents

Dependencies

- Ncl
- Intel compiler (C y C++)
- Intel MPI (C y C++)

Installation

1. First download the tar from the main page

```
wget https://storage.googleapis.com/google-code-archive-downloads/v2/code.google.com/
↪garli/garli-2.01.tar.gz
tar -zxvf garli-2.01.tar.gz
```

1. compilation config and editing the makefile

```
cd garli-2.01
module load ncl/2.1.18_intel-2017_update-1
module load impi/2017_update-1
./configure --prefix=/share/apps/garli/2.0.1/intel_impi/2017_update-1  --build=x86_64-
↪redhat-linux --enable-mpi --with-ncl=/share/apps/ncl/2.1.18/intel/2017_update-1 2>&
↪1 | tee garli-conf.log
make 2>&1 | tee garli-make.log
make install 2>&1 | tee garli-make-install.log
```

Module

```
##%Module1.0#####################################################################
##
## module garli/2.01_intel_impi-2017_update-1
##
## /share/apps/modules/garli/2.01_intel_impi-2017_update-1     Written by Mateo Gomez-
↪Zuluaga
##

proc ModulesHelp { } {
    puts stderr "\tgarli/2.01_intel_impi-2017_update-1 - sets the Environment for␣
↪Garli in \
    \n\tthe share directory /share/apps/garli/2.0.1/intel_impi/2017_update-1\n"
}
```

(continues on next page)

(continued from previous page)

```
module-whatis "\n\n\tSets the environment for using Garli 2.01 \
               \n\tbuilded with Intel Parallel Studio XE 2017\n"

# for Tcl script use only
set       topdir     /share/apps/garli/2.0.1/intel_impi/2017_update-1
set       version    2.0.1
set       sys        x86_64-redhat-linux

module load ncl/2.1.18_intel-2017_update-1
module load impi/2017_update-1

prepend-path PATH    $topdir/bin
```

Slurm template

```
#!/bin/bash
#SBATCH --partition=bigmem
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=24
#SBATCH --time=1:00:00
#SBATCH --job-name=garli_example
#SBATCH -o result_%N_%j.out
#SBATCH -e result_%N_%j.err

export SBATCH_EXPORT=NONE
export OMP_NUM_THREADS=1

module load garli/2.01_intel_impi-2017_update-1

srun Garli $SLURM_NTASKS garli.conf
```

References

- manual

Author

- Mateo Gómez Zuluaga

3.3.26 GATK4

GATK[1], pronounced "Gee Ay Tee Kay" (not "Gat-Kay"), stands for GenomeAnalysisToolkit. It is a collection of command-line tools for analyzing high-throughput sequencing data with a primary focus on variant discovery. The tools can be used individually or chained together into complete workflows. We provide end-to-end workflows, called GATK Best Practices, tailored for specific use cases.

[1] Broad Institute. (2019). GATK | Quick Start Guide. Retrieved 7 October 2019, from https://software.broadinstitute.org/gatk/documentation/quickstart

GATK4-4.1.0.0

Table of Contents

Basic Information

- **Deploy date:** 2 January 2019
- **Official Website:** https://software.broadinstitute.org/gatk/documentation/quickstart
- **License:** GATK4 License
- **Installed on:** *Apolo II*

Installation

This entry covers the entire process performed for the installation GATK4 on a cluster.

Tested on (Requirements)

- **OS base:** CentOS (x86_64) $\geq$ 6.6
- Java 1.8.
- Python $\geq$ 2.6.
- Git $\geq$ 1.8.2 with git-lfs.

Build process

This entry described the installation process of GATK4.

1. Clone the GATK4 repository.

```
git clone https://github.com/broadinstitute/gatk
```

2. Load the Java module.

```
module load java/jdk-1.8.0_112
```

3. Enter the directory and compile it.

Note: You can choose tha way to build:

- ./gradlew bundle or ./gradlew
  - This creates a zip archive in the build/ directory with a name like gatk-VERSION.zip containing a complete standalone GATK distribution, you can also run GATK commands directly from the root of your git clone after running this command.
- ./gradlew installDist
  - Does a fast build that only lets you run GATK tools from inside your git clone, and locally only (not on a cluster).
- ./gradlew installAll
  - Does a semi-fast build that only lets you run GATK tools from inside your git clone, but works both locally and on a cluster.
- ./gradlew localJar
  - Builds only the GATK jar used for running tools locally (not on a Spark cluster). The resulting jar will be in build/libs with a name like gatk-package-VERSION-local.jar, and can be used outside of your git clone.
- ./gradlew sparkJar
  - Builds only the GATK jar used for running tools on a Spark cluster (rather than locally).

```
cd gatk
./gradlew installDist
```

4. Move your compile files to the path where you want GATK4.

```
mv gatk/ /your/path/
```

5. Create the module.

Module Files

Apolo II

Listing 38: GATK4-4.1.0.0

```
##%Module1.0#####################################################################
##
## module load gatk4/4.1.0.0
##
## /share/apps/modules/gatk4/4.1.0.0
## Written by Manuela Carrasco Pinzon
##

proc ModulesHelp {} {
     global version modroot
     puts stderr "Sets the environment for using GATK4\
                  \nin the shared directory /share/apps/gatk4/4.1.0.0/"
}

module-whatis "(Name________) gatk4"
module-whatis "(Version_____) 4.1.0.0"
```

(continues on next page)

(continued from previous page)

```
module-whatis "(Compilers___) java 1.8"
module-whatis "(System_____) x86_64-redhat-linux"
module-whatis "(Libraries___) "

# for Tcl script use only
set         topdir        /share/apps/gatk4/4.1.0.0/
set         version       4.1.0.0
set         sys           x86_64-redhat-linux

conflict gatk
module load java/jdk-1.8.0_112


prepend-path    PATH                    $topdir
prepend-path    PATH                    $topdir/libexec

prepend-path    MANPATH                 $topdir/share/man
```

Usage

This section describes the method to submit jobs with the resource manager SLURM.

1. Load the necessary environment.

```
module load gatk4/4.1.0.0
```

2. Run GATK4 with SLURM.

   An example:

   For this example, we use the GATK tutorial Run the Pathseq pipeline

Listing 39: `gatk.sh`

```
#!/bin/sh
↪
↪

#SBATCH --partition=longjobs
↪
↪
#SBATCH --nodes=1
↪
↪
#SBATCH --ntasks-per-node=1
↪
↪
#SBATCH --time=05:00
↪
↪
#SBATCH --job-name=gatk_example
↪
↪
#SBATCH -o gatk4_%j.out
↪
↪
```

(continues on next page)

(continued from previous page)

```
#SBATCH -e gatk4_%j.err
↪
↪
#SBATCH --mail-type=END,FAIL
↪
↪
#SBATCH --mail-user=youremail@email.com

# Don't share environment variables
↪
↪

module load gatk4/4.1.0.0

gatk PathSeqPipelineSpark \
    --input test_sample.bam \
    --filter-bwa-image hg19mini.fasta.img \
    --kmer-file hg19mini.hss \
    --min-clipped-read-length 70 \
    --microbe-fasta e_coli_k12.fasta \
    --microbe-bwa-image e_coli_k12.fasta.img \
    --taxonomy-file e_coli_k12.db \
    --output output.pathseq.bam \
    --scores-output output.pathseq.txt
```

Note: If you want to run some tests, go to the GATK4 page with tutorials.

Authors

- Manuela Carrasco Pinzón <mcarras1@eafit.edu.co>

3.3.27 GROMACS

GROMACS[1] is a versatile package to perform molecular dynamics, i.e. simulate the Newtonian equations of motion for systems with hundreds to millions of particles. It is primarily designed for biochemical molecules like proteins, lipids and nucleic acids that have a lot of complicated bonded interactions, but since GROMACS is extremely fast at calculating the nonbonded interactions (that usually dominate simulations) many groups are also using it for research on non-biological systems, e.g. polymers.

GROMACS 2019.3

Table of Contents

[1] GROMACS. (2019, June 14). GROMACS. Fast, Flexible and Free. Retrieved July 10, 2019, from http://www.gromacs.org/About_Gromacs

Basic information

- **Official Website:** http://manual.gromacs.org/documentation/
- **License:** GNU Lesser General Public License (LGPL), version 2.1.
- **Installed on:** *Apolo II*

Tested on (Requirements)

- **OS base:** CentOS (x86_64) $\geq$ 6.6 (Rocks 6.2)
- **Compiler:** Intel MPI Library $\geq$ 17.0.1 (Apolo)
- **Math Library:** FFTW 3.3.8 (Built in) and OpenBlas 0.2.19

Installation

The following procedure present the way to compile **GROMACS 2019.3** for parallel computing using the GROMACS built-in thread-MPI and CUDA.[1]

Note: For the building, the Intel compiler 2017 was used due to compatibility issues with CUDA which only supports, for Intel as backend compiler, up to 2017 version.

---

1. Download the latest version of GROMACS

```
$ wget http://ftp.gromacs.org/pub/gromacs/gromacs-2019.3.tar.gz
$ tar xf gromacs-2019.3.tar.gz
```

2. Inside the folder, on the top create a `build` directory where the installation binaries will be put by cmake.

```
$ cd gromacs-2019.3
$ mkdir build
$ cd build
```

3. Load the necessary modules for the building.

```
$ module load cmake/3.7.1 \
              cuda/9.0 \
              openblas/0.2.19_gcc-5.4.0 \
              intel/2017_update-1 \
              python/2.7.15_miniconda-4.5.4
```

4. Execute the cmake command with the desired directives.

[1] GROMACS Documentation. (2019, June 14). GROMACS. Fast, Flexible and Free. Retrieved July 10, 2019, from http://manual.gromacs.org/documentation/current/manual-2019.3.pdf

```
$ cmake .. -DGMX_GPU=on -DCUDA_TOOLKIT_ROOT_DIR=/share/apps/cuda/9.0/ -DGMX_CUDA_
↪TARGET_SM="30;37;70" \
           -DGMX_SIMD=AVX2_256 -DCMAKE_INSTALL_PREFIX=/share/apps/gromacs/2019.3_
↪intel-17_cuda-9.0 \
           -DGMX_FFT_LIBRARY=fftw3 -DGMX_BUILD_OWN_FFTW=ON -DGMX_EXTERNAL_
↪BLAS=on -DREGRESSIONTEST_DOWNLOAD=on
```

Note: The above command will enable the GPU usage with CUDA for the specified architecures, sm_30 and sm_37 for Tesla K80 and sm_70 for V100 because these are the GPUs used in Apolo.[2]

Note: For "FFT_LIBRARY" there are some options like Intel MKL. Generally, it is recommended to use the FFTW because there is no advantage in using MKL with GROMACS, and FFTW is often faster.[1]

To build the distributed GROMACS version you have to use an MPI library. The GROMACS team recommends OpenMPI version 1.6 (or higher), MPICH version 1.4.1 (or higher).

```
$ module load cmake/3.7.1 \
              cuda/9.0 \
              openblas/0.2.19_gcc-5.4.0 \
              openmpi/1.10.7_gcc-5.4.0 \
              python/2.7.15_miniconda-4.5.4
```

```
$ cmake .. -DCMAKE_C_COMPILER=mpicc -DCMAKE_CXX_COMPILER=mpicxx -DGMX_MPI=on -
↪DGMX_GPU=on \
           -DCUDA_TOOLKIT_ROOT_DIR=/share/apps/cuda/9.0/ -DGMX_CUDA_TARGET_SM="30;
↪37;70" \
           -DGMX_SIMD=AVX2_256 -DCMAKE_INSTALL_PREFIX=/share/apps/gromacs/2019.3_
↪intel-17_cuda-9.0 \
           -DGMX_FFT_LIBRARY=fftw3 -DGMX_BUILD_OWN_FFTW=ON -DGMX_EXTERNAL_BLAS=on␣
↪-DREGRESSIONTEST_DOWNLOAD=on
```

For more information about the compile options you can refer to the Gromacs Documentation.[1]

5. Execute the make commands sequence.

```
$ make -j <N>
$ make check
$ make -j <N> install
```

Warning: Some tests may fail, but the installation can continue depending on the number of failed tests.

Usage

This section describes a way to submit jobs with the resource manager SLURM.

1. Load the necessary environment.

[2] Matching SM architectures. (2019, November 11). Blame Arnon Blog. Retrieved July 10, 2019, from https://arnon.dk/matching-sm-architectures-arch-and-gencode-for-various-nvidia-cards/

```
# Apolo
module load gromacs/2019.3_intel-17_cuda-9.0

# Cronos
module load gromacs/2016.4_gcc-5.5.0
```

2. Run Gromacs with SLURM.

   a. An example with GPU (Apolo), given by one of our users:

```
#!/bin/bash

#SBATCH --job-name=gmx-GPU
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=8
#SBATCH --cpus-per-task=4
#SBATCH --time=10:00:00
#SBATCH --partition=accel-2
#SBATCH --gres=gpu:2
#SBATCH --output=gmx-GPU.%j.out
#SBATCH --error=gmx-GPU.%j.err

module load gromacs/2019.3_intel-17_cuda-9.0

export OMP_NUM_THREADS=$SLURM_CPUS_PER_TASK

gmx grompp -f step6.0_minimization.mdp -o step6.0_minimization.tpr -c␣
↪step5_charmm2gmx.pdb -r step5_charmm2gmx.pdb -p topol.top
gmx mdrun -v -deffnm step6.0_minimization -ntmpi $SLURM_NTASKS -ntomp
↪$SLURM_CPUS_PER_TASK -gpu_id 01

# Equilibration
cnt=1
cntmax=6

while [ $cnt -le $cntmax ]; do
    pcnt=$((cnt-1))
    if [ $cnt -eq 1 ]; then
        gmx grompp -f step6.${cnt}_equilibration.mdp -o step6.${cnt}_
↪equilibration.tpr -c step6.${pcnt}_minimization.gro -r step5_charmm2gmx.
↪pdb -n index.ndx -p topol.top
        gmx mdrun -v -deffnm step6.${cnt}_equilibration -ntmpi $SLURM_
↪NTASKS -ntomp $SLURM_CPUS_PER_TASK -gpu_id 01
    else
        gmx grompp -f step6.${cnt}_equilibration.mdp -o step6.${cnt}_
↪equilibration.tpr -c step6.${pcnt}_equilibration.gro -r step5_
↪charmm2gmx.pdb -n index.ndx -p topol.top
        gmx mdrun -v -deffnm step6.${cnt}_equilibration -ntmpi $SLURM_
↪NTASKS -ntomp $SLURM_CPUS_PER_TASK -gpu_id 01
    fi
    ((cnt++))
done

# Production
cnt=1
cntmax=10

while [ $cnt -le $cntmax ]; do
```

(continues on next page)

(continued from previous page)

```
41      if [ $cnt -eq 1 ]; then
42          gmx grompp -f step7_production.mdp -o step7_${cnt}.tpr -c step6.6_
    ↪equilibration.gro -n index.ndx -p topol.top
43          gmx mdrun -v -deffnm step7_${cnt} -ntmpi $SLURM_NTASKS -ntomp
    ↪$SLURM_CPUS_PER_TASK -gpu_id 01
44      else
45          pcnt=$((cnt-1))
46          gmx grompp -f step7_production.mdp -o step7_${cnt}.tpr -c step7_$
    ↪{pcnt}.gro -t step7_${pcnt}.cpt -n index.ndx -p topol.top
47          gmx mdrun -v -deffnm step7_${cnt} -ntmpi $SLURM_NTASKS -ntomp
    ↪$SLURM_CPUS_PER_TASK -gpu_id 01
48      fi
49      ((cnt++))
50  done
```

Note lines 18, 28, 31, 43, 47 the use of `gmx mdrun` with the flag `-gpu_id 01`:

- If Gromacs was compiled with Cuda, it will use the GPUs available by default.
- The flag `-gpu_id 01` tells Gromacs which GPUs can be used. The `01` means use GPU with device ID 0 and GPU with device ID 1.
- Note in line 9 the use of `#SBATCH -gres=gpu:2`. `gres` stands for *generic resource scheduling*. `gpu` requests GPUs to Slurm, and `:2` specifies the quantity.
- Note that we have 3 GPUs in Accel-2, but we are indicating only two GPUs. This is useful when some other user is using one or more GPUs.
- Also, note that the number of tasks per node must be a multiple of the number of GPUs that will be used.
- Setting a `cpus-per-task` to a value between 2 and 6 seems to be more efficient than values greather than 6.
- The files needed to run the example above are `here`.
- For more information see[3].

b. An example with CPU only (Cronos):

```
1   #!/bin/bash
2
3   ##################################################################################
    ↪######
4   ##################################################################################
    ↪######
5   #
6   # Find out the density of TIP4PEW water.
7   # How to run the simulation was taken from:
8   # https://www.svedruziclab.com/tutorials/gromacs/1-tip4pew-water/
9   #
10  ##################################################################################
    ↪######
11  ##################################################################################
    ↪######
12
13  #SBATCH --job-name=gmx-CPU
```

(continues on next page)

[3] Getting good performance from mdrun. (2019). GROMACS Development Team. Retrieved September 3, 2019, from http://manual.gromacs.org/documentation/current/user-guide/mdrun-performance.html#running-mdrun-within-a-single-node

(continued from previous page)

```
14 #SBATCH --nodes=4
15 #SBATCH --ntasks-per-node=16
16 #SBATCH --time=03:00:00
17 #SBATCH --partition=longjobs
18 #SBATCH --output=gmx-CPU.%j.out
19 #SBATCH --error=gmx-CPU.%j.err
20 #SBATCH --mail-user=example@eafit.edu.co
21 #SBATCH --mail-type=END,FAIL
22
23 module load gromacs/2016.4_gcc-5.5.0
24
25 # Create box of water.
26 gmx_mpi solvate -cs tip4p -o conf.gro -box 2.3 2.3 2.3 -p topol.top
27
28 # Minimizations.
29 gmx_mpi grompp -f mdp/min.mdp -o min -pp min -po min
30 srun --mpi=pmi2 gmx_mpi mdrun -deffnm min
31
32 gmx_mpi grompp -f mdp/min2.mdp -o min2 -pp min2 -po min2 -c min -t min
33 srun --mpi=pmi2 gmx_mpi mdrun -deffnm min2
34
35 # Equilibration 1.
36 gmx_mpi grompp -f mdp/eql.mdp -o eql -pp eql -po eql -c min2 -t min2
37 srun --mpi=pmi2 gmx_mpi mdrun -deffnm eql
38
39 # Equilibration 2.
40 gmx_mpi grompp -f mdp/eql2.mdp -o eql2 -pp eql2 -po eql2 -c eql -t eql
41 srun --mpi=pmi2 gmx_mpi mdrun -deffnm eql2
42
43 # Production.
44 gmx_mpi grompp -f mdp/prd.mdp -o prd -pp prd -po prd -c eql2 -t eql2
45 srun --mpi=pmi2 gmx_mpi mdrun -deffnm prd
```

- Note the use of `gmx_mpi` instead of `gmx`.
- Also, note the use of `srun --mpi=pmi2` instead of `mpirun -np <num-tasks>`. The command `srun --mpi=pmi2` gives to `gmx_mpi` the context of where and how many tasks to run.
- In lines 13 and 14 note that it is requesting 4 nodes and 16 mpi tasks on each node. Recall that each node in Cronos has 16 cores.
- In lines 16, 29, 32, 36, 40, 44 note that `srun --mpi=pmi2` is not used. This is due that, those are preprocessing steps, they do not need to run distributedly.
- The needed files to run the example simulation can be found `here`.

References

Authors

- Johan Sebastián Yepes Ríos <jyepesr1@eafit.edu.co>
- Hamilton Tobón Mosquera <htobonm@eafit.edu.co>

GROMACS 5.1.4

- **Installation Date:** 02/22/2017
- **URL:** http://www.gromacs.org/
- – **Installed on:** *Apolo II*
- **License:** GNU Lesser General Public License (LGPL), version 2.1.

Table of Contents

Dependencies

- GNU GCC >= 5.4.0
- OpenMPI >= 1.8.8 (Do not include version with PMI2)
- Python> = 2.7.X (Check that it does not include MKL or BLAS libraries)
- CUDA> = 8.0 (Must include SDK)
- cmake

Installation

After resolving the aforementioned dependencies, you can proceed with the installation of GROMACS.

1. Download the latest version of GROMACS

```
$ wget ftp://ftp.gromacs.org/pub/gromacs/gromacs-5.1.4.tar.gz
$ tar -zxvf gromacs-5.1.4.tar.gz
```

2. After unpacking GROMACS, we continue with the following configuration and compilation steps:

> **Note:** You have to load all the apps in the dependencies, or add them to the PATH, we are assuming that all is done

```
$ cd gromacs-5.4.1
$ mkdir build
$ cd build
```

3. compile it

```
$ cmake .. -DGMX_GPU=ON -DCUDA_TOOLKIT_ROOT_DIR=/share/apps/cuda/8.0 -DGMX_MPI=ON
    -DCMAKE_INSTALL_PREFIX=/share/apps/gromacs/5.1.4/gcc/5.4.0/cuda/8.0
    -DGMX_EXTERNAL_BLAS=/share/apps/openblas/0.2.19/gcc/5.4.0/lib
    -DGMX_BUILD_OWN_FFTW=ON -DGMX_EXTERNAL_BOOST=ON
    -DBoost_INCLUDE_DIR="/share/apps/boost/1.63.0/gcc/5.4.0_openmpi-1.8.8-x86_64_
↪cuda-8.0/include"
    -DBoost_LIBRARY_DIRS="/share/apps/boost/1.63.0/gcc/5.4.0_openmpi-1.8.8-x86_64_
↪cuda-8.0/lib"
    -DBoost_DIR="/share/apps/boost/1.63.0/gcc/5.4.0_openmpi-1.8.8-x86_64_cuda-8.0/
↪"
    -DNVML_INCLUDE_DIR=/share/apps/cuda/8.0/include 2>&1 | tee gromacs-cmake.log
$ cmake .. CC=mpicc CXX=mpicxx cmake .. -DGMX_GPU=ON -DGPU_DEPLOYMENT_KIT_ROOT_
↪DIR=/share/apps/cuda/8.0
    -DGMX_MPI=ON -DCMAKE_INSTALL_PREFIX=/share/apps/gromacs/5.1.4/gcc-5.4.0_cuda-
↪8.0
    -DGMX_BUILD_OWN_FFTW=ON -DGMX_PREFER_STATIC_LIBS=ON -DCMAKE_BUILD_
↪TYPE=Release -DGMX_BUILD_UNITTESTS=ON
    -DREGRESSIONTEST_DOWNLOAD=ON 2>&1 | tee gromacs-cmake.log
$ make 2>&1 | tee gromacs-make.log
$ make check 2>&1 | tee gromacs-check.log
$ make install 2>&1 | tee gromacs-make-install.log
```

References

- Gromacs instructions. Retrieved July 10, 2019, from http://www.gromacs.org/Documentation/Installation_Instructions_5.0
- Gromacs github. Retrieved July 10, 2019, from https://github.com/gromacs/gromacs/blob/master/cmake/FindNVML.cmake
- Lindqvist. Retrieved July 10, 2019, from http://verahill.blogspot.com.co/2013/04/396-compiling-gromacs-46-with-openblas.html
- Compiling GROMACS on Cluster. Retrieved July 10, 2019, from https://ringo.ams.stonybrook.edu/index.php/Compiling_GROMACS_on_Cluster
- Lindqvist. Retrieved July 10, 2019, from http://verahill.blogspot.com/2012/03/building-gromacs-with-fftw3-and-openmpi.html
- How to compile gromacs. Retrieved July 10, 2019, from https://mini.ourphysics.org/wiki/index.php/How_to_compile_Gromacs
- Github issue. Retrieved July 10, 2019, from https://github.com/linux-sunxi/linux-sunxi/issues/62
- Nvidia gromacs. Retrieved July 10, 2019, from https://ngc.nvidia.com/catalog/containers/hpc:gromacs
- Running VASP on Nvidia GPUs. Retrieved July 10, 2019, from https://www.nsc.liu.se/~pla/
- Gromacs (GPU). Retrieved July 10, 2019, from http://www.hpcadvisorycouncil.com/pdf/GROMACS_GPU.pdf

Authors

- Mateo Gómez Zuluaga

GROMACS 5.1.4 PLUMED 2.3.5

- **Installation Date:** 14/06/2018

- **URL:** http://www.gromacs.org/
- – **Installed on:** *Apolo II*
- **License:** GNU Lesser General Public License (LGPL), version 2.1.

Table of Contents

Dependencies

- GNU GCC >= 5.4.0
- Mpich2 >= 3.2
- Python >= 2.7.15 (Miniconda)
- OpenBLAS >= 0.2.19
- CUDA >= 9.0 (have to include SDK)
- Plumed >= 2.3.5
- Boost >= 1.67.0

Installation

After resolving the aforementioned dependencies, you can proceed with the installation of GROMACS.

1. Download the latest version of GROMACS

```
$ wget ftp://ftp.gromacs.org/pub/gromacs/gromacs-5.1.4.tar.gz
$ tar -zxvf gromacs-5.1.4.tar.gz
```

2. After unpacking GROMACS, we continue with the following configuration and compilation steps:

> **Note:** You have to load all the apps in the dependencies, or add them to the PATH, we are assuming that all is done

```
$ cd gromacs-5.4.1
$ plumed patch -p --shared #select the fifth option
$ mkdir build
$ cd build
```

3. compile it

```
$ CC=mpicc CXX=mpicxx FC=mpif90 cmake .. -DGMX_GPU=ON -DCUDA_TOOLKIT_ROOT_DIR=/
↪share/apps/cuda/9.0
    -DGMX_MPI=ON -DCMAKE_INSTALL_PREFIX=/share/apps/gromacs/5.1.4/gcc-5.4.0_
↪plumed-2.3.5
    -DGMX_EXTERNAL_BLAS="/share/apps/openblas/0.2.19/gcc/5.4.0/lib/libopenblas.so"
    -DGMX_BUILD_OWN_FFTW=ON -DGMX_EXTERNAL_BOOST=ON
    -DBoost_INCLUDE_DIR="/share/apps/boost/1.67.0/gcc-5.4.0_mpich2-3.2/include"
    -DBoost_LIBRARY_DIRS="/share/apps/boost/1.67.0/gcc-5.4.0_mpich2-3.2/lib"
    -DBoost_DIR="/share/apps/boost/1.67.0/gcc-5.4.0_mpich2-3.2" -DNVML_INCLUDE_
↪DIR=/share/apps/cuda/9.0/include
    -DCMAKE_BUILD_TYPE=Release -DGMX_BUILD_UNITTESTS=ON  -DREGRESSIONTEST_
↪DOWNLOAD=ON  2>&1 | tee gromacs-cmake.log
$ make 2>&1 | tee gromacs-make.log
$ make check 2>&1 | tee gromacs-check.log
$ make install 2>&1 | tee gromacs-make-install.log
```

Use mode

To run Gromacs + Plumed it is necessary to have the following files: - plumed.dat - md_1B2S_AA27K.tpr

Definition of input files

```
# set up two variables for Phi and Psi dihedral angles
phi: TORSION ATOMS=5,7,9,15
psi: TORSION ATOMS=7,9,15,17
#
# Activate metadynamics in phi and psi
# depositing a Gaussian every 500 time steps,
# with height equal to 1.2 kJoule/mol,
# and width 0.35 rad for both CVs.
#
metad: METAD ARG=phi,psi PACE=500 HEIGHT=1.2 SIGMA=0.35,0.35 FILE=HILLS
# monitor the two variables and the metadynamics bias potential
PRINT STRIDE=10 ARG=phi,psi,metad.bias FILE=COLVAR
```

References

- Gromacs instructions. Retrieved July 10, 2019, from http://www.gromacs.org/Documentation/Installation_Instructions_5.0
- Gromacs github. Retrieved July 10, 2019, from https://github.com/gromacs/gromacs/blob/master/cmake/FindNVML.cmake
- Lindqvist. Retrieved July 10, 2019, from http://verahill.blogspot.com.co/2013/04/396-compiling-gromacs-46-with-openblas.html
- Compiling GROMACS on Cluster. Retrieved July 10, 2019, from https://ringo.ams.stonybrook.edu/index.php/Compiling_GROMACS_on_Cluster
- Lindqvist.Retrieved July 10, 2019, from http://verahill.blogspot.com/2012/03/building-gromacs-with-fftw3-and-openmpi.html
- How to compile gromacs. Retrieved July 10, 2019, from https://mini.ourphysics.org/wiki/index.php/How_to_compile_Gromacs
- Github issue. Retrieved July 10, 2019, from https://github.com/linux-sunxi/linux-sunxi/issues/62

- Nvidia gromacs. Retrieved July 10, 2019, from https://ngc.nvidia.com/catalog/containers/hpc:gromacs
- Running VASP on Nvidia GPUs. Retrieved July 10, 2019, from https://www.nsc.liu.se/~pla/
- Gromacs (GPU). Retrieved July 10, 2019, from http://www.hpcadvisorycouncil.com/pdf/GROMACS_GPU.pdf

Authors

- Mateo Gómez Zuluaga

GROMACS 2016.4

- **Installation Date:** 31/01/2018
- **URL:** http://www.gromacs.org/
- – **Installed on:** Cronos
- **License:** GNU Lesser General Public License (LGPL), version 2.1.

Table of Contents

Dependencies

- GNU GCC >= 5.5.0
- OpenMPI >= 2.1.2
- Fftw >= 3.3.7
- Cmake >= 3.3
- OpenBlas >= 0.2.20

Installation

After resolving the aforementioned dependencies, you can proceed with the installation of GROMACS.

1. Download the latest version of GROMACS[1]

```
$ wget http://ftp.gromacs.org/pub/gromacs/gromacs-2016.4.tar.gz
$ tar -xzvf gromacs-2016.4
```

2. After unpacking GROMACS, we continue with the following configuration and compilation steps:

[1] Gromacs downloads. Retrieved January 31, 2018, from http://manual.gromacs.org/documentation/2016.4/download.html

Note: You have to load all the apps in the dependencies, or add them to the PATH, we are assuming that all is done

```
$ cd gromacs-2016.4
$ mkdir build
$ cd build
$ cmake .. -DCMAKE_INSTALL_PREFIX=/share/apps/gromacs/2016.4 -DREGRESSIONTEST_
↪DOWNLOAD=ON
    -DFFTWF_INCLUDE_DIR=/share/apps/fftw/3.3.7/gcc-5.5.0/  -DCMAKE_PREFIX_PATH=/
↪share/apps/openblas/
    -DGMX_EXTERNAL_BLAS=ON | tee cmake.log
$ make -j4 | tee make.log
$ make check
$ make install
```

References

Authors

- Juan David Arcila-Moreno

3.3.28 GROMACS-LS

GROMACS-LS[1] is a custom version of GROMACS v4.5.5 developed for local stress calculations from molecular simulations. It is a post-processing tool to analyze existing trajectories to obtain stress fields in 3D. GROMACS-LS outputs a simple binary file (see the manual for details on the format) which can be can be converted to other formats or further processed with the included tensortools python utility.

GROMACS-LS 4.5.5

Table of Contents

Basic information

- **Official Website:** https://www.mdstress.org/index.php/gromacs-ls/
- **License:** GNU General Public License (GPL), version 2.

[1] GROMACS-LS. (Na). GROMACS-LS. Custom Version of GROMACS. Retrieved May 14, 2020, from https://www.mdstress.org/files/5914/4657/7530/Local_stress.pdf

- **Installed on:** *Apolo II*

Tested on (Requirements)

- **OS base:** CentOS (x86_64) $\geq$ 6.6 (Rocks 6.2)
- **Compiler:** GCC 5.4.0
- **Math Library:** FFTW 3.3.5
- **Others:** LAPACK Version 3.5.0

Installation

The following procedure presents the way to compile **GROMACS-LS 4.5.5** for local stress calculations from molecular simulations.[1]

Note: For the building, the GCC compiler version 5.4.0 was used due to better compatibility with FFTW3 and LAPACK.

---

1. Load the necessary modules for the building.

```
$ module load cmake/3.7.1 \
              gcc/5.4.0 \
              fftw/3.3.5_gcc-5.4.0_openmpi-1.8.8-x86_64 \
              lapack/3.5.0_gcc-5.4.0
```

Note: LAPACK V3.5.0 is the recommended version of LAPACK from GROMACS-LS, you can see how to install it in our Scientific Libraries section or here *LAPACK 3.5.0*.

---

2. Download the latest version of GROMACS-LS

```
$ wget https://www.mdstress.org/files/4314/6351/1328/gromacs-4.5.5-ls-5.0.tar.gz
$ tar xfz gromacs-4.5.5-ls-5.0.tar.gz
```

3. Inside the folder, on the top create a `build` directory where the installation binaries will be generated by cmake.

```
$ cd gromacs-4.5.5-ls-5.0
$ mkdir build
$ cd build
```

4. Execute the following commands, those will export the environmental variables that GROMACS-LS needs for compilation.

```
$ export FFTW3_ROOT_DIR=/path/to/fftw3
$ export CMAKE_PREFIX_PATH=/path/to/lapack
```

Note: We just installed LAPACK, so in our case `CMAKE_PREFIX_PATH` should point to `/share/apps/lapack/3.5.0/gcc/5.4.0/`

---

[1] GROMACS-LS Documentation. Custom Version of GROMACS. Retrieved May 14, 2020, from https://www.mdstress.org/files/5914/4657/7530/Local_stress.pdf

5. Execute the cmake command with the desired directive destiny.

```
$ cmake .. -DCMAKE_INSTALL_PREFIX=/share/apps/gromacs-ls/4.5.5/gcc/5.4.0/
```

6. Execute the make commands sequence.

```
$ make -j <N>
$ make -j <N> install
```

7. After the installation is completed, you have to create the corresponding module for GROMACS-LS V4.5.5.

```
##%Module1.0#####################################################################
##
## modulefile /share/apps/modules/gromacs-ls/4.5.5_gcc-5.4.0
## Written by Juan Diego Ocampo & Tomas David Navarro Munera
##

proc ModulesHelp { } {
     global version modroot
          puts stderr "\t Gromacs-ls 4.5.5"
}

module-whatis "(Name________) Gromacs-ls"
module-whatis "(Version_____) 4.5.5"
module-whatis "(Compilers___) gcc-5.4.0"
module-whatis "(System______) x86_64-redhat-linux"

set     topdir            /share/apps/gromacs-ls/4.5.5/gcc/5.4.0
set     version           4.5.5
set     sys               x86_64-redhat-linux

module load fftw/3.3.5_gcc-5.4.0_openmpi-1.8.8-x86_64
module load lapack/3.5.0_gcc-5.4.0

prepend-path      PATH                    $topdir/bin

prepend-path      C_INCLUDE_PATH          $topdir/include
prepend-path      CXX_INCLUDE_PATH        $topdir/include
prepend-path      CPLUS_INCLUDE_PATH      $topdir/include

prepend-path      LD_LIBRARY_PATH         $topdir/lib
prepend-path      LIBRARY_PATH            $topdir/lib
prepend-path      LD_RUN_PATH             $topdir/lib

prepend-path      MANPATH                 $topdir/share/man
```

References

Authors

- Tomas David Navarro Munera <tdnavarrom@eafit.edu.co>

3.3.29 gurobi

Description

Gurobi (Optimization) is focused on providing the best optimization solver possible, with outstanding support, and no-surprises pricing.

Table of Contents

GUROBI 5.6.0

- **Installation date:** 18/08/2014
- **URL:** http://www.gurobi.com/
- **Apolo version:** Apolo I
- **License:** ACADEMIC LICENSE

Installation

As it is a license, you have to search the process which is different in each case

Author

- Mateo Gómez Zuluága

Table of Contents

GUROBI 7.5.0

- **Installation date:**
- **URL:** http://www.gurobi.com/
- **Apolo version:** Cronos
- **License:** ACADEMIC LICENSE

Installation

1. First download the tar from the main page, then:

```
$ tar xvfz gurobi7.5.2_linux64.tar.gz -C <installdir>
```

Module

```
##%Module1.0#####################################################################
##
## module gurobi/7.5.2
##
## /share/apps/modules/gurobi/7.5.2 Written by Juan David Pineda-Cárdenas
##

proc ModulesHelp { } {
    puts stderr "\tgurobi/7.5.2 - sets the Environment for gurobi \
    \n\tin the share directory /share/apps/gurobi/7.5.2\n"
}

module-whatis "\n\n\tSets the environment for using gurobi \
              \n\tbuilded with python/2.7.13_intel-18_u1\n"

# for Tcl script use only
set           topdir     /share/apps/gurobi/7.5.2
set   version    7.5.2
set       sys            linux-x86_64
set       user           [exec bash -c "echo \$USER"]
set       hostname   [exec bash -c "echo \$HOSTNAME"]
set           host       [string map {".local" ""} $hostname ]

conflict gurobi

module load python/2.7.13_intel-18_u1
module load java/jdk-8_u152

setenv GUROBI_HOME          $topdir
setenv GUROBI_LICENSE_HOME  /home/$user/gurobi-licenses
setenv GRB_LICENSE_FILE             /home/$user/gurobi-licenses/$host/gurobi.lic

prepend-path PATH           $topdir/bin
prepend-path LD_LIBRARY_PATH        $topdir/lib
prepend-path LIBRARY_PATH   $topdir/lib
prepend-path LD_RUN_PATH    $topdir/lib

prepend-path CXX_INCLUDE_PATH       $topdir/include
prepend-path CPLUS_INCLUDE_PATH $topdir/include
prepend-path C_INCLUDE_PATH         $topdir/include

prepend-path CLASSPATH              $topdir/lib/gurobi.jar
```

References

- http://www.gurobi.com/documentation/7.5/quickstart_linux.pdf

Author

- Andrés Felipe Zapata Palacio

Table of Contents

GUROBI 9.5.0

- **Installation date:** 17/02/2022
- **URL:** http://www.gurobi.com/
- **Apolo version:** Apolo II
- **License:** ACADEMIC LICENSE

Installation

1. First download the tar from the main page, then:

```
$ tar -xvf gurobi9.5.0_linux64.tar.gz -C <installdir>
```

Module

```
##%Module1.0#####################################################################
##
## module gurobi/9.5.0
##
## /share/apps/modules/gurobi/9.5.0
##
## Written by Laura Sanchez Cordoba
##

proc ModulesHelp { } {
    puts stderr "\tgurobi/9.5.0 - sets the Environment for gurobi 9.5.0"
}

module-whatis "\n\n\tSets the environment for using gurobi, \
               \n\tminiconda3-4.10.3-gcc-11.2.0-vcglj27"

# for Tcl script use only
set       topdir     /share/apps/gurobi/9.5.0/gurobi950/linux64
set       version    9.5.0
set       sys        linux-x86_64
set       user       [exec bash -c "echo \$USER"]
```

(continues on next page)

(continued from previous page)

```
set       hostname   [exec bash -c "echo \$HOSTNAME"]
set       host       [string map {".local" ""} $hostname ]

conflict gurobi

module load miniconda3-4.10.3-gcc-11.2.0-vcglj27

setenv GUROBI_HOME              $topdir
setenv GUROBI_LICENSE_HOME      /home/$user/gurobi-licenses/v950
setenv GRB_LICENSE_FILE         /home/$user/gurobi-licenses/v950/$host/gurobi.lic

prepend-path PATH               $topdir/bin
prepend-path LD_LIBRARY_PATH    $topdir/lib
prepend-path LIBRARY_PATH       $topdir/lib
prepend-path LD_RUN_PATH        $topdir/lib

prepend-path CXX_INCLUDE_PATH   $topdir/include
prepend-path CPLUS_INCLUDE_PATH $topdir/include
prepend-path C_INCLUDE_PATH     $topdir/include

prepend-path CLASSPATH          $topdir/lib/gurobi.jar

prepend-path PYTHONPATH         $topdir
```

Load the module

```
$ module load gurobi/9.5.0
```

References

- https://www.gurobi.com/documentation/9.5/quickstart_linux/index.html

Author

- Laura Sánchez Córdoba

3.3.30 IQ-TREE

The IQ-TREE[1] software was created as the successor of IQPNNI and TREE-PUZZLE (thus the name IQ-TREE). IQ-TREE was motivated by the rapid accumulation of phylogenomic data, leading to a need for efficient phylogenomic software that can handle a large amount of data and provide more complex models of sequence evolution. To this end, IQ-TREE can utilize multicore computers and distributed parallel computing to speed up the analysis. IQ-TREE automatically performs checkpointing to resume an interrupted analysis.

IQ-TREE 2.1.2

Table of Contents

[1] https://github.com/iqtree/iqtree2/

Basic information

- **Official Website:** https://github.com/iqtree/iqtree2/
- **License:** GNU GENERAL PUBLIC LICENSE Version 3
- **Installed on:** Apolo II
- **Installation date:** 07/02/2021

Tested on (Requirements)

- **OS base:** CentOS (x86_64) $\geq$ 6.6 (Rocks 6.2)
- **Dependencies:**
  - GCC > 7.4.0

Installation

1. Download the desired version of the software (Source code - tar.gz)

```
mkdir apps/iqutree -p
cd apps/iqutree/
wget https://github.com/iqtree/iqtree2/releases/download/v2.1.2/iqtree-2.
↪1.2-Linux.tar.gz
tar xfz iqtree-2.1.2-Linux.tar.gz
```

2. Enter to the extracted folder.

```
cd icu-release-68-1/iqtree-2.1.2-Linux
module load gcc/7.4.0
```

3. Continue with the following steps:

```
sudo mkdir -p /share/apps/iqutree/2.1.2/
sudo cp * /share/apps/iqutree/2.1.2/ -r
```

Module

```
#%Module1.0#####################################################################
↪###
##
## module iqutree/2.1.2
##
## /share/apps/modules/iqutree/2.1.2 Written by Tomas Navarro Munera
##

proc ModulesHelp { } {
    puts stderr "\tiqutree/2.1.2 - sets the Environment for BEAST2 \
    \n\tin the share directory /share/apps/iqutree/2.1.2\n"
}

module-whatis "\n\n\tSets the environment for using beast2\n"

# for Tcl script use only
set     topdir          /share/apps/iqutree/2.1.2
set     version         2.1.2
set     sys             linux-x86_64

conflict iqutree

module load gcc/7.4.0

prepend-path PATH               $topdir/bin
```

Usage mode

```
module load iqutree/2.1.2
```

Resources

- https://github.com/iqtree/iqtree2/

Authors

- Tomas Navarro <tdnavarrom@eafit.edu.co>

3.3.31 Kraken

[1] **Kraken 2 is the newest version of Kraken, a taxonomic classification system using** exact k-mer matches to achieve high accuracy and fast classification speeds. This classifier matches each k-mer within a query sequence to the lowest common ancestor (LCA) of all genomes containing the given k-mer. The k-mer assignments inform the classification algorithm.

Kraken 2.1.0

[1] https://ccb.jhu.edu/software/kraken2/

Table of Contents

Basic information

- **Official Website:** https://ccb.jhu.edu/software/kraken2/
- **Downloads page:** https://github.com/DerrickWood/kraken2
- **Installed on:** APOLO II

Tested on (Requirements)

- **OS base:** CentOS (x86_64) $\geq$ 6
- **Compiler** Perl 5.26.1 and GCC 7.4.0

Installation

1. First of all, you must load the following modules for the compilation.

```
$ module load gcc/7.4.0
$ module load perl/5.26.1_gcc-4.4.7
```

2. After that, download the source code from github and move into the directory generated.

```
$ git clone https://github.com/DerrickWood/kraken2.git
$ cd kraken2/
```

3. Before to start the installation, you should make two directories, the first one is an installation directory acting as "prefix", the second one is the bin directory, the last one is useful if you want to place the kraken2 scripts in you $PATH.

```
$ mkdir -p $HOME/kraken/{instalation,bin}
$ # This sets a temporal environment variable to set the "prefix path"
$ KRAKEN_DIR="$HOME/kraken/instalation/"
```

4. Then you can continue with the installation, after execute the install script you should copy three binary files from the "installation prefix" (install directory) to the binaries (bin directory), mentioned above.

```
$ ./install_kraken2.sh $KRAKEN_DIR
$ cd $KRAKEN_DIR/..
$ cp instalation/kraken2{,-build,-inspect} bin/
```

5. If the installation was successful then you should:

```
$ cd bin/
$ ./kraken2 --version
```

6. Optional: If you want, you can add the $HOME/kraken/bin/ directory to your $PATH.

Authors

- Laura Sanchez Cordoba
- Samuel Palacios Bernate

3.3.32 IOAPI

IOAPI[1] This library provides Fortran and C APIs for environmental model data access together with related utility routines and tools for analysis and manipulation of data stored by way of that API.

Table of Contents

IOAPI 3.2.1

- **Installation date:** 03/03/2022
- **URL:** https://github.com/cjcoats/ioapi-3.2
- **Apolo version:** Apolo II

Dependencies

- GCC = 9.3.0
- MPICH = 3.4.2
- Zlib = 1.2.11
- Curl = 7.77.0
- Netcdf-c disable netcdf-4 = 4.8.0
- Netcdf-fortran = 4.5.3
- Zstd = 1.5.2

[1] https://github.com/cjcoats/ioapi-3.2

Installation

After solving the aforementioned dependencies, you can proceed with the installation.

1. Download the binaries

```
$ cd /home/blopezp
$ mkdir ioapi
$ cd ioapi
$ git clone https://github.com/cjcoats/ioapi-3.2.git
```

2. Define the architecture to use and create the directory with the same name of the architecture, in our case "Linux2_x86_64gfort".

```
$ export BIN=Linux2_x86_64gfort
$ mkdir Linux2_x86_64gfort
$ ln -sf /share/apps/netcdf-fortran/4.5.3_disable-netcdf-4/gcc-9.3.0/lib/* Linux2_x86_
↪64gfort/
```

3. Units and files.

```
$ module load gcc/9.3.0
$ module load mpich/3.4.2_gcc-9.3.0
$ module load zlib/1.2.11_gcc-9.3.0
$ module load curl/7.77.0_gcc-9.3.0
$ module load netcdf-fortran/4.5.3_gcc-9.3.0_disable-netcdf-4

$ cp ioapi/Makefile.nocpl ioapi/Makefile
$ cp m3tools/Makefile.nocpl m3tools/Makefile
$ cp Makefile.template Makefile
```

4. Add the lines in the Makefile.

```
$ Vim Makefile

CPLMODE     = nocpl
PVMINCL     = /dev/null
LIBINST    = /home/blopezp/ioapi/lib
BININST    = /home/blopezp/ioapi/bin
VERSION     = 3.2-${CPLMODE}
BASEDIR     = ${PWD}
NCFLIBS     = -L/share/apps/netcdf-fortran/4.5.3_disable-netcdf-4/gcc-9.3.0/lib -
↪lnetcdff -lnetcdf -lcurl -lzstd -lz
IODIR       = $(BASEDIR)/ioapi
FIXDIR      = $(IODIR)/fixed_src
HTMLDIR     = $(BASEDIR)/HTML
TOOLDIR     = $(BASEDIR)/m3tools
OBJDIR      = $(BASEDIR)/$(BIN)
```

5. Add the lines in the ioapi/Makefile.

```
$ vim ioapi/Makefile

BASEDIR = /home/blopezp/ioapi
INSTDIR = /home/blopezp/ioapi/lib
IODIR   = ${BASEDIR}/ioapi
```

5. Add the lines in the ioapi/Makeinclude.Linux2_x86_64gfort

```
$ vim ioapi/Makeinclude.Linux2_x86_64gfort

OMPFLAGS  =
OMPLIBS   =
-DIOAPI_NO_STDOUT=1 -DIOAPI_NCF4=1
```

6. Add the lines in the m3tools/Makefile.

```
BASEDIR = /home/blopezp/ioapi
SRCDIR  = ${BASEDIR}/m3tools
IODIR   = ${BASEDIR}/ioapi
OBJDIR  = ${BASEDIR}/${BIN}
INSTDIR = /home/blopezp/ioapi/bin
```

7. Compile Makefile

```
$ make |& tee make.log
$ make install |& tee make-install.log
```

8. Test.

```
$ ./Linux2_x86_64gfort/juldate
```

Note: Users need IOAPI in their respective home for the compilation of WRF-CMAQ, as they need to change data.

References

- https://github.com/cjcoats/ioapi-3.2.1

Author

- Bryan López Parra <blopezp@eafit.edu.co>

3.3.33 LAMMPS

LAMMPS[1] is a classical molecular dynamics code with a focus on materials modeling. It's an acronym for Large-scale Atomic/Molecular Massively Parallel Simulator.

LAMMPS has potentials for solid-state materials (metals, semiconductors) and soft matter (biomolecules, polymers) and coarse-grained or mesoscopic systems. It can be used to model atoms or, more generically, as a parallel particle simulator at the atomic, meso, or continuum scale.

LAMMPS runs on single processors or in parallel using message-passing techniques and a spatial-decomposition of the simulation domain. Many of its models have versions that provide accelerated performance on CPUs, GPUs, and Intel Xeon Phis. The code is designed to be easy to modify or extend with new functionality.

LAMMPS - 22Aug2018

[1] LAMMPS Molecular Dynamics Simulator. Retrieved November 20, 2018, from https://lammps.sandia.gov/

Table of Contents

Basic information

- **Official Website:** https://lammps.sandia.gov/
- **License:** GNU GENERAL PUBLIC LICENSE (GPL)
- **Installed on:** *Apolo II* , *Cronos*

Tested on (Requirements)

- **OS base:** CentOS (x86_64) $\geq$ 6.6 (Rocks 6.2)
- **Compiler:** Intel MPI Library $\geq$ 17.0.1
- **Math Library:** Intel MKL $\geq$ 17.0.1

Installation

The following procedure will compile LAMMPS as an executable, using the Intel MPI implementation and MKL as the Linear Algebra Library.

1. Load the necessary modules for compiling LAMMPS

```
$ module load impi
$ module load mkl
$ module load python/3.x.x
```

Note: The installation script requires Python $\geq$ 3.0

2. Download and compile the desired packages before compiling lammps.

   The flag `yes-all` activates the compilation of all the packages. However, some lammps packages require external libraries and additional configuration procedures.

   After executing `yes-all`, these packages that require additional configuration procedures can be disabled executing `make no-lib`.

These make options (yes-all, no-lib) are used to reproduce the list of packages present in LAMMPS Ubuntu prebuild version. For more information read[1].

```
$ git clone -b stable_22Aug2018 https://github.com/lammps/lammps.git
$ cd lammps/src
$ make yes-all      # install all pkgs in src dir
$ make no-lib       # remove all pkgs with libs
```

3. To check which packages are going to be installed, execute:

```
$ make ps
```

4. If you want to install an additional package, read the specific procedure in[2].

   After following the procedure described in the page, execute:

```
$ make yes-<LIB_NAME>
```

   Example for USER_COLVARS

```
$ make yes-user-colvars
```

   Verify again executing `make ps` that you have enabled correctly the installation of the desired packages.

   Warning: If you have an existing version of LAMMPS and you want to add a new package it's necessary to recompile LAMMPS.

5. Compile LAMMPS with it's packages. This procedure will compile it using the Intel architecture options defined by default in `src/MAKE/OPTIONS/Makefile.intel_cpu_intelmpi`

```
$ make intel_cpu_intelmpi
```

   Note: `icpc: command line warning #10006: ignoring unknown option '-qopt-zmm-usage=high'`

   This message appears when you compile LAMMPS using the intel_cpu_intelmpi architecture but the Intel processor doesn't have the AVX512 instruction set. If this is the case, just ignore the warning message. For more information about the flag `qopt-zmm-usage` read[3].

6. If you want to install LAMMPS in a specific directory, create the directories and copy the binary as follows:

```
$ mkdir -p <INSTALL_DIR>/bin
$ cp lmp_intel_cpu_intelmpi <INSTALL_DIR>/bin/
$ cd <INSTALL_DIR>/bin/
$ ln -s lmp_intel_cpu_intelmpi lammps
```

   Note: For more information about the installation process, read the official page[4].

[1] Download an executable for Linux, Pre-built Ubuntu Linux executables -LAMMPS documentation. Retrieved January 17, 2019, from https://lammps.sandia.gov/doc/Install_linux.html#ubuntu

[2] Include packages in build - LAMMPS documentation Retrieved June 10, 2019, from https://lammps.sandia.gov/doc/Build_package.html

[3] Intel® C++ Compiler 19.0 Developer Guide and Reference, qopt-zmm-usage, Qopt-zmm-usage. Retrieved January 17, 2019, from https://software.intel.com/en-us/cpp-compiler-developer-guide-and-reference-qopt-zmm-usage-qopt-zmm-usage

[4] Download source via Git - LAMMPS documentation. Retrieved January 17, 2019, from https://lammps.sandia.gov/doc/Install_git.html

7. Finally, if the program will be used with Environment modules, create the respective module.

Alternative Installation modes

- If you want to compile LAMMPS as a static library called `liblammps_machine.a`, then execute:

```
$ make mode=lib <machine>
```

- If you want to compile LAMMPS as a shared library called `liblammps_machine.so`, then execute:

```
$ make mode=shlib <machine>
```

Test LAMMPS

After installing LAMMPS, run the benchmarks present in the repository.

```
$ sbatch example.sh
```

The following code is an example for running LAMMPS using SLURM:

```
#!/bin/bash
#SBATCH --job-name=LAMMPS_Bench
#SBATCH --nodes=1
#SBATCH --ntasks=16
#SBATCH --partition=longjobs
#SBATCH --time=01:00:00
#SBATCH -o results_%j.out
#SBATCH -e results_%j.err

export OMP_NUM_THREADS=1

module load lammps

export WDIR=<REPO_DIR>/bench

srun --mpi=pmi2 lammps -in $WDIR/in.lj
srun --mpi=pmi2 lammps -in $WDIR/in.chain
srun --mpi=pmi2 lammps -in $WDIR/in.eam
srun --mpi=pmi2 lammps -in $WDIR/in.chute
srun --mpi=pmi2 lammps -in $WDIR/in.rhodo
```

Modulefile

Apolo II

Listing 40: `Module file`

```
#%Module1.0#####################################################################
##
## module load lammps/22Aug18_impi-2017_update-1
##
## /share/apps/modules/lammps/22Aug18_impi-2017_update-1
## Written by Andres Felipe Zapata Palacio
```

(continues on next page)

(continued from previous page)

```
##

proc ModulesHelp {} {
     global version modroot
     puts stderr "Sets the environment for using lammps 22Aug18\
                  \nin the shared directory \
                  \n/share/apps/lammps/22Aug18/impi-2017_update-1/\
                  \nbuilded with impi-2017_update-1"
}

module-whatis "(Name________) lammps"
module-whatis "(Version_____) 22Aug18"
module-whatis "(Compilers___) impi-2017_update-1"
module-whatis "(System______) x86_64-redhat-linux"
module-whatis "(Libraries___) mkl-2017_update-1"

# for Tcl script use only
set         topdir        /share/apps/lammps/22Aug18/impi-2017_update-1/
set         version       22Aug18
set         sys           x86_64-redhat-linux

conflict lammps
module load impi/2017_update-1
module load mkl/2017_update-1

setenv               LAMMPSROOT               $topdir

prepend-path        PATH                      $topdir/bin
```

Cronos

Listing 41: `Module file`

```
#%Module1.0#####################################################################
##
## module load lammps/22Aug18_impi-18.0.2
##
## /share/apps/modules/lammps/22Aug18_impi-18.0.2
## Written by Andres Felipe Zapata Palacio
##

proc ModulesHelp {} {
     global version modroot
     puts stderr "Sets the environment for using lammps 22Aug18\
                  \nin the shared directory \
                  \n/share/apps/lammps/22Aug18/impi-18.0.2/\
                  \nbuilded with impi-18.0.2"
}

module-whatis "(Name________) lammps"
module-whatis "(Version_____) 22Aug18"
module-whatis "(Compilers___) impi-18.0.2"
module-whatis "(System______) x86_64-redhat-linux"
module-whatis "(Libraries___) mkl-18.0.2"

# for Tcl script use only
set         topdir        /share/apps/lammps/22Aug18/impi-18.0.2/
```

(continues on next page)

(continued from previous page)

```
set         version       22Aug18
set         sys           x86_64-redhat-linux

conflict lammps
module load impi/18.0.2
module load mkl/18.0.2

setenv                LAMMPSROOT                $topdir

prepend-path        PATH                        $topdir/bin
```

References

Authors

- Andrés Felipe Zapata Palacio <azapat47@eafit.edu.co>

3.3.34 leDock

Description

LeDock is designed for fast and accurate flexible docking of small molecules into a protein. It achieves a pose-prediction accuracy of greater than 90% on the Astex diversity set and takes about 3 seconds per run for a drug-like molecule. It has led to the discovery of novel kinase inhibitors and bromodomain antagonists from high-throughput virtual screening campaigns. It directly uses SYBYL Mol2 format as input for small molecules.

This software doesn't have an official nomination of its versions.

For additional information you can open those links:

- http://www.lephar.com/software.htm
- http://www.lephar.com/download.htm

Table of Contents

LeDock 1.0

- **Installation date:** 19/10/2017
- **URL:** http://www.lephar.com/
- **Apolo version:** Apolo II
- **License:** Not found

Dependencies

- GNU GCC >= 4.4.7
- glibc >= 2.14
- patchelf >= 0.9

Installation

Load the needed modules or add the to the path

1. Download the binaries

```
$ wget http://www.lephar.com/download/ledock_linux_x86
```

2. create the directories

```
$ mkdir -p /share/apps/ledock/1.0/bin
$ mv ledock_linux_x86 /share/apps/ledock/1.0/bin/ledock
```

3. execute permissions

```
$ chmod +x /share/apps/lepro/2013/bin/ledock
```

Patch of the binary

```
$ patchelf --set-interpreter /share/apps/glibc/2.20/lib/ld-linux-x86-64.so.2 /share/
↪apps/ledock/1.0/bin/ledock
```

Module

```
##%Module1.0#####################################################################\
######
##
## module load ledock/1.0
##
## /share/apps/modules/ledock/1.0
## Written by Juan David Arcila Moreno
##

proc ModulesHelp {} {
```

(continues on next page)

(continued from previous page)

```
     global version modroot
     puts stderr "Sets the environment for using ledock 1.0\
                  \nin the shared directory \
                  \n/share/apps/ledock/1.0/\
                  \nbuilded with gcc-5.4.0\
                  \nledock needs glibc>=2.14"
}

module-whatis "(Name_________) ledock"
module-whatis "(Version______) 1.0"
module-whatis "(Compilers____) gcc-4.4.7"
module-whatis "(System_______) x86_64-redhat-linux"
module-whatis "(Libraries____) glibc-2.14"

# for Tcl script use only
set         topdir        /share/apps/ledock/1.0/
set         version       1.0
set         sys           x86_64-redhat-linux

conflict ledock


prepend-path    PATH                    $topdir/bin
```

Use mode

```
$ module load ledock/1.0
$ ledock config.file    !docking
$ ledock -spli dok.file !split into separate coordinates
```

References

- http://www.lephar.com/
- https://nixos.org/patchelf.html/

Author

- Juan David Arcila-Moreno

3.3.35 lePro

Description

LePro is designed to automatically add hydrogen atoms to the protein by explicitely considering the protonation state of histidine. All crystal water, ions, small ligands and cofactors were removed. It also generates a docking input file for LeDock with the binding site determined as to include any protein atom within 0.4 nm of any atom of the first complexed small molecule.

This software doesn't have an official nomination of its versions.

For additional information you can open those links:

- http://www.lephar.com/software.htm
- http://www.lephar.com/download.htm

Table of Contents

LEPRO 2013

- **Installation date:** 12/10/2017
- **URL:** http://www.lephar.com/
- **Apolo version:** Apolo II
- **License:** Not found

Dependencies

- GNU GCC >= 4.4.7

Installation

Load the needed modules or add the to the path

1. Download the repository

```
$ wget http://www.lephar.com/download/lepro_linux_x86
```

2. We create the installation folder and move the binary

```
$ mkdir -p /share/apps/lepro/2013/bin
$ mv lepro_linux_x86 /share/apps/lepro/2013/bin/lepro
```

3. execute permissions

```
$ chmod +x /share/apps/lepro/2013/bin/lepro
```

Module

```
#%Module1.0#####################################################################
##
## module load lepro/2013
##
## /share/apps/modules/lepro/2013
## Written by Juan David Arcila Moreno
##

proc ModulesHelp {} {
     global version modroot
     puts stderr "Sets the environment for using lepro 2013\
                  \nin the shared directory \
                  \n/share/apps/lepro/2013/\
                  \nDownload form http://www.lephar.com/download.htm"
}

module-whatis "(Name_________) lepro"
module-whatis "(Version_____) 2013"
module-whatis "(Compilers___) "
module-whatis "(System______) x86_64-redhat-linux"
module-whatis "(Libraries___) "

# for Tcl script use only
set         topdir        /share/apps/lepro/2013/
set         version       2013
set         sys           x86_64-redhat-linux

conflict lepro


prepend-path    PATH                    $topdir/bin
```

Use mode

```
$ module load lepro/2013
$ lepro [PDB file]
```

References

- http://www.lephar.com/

Author

- Juan David Arcila-Moreno

3.3.36 LINKS

LINKS[1] Long Interval Nucleotide K-mer Scaffolder.

[1] LINKS. (2019, Oct 8). Retrieved December 2, 2019, from https://github.com/bcgsc/LINKS

LINKS 1.8.7

Basic information

- **Deploy date:** 22 October 2019
- **Official Website:** https://github.com/bcgsc/abyss
- **License:** GNU GPL v3
- **Installed on:** *Apolo II*
- **Dependencies:**
  - GCC
  - Perl

Installation

1. Make sure you have GCC and Perl installed, then run:

```
$ git clone https://github.com/bcgsc/LINKS.git
$ cd LINKS/releases/links_v1.8.7
$ tar xvf lib.tar.gz
$ cd lib/
$ rm -rf bloomfilter
$ git clone git://github.com/bcgsc/bloomfilter.git
$ cd bloomfilter/swig
$ module load gcc/5.4.0
$ module load perl/5.26.1_gcc-4.4.7
$ swig -Wall -c++ -perl5 BloomFilter.i
$ g++ -std=c++11 -c BloomFilter_wrap.cxx -I/share/apps/perl/5.26.1_gcc-4.
↪4.7/lib/5.26.1/x86_64-linux-thread-multi-ld/CORE -fPIC -Dbool=char -O3 -
↪mavx2
$ g++ -std=c++11 -Wall -shared BloomFilter_wrap.o -o BloomFilter.so -O3 -
↪mavx2
$ ./test.pl
```

   Make sure the tests run correctly. Then install it:

```
# Go back to the links 1.8.7 root directory
$ cd ../../../
$ rm lib.tar.gz

$ mkdir -p /share/apps/links/1.8.7/gcc/5.4.0

# There is no installation script, so install it manually
$ cp -r ./* /share/apps/links/1.8.7/gcc/5.4.0/
```

2. Create and place the needed module file. Create a file with the following content:

Listing 42: `1.8.7_gcc-5.4.0_perl-5.26.1`

```
##%Module1.0#####################################################################
↪#####
##
## module load links/1.8.7
```

(continues on next page)

(continued from previous page)

```
##
## /share/apps/links/1.8.7/gcc/5.4.0
## Written by Vincent A. Arcila L and Hamilton Tobon Mosquera.
##

proc ModulesHelp {} {
        global version modroot
        puts stderr "Sets the environment for using links 1.8.7\
        \nin the shared directory /share/apps/links/1.8.7/gcc/5.4.0\
        \nbuilt with GCC 5.4.0.
}

module-whatis "(Name________) links"
module-whatis "(Version_____) 1.8.7"
module-whatis "(Compilers___) gcc-5.4.0"
module-whatis "(System______) x86_64-redhat-linux"
module-whatis "(Libraries___) "

# for Tcl script use only
set         topdir        /share/apps/links/1.8.7/gcc/5.4.0
set         version       1.8.7
set         sys           x86_64-redhat-linux

conflict links
module load perl/5.26.1_gcc-4.4.7

prepend-path   PATH            $topdir
prepend-path   PATH            $topdir/test
prepend-path   PATH            $topdir/tools
```

Create the needed folder and place it:

```
$ sudo mkdir /share/apps/modules/links
$ sudo mv 1.8.7_gcc-5.4.0_perl-5.26.1 /share/apps/modules/links/
```

Troubleshooting

There is a problem compiling with GCC 7.4.0, it seems like a problem with some instructions that are not supported by the C++ standard that GCC 7.4.0 implements. So, try using GCC 5.4.0.

Authors

- Hamilton Tobon-Mosquera <htobonm@eafit.edu.co>

3.3.37 MAFFT

MAFFT[1]. (for multiple alignment using fast Fourier transform) is a program used to create multiple sequence alignments of amino acid or nucleotide sequences.

[1] Wikipedia contributors. (2018, July 9). MAFFT. In Wikipedia, The Free Encyclopedia. Retrieved 07:37, July 27, 2018, from https://en.wikipedia.org/wiki/MAFFT

Published in 2002, the first version of MAFFT used an algorithm based on progressive alignment, in which the sequences were clustered with the help of the Fast Fourier Transform.[1] Subsequent versions of MAFFT have added other algorithms and modes of operation, including options for faster alignment of large numbers of sequences, higher accuracy alignments, alignment of non-coding RNA sequences, and the addition of new sequences to existing alignments.

MAFFT-7.402

Basic Information

- **Deploy date:** 27 July 2018
- **Official Website:** https://mafft.cbrc.jp/alignment/software/
- **License:** BSD License
  - **Extensions:**
    - **Mxcarna:** BSD License (For more information check README file in `mafft-7.402-with-extensions/extensions/mxscarna_src/`
    - **Foldalign:** GNU GPL-2
    - **Contrafold:** BSD License
- **Installed on:** *Apolo II*, *Cronos*

Installation

This entry covers the entire process performed for the installation and configuration of MAFFT with extensions (MXSCARNA, Foldalign and CONTRAfold) on a cluster.

Contents

Tested on (Requirements)

- **OS base:** CentOS (x86_64) $\geq$ 6.6
- **Compiler:** Intel Parallel Studio XE Compiler Cluster Edition $\geq$ 17.0.1.
- **Optional extensions:**
  - Mxscarna (Included in the mafft-7.402-with-extensions-src.tgz package)
  - Foldalign $\geq$ 2.5.1
  - Contrafold $\geq$ 2.02

Build process

Note:

- In *Apolo II* was used the Intel Compiler 17.0.1.
  - `module load intel/2017_Update-1`
- In *Cronos* was used the Intel Compiler 18.0.2.
  - `module load intel/18.0.2`

This entry described the installation process of MAFFT with extensions.

1. Get the MAFFT with the extensions package.

```
wget https://mafft.cbrc.jp/alignment/software/mafft-7.402-with-extensions-src.tgz
tar xfvz mafft-7.402-with-extensions-src.tgz
```

2. Edit MAFFT's Makefile with the following lines.

   `mafft-7.402-with-extensions/core/Makefile`

   From:

```
PREFIX = /usr/local
...
CC = gcc
#CC = icc
CFLAGS = -03
...
```

   To:

```
PREFIX = /your/path
...
#CC = gcc
CC = icc
CFLAGS = -03 -fast
...
```

3. Load the necessary environment and build it.

```
module load intel/2017_update-1
make clean
make
make install
```

Extensions

This entry described the extension's installation process.

Mxscarna

MXSCARNA[1]. (Multiplex Stem Candidate Aligner for RNAs) is a multiple alignment tool for RNA sequences using progressive alignment based on the pairwise structural alignment algorithm of SCARNA. This software is fast enough for large scale analyses, while the accuracies of the alignments are better than or comparable with the existing algorithms which are computationally much more expensive in time and memory.

1. Edit the Mxcarna's Makefile with the following lines.

   `mafft-7.402-with-extensions/extensions/mxscarna_src/Makefile`

   `Makefile`

   From:

```
...
CXX = g++
...
```

   To:

```
...
CXX = icpc
...
```

2. Load the necessary environment and build it.

```
cd ../
module load intel/2017_update-1
make clean
make
make install
```

3. Move the binaries to libexec MAFFT directory.

```
cp mxscarna /you/path/to/maft/libexec/mafft/
```

Foldalign

FOLDALIGN[2]. an algorithm for local or global simultaneous folding and aligning two or more RNA sequences and is based on the Sankoffs algorithm (SIAM J. Appl. Math., 45:810-825, 1985). Foldalign can make pairwise local or global alignments and structure predictions. FoldalignM makes multiple global alignment and structure prediction.

1. Get the Foldalign package and move it to the MAFFT extension's directory.

```
wget https://rth.dk/resources/foldalign/software/foldalign.2.5.1.tgz
tar xfvz foldalign.2.5.1.tgz
cp foldalign mafft-7.402-with-extensions/extensions/
cd mafft-7.402-with-extensions/extensions/foldalign
```

2. Edit Foldalign's Makefile with the following lines.

   `mafft-7.402-with-extensions/src/mafft-7.402-with-extensions/extensions/foldalign/Makefile`

   `Makefile`

[1] MXSCARNA. (n.d.). Retrieved August 10, 2018, from https://www.ncrna.org/softwares/mxscarna/
[2] Foldalign: RNA Structure and Sequence Alignment. (n.d.). From https://rth.dk/resources/foldalign/

from:

```
...
cc = g++
...
```

To:

```
...
cc = icpc
...
```

3. Load the necessary environment and build it.

```
module load intel/2017_update-1
make clean
make
```

4. Move the binaries to libexec MAFFT directory.

```
cp bin/* /you/path/to/maft/libexec/mafft/
```

Contrafold

CONTRAfold[3]. is a novel secondary structure prediction method based on conditional log-linear models (CLLMs), a flexible class of probabilistic models that generalize upon SCFGs by using discriminative training and feature-rich scoring. By incorporating most of the features found in typical thermodynamic models, CONTRAfold achieves the highest single sequence prediction accuracies to date, outperforming currently available probabilistic and physics-based techniques.

1. Get the Contrafold package and move it to the MAFFT extension's directory.

```
wget http://contra.stanford.edu/contrafold/contrafold_v2_02.tar.gz
tar xfvz contrafold_v2_02
cp  contrafold_v2_02/contrafold mafft-7.402-with-extensions/extensions/
cd mafft-7.402-with-extensions/extensions
```

2. load the necessary environment and build it.

```
cd contrafold/src/
module load intel/2017_update-1
module load openmpi/1.8.8-x86_64_intel-2017_update-1
make clean
make intelmulti
```

3. Move the binaries to libexec MAFFT directory.

```
cp contrafold /you/path/to/maft/libexec/mafft/
```

Troubleshooting

When you try to compile contrafold, it prints:

[3] Do, C., & Marina, S. (n.d.). Contrafold: CONditional TRAining for RNA Secondary Structure Prediction. From http://contra.stanford.edu/contrafold/

```
perl MakeDefaults.pl contrafold.params.complementary contrafold.params.
↪noncomplementary contrafold.params.profile
g++ -O3 -DNDEBUG -W -pipe -Wundef -Winline --param large-function-growth=100000 -Wall␣
↪ -c Contrafold.cpp
In file included from LBFGS.hpp:52,
                 from InnerOptimizationWrapper.hpp:12,
                 from OptimizationWrapper.hpp:12,
                 from Contrafold.cpp:16:
LBFGS.ipp: En la instanciación de 'Real LBFGS<Real>::Minimize(std::vector<_Tp>&) [con␣
↪Real = double]':
OptimizationWrapper.ipp:260:9:   se requiere desde 'void OptimizationWrapper<RealT>
↪::LearnHyperparameters(std::vector<int>, std::vector<_Tp>&) [con RealT = double]'
Contrafold.cpp:451:9:   se requiere desde 'void RunTrainingMode(const Options&, const␣
↪std::vector<FileDescription>&) [con RealT = double]'
Contrafold.cpp:68:54:   se requiere desde aquí
LBFGS.ipp:110:33: error: 'DoLineSearch' no se declaró en este ámbito, y no se␣
↪encontraron declaraciones en la búsqueda dependiente de argumentos en el punto de␣
↪la instanciación [-fpermissive]
      Real step = DoLineSearch(x[k%2], f[k%2], g[k%2], d,
                  ~~~~~~~~~~~~^~~~~~~~~~~~~~~~~~~~~~~~~~~
                               x[(k+1)%2], f[(k+1)%2], g[(k+1)%2],
                               ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
                               Real(0), std::min(Real(10), MAX_STEP_NORM /␣
↪std::max(Real(1), Norm(d))));
                               ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
↪~~~~~~~~~~~~~~~~~
LBFGS.ipp:110:33: nota: no se encontraron declaraciones en la base dependiente␣
↪'LineSearch<double>' pur la búsqueda no calificada
LBFGS.ipp:110:33: nota: use 'this->DoLineSearch' en su lugar
make: *** [Makefile:47: Contrafold.o] Error 1
```

Or something similar about a compilation error, it appears because in Utilities.hpp is missing an include.

1. Edit Utilities.hpp and add the *limits.h* library.

   `mafft-7.402-with-extensions/extensions/contrafold/src/Utilities.hpp`

   `Utilities.hpp`

   from:

   ```
   #define UTILITIES_HPP

   #include <algorithm>
   ...
   ```

   To:

   ```
   #define UTILITIES_HPP

   #include <limits.h>
   #include <algorithm>
   ...
   ```

2. Repeat step 2.

Module Files

Apolo II

Listing 43: 7.402-with-extensions_intel-17.0.1

```
#%Module1.0#####################################################################
##
## module load mafft/7.402-with-extensions_intel-17.0.1
##
## /share/apps/modules/mafft/7.402-with-extensions_intel-17.0.1
## Written by Manuela Carrasco Pinzon
##

proc ModulesHelp {} {
     global version modroot
     puts stderr "Sets the environment for using mafft 7.402-with-extensions\
                  \nin the shared directory /share/apps/mafft/7.402-with-extensions/
↪intel-17.0.1/\
                  \nbuilded with Intel Parallel Studio XE Cluster Edition 2017 Update␣
↪1."
}

module-whatis "(Name________) mafft"
module-whatis "(Version_____) 7.402-with-extensions"
module-whatis "(Compilers___) intel-17.0.1"
module-whatis "(System______) x86_64-redhat-linux"
module-whatis "(Libraries___) "

# for Tcl script use only
set         topdir        /share/apps/mafft/7.402-with-extensions/intel-17.0.1
set         version       7.402-with-extensions
set         sys           x86_64-redhat-linux

conflict mafft
module load intel/2017_update-1
module load openmpi/1.8.8-x86_64_intel-2017_update-1


prepend-path        PATH                        $topdir/bin
prepend-path        PATH                        $topdir/libexec

prepend-path        MANPATH                        $topdir/share/man
```

Cronos

Listing 44: 7.402-with-extensions_intel-18.0.2

```
#%Module1.0#####################################################################
##
## module load mafft/7.402-with-extensions_intel-18.0.2
##
## /share/apps/modules/mafft/7.402-with-extensions_intel-18.0.2
## Written by Manuela Carrasco Pinzon
##

proc ModulesHelp {} {
```

(continues on next page)

(continued from previous page)

```
    global version modroot
    puts stderr "Sets the environment for using mafft 7.402-with-extensions\
                 \nin the shared directory /share/apps/mafft/7.402-with-extensions/
↪intel-18.0.2/\
                 \nbuilded with Intel Parallel Studio XE Cluster Edition 2018."
}

module-whatis "(Name_________) mafft"
module-whatis "(Version______) 7.402-with-extensions"
module-whatis "(Compilers____) intel-18.0.2"
module-whatis "(System_______) x86_64-redhat-linux"
module-whatis "(Libraries____) "

# for Tcl script use only
set         topdir        /share/apps/mafft/7.402-with-extensions/intel-18.0.2
set         version       7.402-with-extensions
set         sys           x86_64-redhat-linux

conflict mafft
module load intel/18.0.2
module load openmpi/3.1.1_intel-18.0.2


prepend-path    PATH                    $topdir/bin
prepend-path    PATH                    $topdir/libexec

prepend-path    MANPATH                 $topdir/share/man
```

Usage

This section describes the method to submit jobs with the resource manager SLURM.

1. Load the necessary environment.

```
module load mafft/7.402-with-extensions_intel-X.X.X
```

Note:

Remember to load the proper environment for Cronos or Apolo

- Apolo
  - `module load mafft/7.402-with-extensions_intel-17.0.1`
- Cronos
  - `module load mafft/7.402-with-extensions_intel-18.0.2`

2. Run MAFFT with SLURM.

   An example:

Note: This example was tested with `example.fa` that contains unlined DNA secuences.

Listing 45: `mafft.sh`

```
#!/bin/bash

#SBATCH --partition=longjobs
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=32
#SBATCH --time=120:00:00
#SBATCH --job-name=test
#SBATCH -o result_%N_%j.out       # File to which STDOUT will be written
#SBATCH -e result_%N_%j.err       # File to which STDERR will be written
#SBATCH --mail-type=ALL
#SBATCH --mail-user=test@example.com

export SBATCH_EXPORT=NONE
export OMP_NUM_THREADS=1

mafft --quiet --auto --thread $SLURM_NTASKS example.fa > example_al.fa
```

Options

- `quiet` Do not report progress and this flag is mandatory to run unattended jobs.
- `auto` Automatically selects an appropriate strategy from L-INS-i, FFT-NS-i and FFT-NS-2, according to data size.
- `thread` Number of threads (number of cores reserved on SLURM).

Note: For more information, please read the manual entry for Mafft `man mafft` or `mafft -h`

Authors

- Manuela Carrasco Pinzón <mcarras1@eafit.edu.co>

3.3.38 MDSTRESS-LIB

MdStress-lib[1] is a standalone C++ library that can be incorporated into any code for local stress calculations. It implements various flavors of the spatial Irving-Kirkwood-Noll stress including the central force decomposition as well as the virial stress per atom

MDSTRESS-LIB

Table of Contents

[1] MdStress-lib official documentation. Retrieved on June 16 of 2020 from https://www.mdstress.org/

Basic information

- **Official Website:** https://www.mdstress.org/index.php/mdstresslib/
- **Downloads page:** https://www.mdstress.org/index.php/downloads/
- **Installed on:** APOLO II

Tested on (Requirements)

- **OS base:** CentOS (x86_64) $\geq$ 8
- **Dependencies to run mdstress-lib:**
  - fftw
  - lapack

Installation

For the installation process we'll follow the guide on the gromacs documentation[1]

1. First of all, we need to load the following modules for the compilation

```
$ module load fftw/3.3.5_intel_impi-2017_update-1
$ module load cmake/3.7.1
$ module load lapack/3.6.1_gcc-4.9.4
```

2. Then download the tar.gz file and unpack it

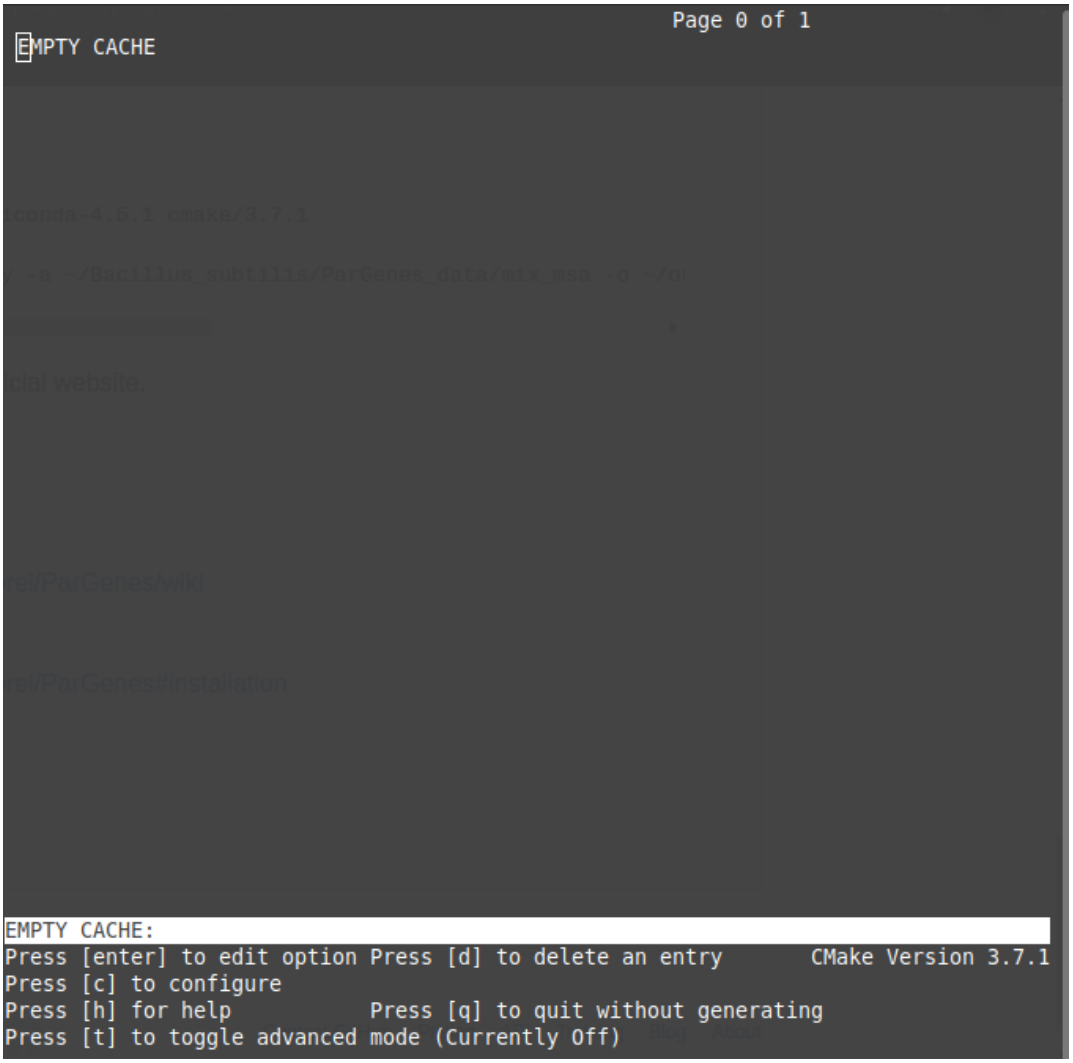
```
$ wget --trust-server-name https://www.mdstress.org/index.php/download_
↪file/view/50/173/
$ tar -xzvf mdstress-library-12282019.tar.gz
$ cd mdstress-library
```

3. Then we need to create a directory named "built", and then we run the cmake tool

```
$ mkdir build
$ cd build
$ ccmake ../
```

4. The interface of cmake will appear and you have to edit a few things

[1] Gromacs documentation, retrieved on May 18, 2020 from https://www.mdstress.org/index.php/documentation/



- Then press c to configure
- Set CMAKE_INSTALL_PREFIX to the installation dir
- Press c again and the it will appear the option to press g, press it and do **make** and **make install**

References

Author Manuela Herrera-López

3.3.39 MedeA

The MedeA modeling suite is the leading software package for atomistic-scale simulation of materials. MedeA is modular, and that the MedeA Environment is the most fundamental piece of MedeA into which all of the other modules "plug". All users need the MedeA Environment. Indeed the MedeA Environment is provided to users together with other essential components (InfoMaticA and JobServer/TaskServer) in the Welcome to MedeA Bundle which is required for all MedeA users.

The MedeA Environment includes a rich set of building and basic analysis capabilities which do not need to be licensed separately:

- Surface Builder
- Molecular Builder
- PolymerBuilder
- Nanoparticle and Nanotube Builders
- Substitutional Search
- Random Substitutions
- Special Quasirandom Structures (SQS)
- Layer Builder

In addition, Forcefields for classical simulations are assigned and managed within the MedeA Environment

2.20.4

Table of contents

- **Installation date:** 7/04/2017
- **URL:** http://www.materialsdesign.com/medea
- **Apolo version:** Apolo II
- **License:** Commercial (Payment)

Requirements

- mysql-client >= 5.1
- mysql-server >= 5.1
- libXScrnSaver
- User previously created in Apollo (i.e. mvelez)
- SSH without password between the machine that has the MedeA and Apollo GUI.

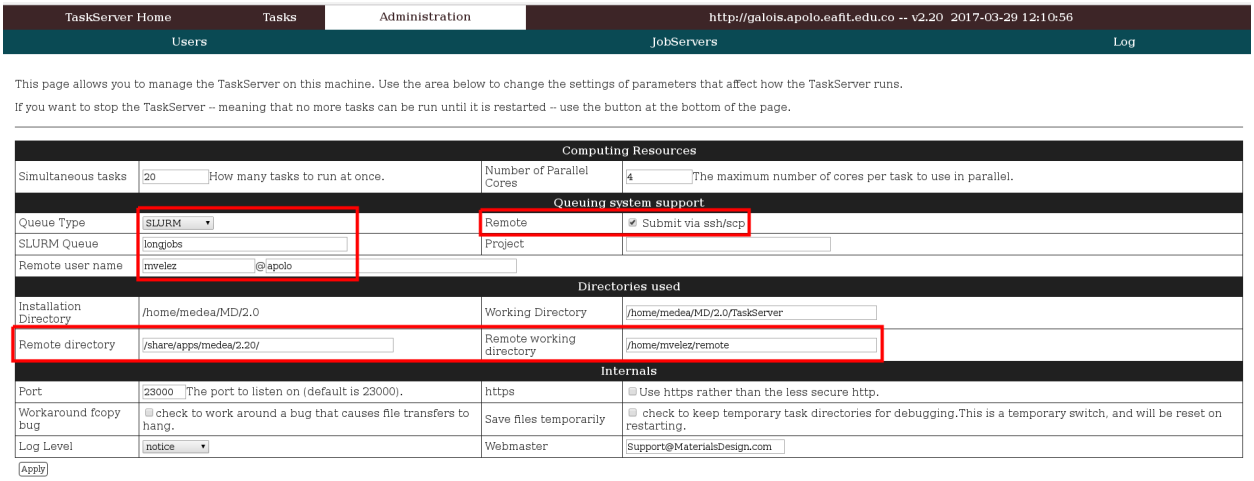
Steps:

- Packages

```
yum install glibc.i686 libstdc++i686 libX11.i686 libXft.i686 zlib.i686
yum install mysql-server mysql-client
```

- SSH without password

```
ssh-keygen
ssh-copy-id mvelez@apolo.eafit.edu.co
```



3.3.40 Mothur

Mothur[1] is an open-source software package for bioinformatics data processing. The package is frequently used in the analysis of DNA from uncultured microbes. Mothur is capable of processing data generated from several DNA sequencing methods including 454 pyrosequencing, Illumina HiSeq and MiSeq, Sanger, PacBio, and IonTorrent.

MOTHUR

Table of Contents

Basic information

- **Installation Date:** 05/09/2016
- **License:** GNU LESSER GENERAL PUBLIC LICENSE Version 3
- **Installed on:** Apolo II

[1] Wikipedia contributors. (2019, 6 noviembre). Mothur. Recuperado 18 noviembre, 2019, de https://en.wikipedia.org/wiki/Mothur

Tested on (Requirements)

- **Dependencies to run PLUMED:**
  - zlib y zlib-devel
  - bzip2 y bzip2-devel
  - boost >= 1.58.0

Installation

After solving the previously mentioned dependencies, you can proceed with the installation.

1. Download the latest version of the software (Source code - tgz) (https://github.com/mothur/mothur/releases):

```
cd ~/apps/mothur/src/
wget https://github.com/mothur/mothur/archive/v1.38.1.1.tar.gz
tar -zxvf v1.38.1.1.tar.gz
```

2. Makefile configuration, make the following changes:

```
cd mothur-1.38.1.1
emacs Makefile
BOOST_LIBRARY_DIR="\"/share/apps/boost-c++-lib/1.59.0/intel-15/lib\""
BOOST_INCLUDE_DIR="\"share/apps/boost-c++-lib/1.59.0/intel-15/include\""
MOTHUR_FILES="\"/home/mgomezzul/apps/mothur/src/mothur-1.38.1.1\""
```

Module

```
#%Module1.0#####################################################################
↪##
##
## module load plumed/2.3.5
##
## /share/apps/modules/plumed/2.3.5
## Written by Mateo Gómez-Zuluaga
##

proc ModulesHelp {} {
    global version modroot
    puts stderr "Sets the environment for using plumed 2.3.5\
        \nin the shared directory /share/apps/plumed/2.3.5\
        \nbuilded with gcc-5.4.0, mpich2-3.2, openblas-0.2.19\
        \ngsl-2.4 and libmatheval-1.11.0."
}

module-whatis "(Name________) plumed"
module-whatis "(Version_____) 2.3.5"
module-whatis "(Compilers___) gcc-5.4.0_mpich2-3.2"
module-whatis "(System______) x86_64-redhat-linux"
module-whatis "(Libraries___) opeblas-0.2.19, libmatheval-1.11.0, gsl-2.4"

# for Tcl script use only
set         topdir        /share/apps/plumed/2.3.5
```

(continues on next page)

(continued from previous page)

```
set         version       2.3.5
set         sys           x86_64-redhat-linux

conflict plumed

module load mpich2/3.2_gcc-5.4.0
module load openblas/0.2.19_gcc-5.4.0
module load gsl/2.4_gcc-5.4.0
module load libmatheval/1.1.11


prepend-path    PATH                    $topdir/bin

prepend-path    LD_LIBRARY_PATH         $topdir/lib
prepend-path    LIBRARY_PATH            $topdir/lib
prepend-path    LD_RUN_PATH             $topdir/lib
prepend-path    DYLD_LIBRARY_PATH       $topdir/lib

prepend-path    C_INCLUDE_PATH          $topdir/include
prepend-path    CXX_INCLLUDE_PATH       $topdir/include
prepend-path    CPLUS_INCLUDE_PATH      $topdir/include

setenv          PLUMED_KERNEL           $topdir/lib/libplumedKernel.so
```

Mode of Use

- The use of **Plumed** in this case is limited to patching the **Gromacs** source code for this MD to use **Plumed** for physical handling.

```
module load wrf/3.7.1_gcc-5.4.0
```

References

- http://www.plumed.org
- https://plumed.github.io/doc-v2.3/user-doc/html/_installation.html
- https://plumed.github.io/doc-v2.3/user-doc/html/gromacs-5-1-4.html
- http://www.jyhuang.idv.tw/JYH_ComputingPackages.html
- http://pdc-software-web.readthedocs.io/en/latest/software/plumed/centos7/2.3b/
- https://plumed.github.io/doc-v2.4/user-doc/html/_g_m_x_g_p_u.html

Authors

- Mateo Gómez Zuluaga

Mothur 1.42.3

Basic Information

- **Deploy date:** 24 January 2019
- **Official Website:** https://www.mothur.org
- **License:** GNU General Public License
- **Installed on:** *Apolo II*

Installation

This entry covers the entire process performed for the installation and configuration of Mothur.

Tested on (Requirements)

- **OS base:** CentOS (x86_64) $\geq$ 6.6
- bzip2
- bzip2-devel
- libz
- zlib-devel
- HDF5

Build process

1. Load the right environment.

```
module load hdf5
```

1. Download and decompress the Mothur's source code

```
wget https://github.com/mothur/mothur/archive/v.1.42.3.tar.gz

tar fxz v.1.42.3.tar.gz
```

1. Enter to Mothur makefile and edit the PREFIX path and HDF5 file paths.

```
cd mothur-v.1.42.3
emacs Makefile
```

```
PREFIX := "/path/to/install/mothur"
...
HDF5_LIBRARY_DIR ?= "/path/to/hdf5/lib"
HDF5_INCLUDE_DIR ?= "/path/to/hdf5/include"
```

1. Compile Mothur

```
make
```

Module Files

Apolo II

Listing 46: `Mothur-1.42.3`

```
##%Module1.0#####################################################################
##
## modules mothur/1.42.3_intel-19.0.4
##
## /share/apps/mothur/1.42.3/intel-19.0.4_boost-1.67.0 Written by Johan Sebastian␣
↪Yepes Rios
##

proc ModulesHelp { } {
    puts stderr "\tmothur/1.42.3_intel-19.0.4 - sets the Environment for mothur in \
    \n\tthe share directory /share/apps/mothur/1.42.3/intel-19.0.4_boost-1.67.0\n"
    }

module-whatis "\n\n\tSets the environment for using Mothur-1.42.3 \
               \n\tbuilded with Intel Compiler 2019 and Boost 1.67.0 version\n"

# for Tcl script use only
set       topdir     /share/apps/mothur/1.42.3/intel-19.0.4_boost-1.67.0
set       version    1.42.3
set       sys        x86_64-redhat-linux

module load intel/19.0.4 gcc/5.4.0 boost/1.67.0_intel-17.0.1 hdf5/1.10.5_gcc-5.4.0

prepend-path    PATH $topdir/bin
```

Usage

This section describes the method to submit jobs with the resource manager SLURM.

1. Load the necessary environment.

```
module load mothur/1.42.1_intel-2017_update-1
```

2. Run Mothur with SLURM.

   An example with AmazonData.zip:

Listing 47: `mothur.sh`

```
#!/bin/bash
↪

#SBATCH --job-name=serial_test       # Job name
↪
#SBATCH --mail-type=ALL          # Mail notification
↪
#SBATCH --mail-user=<user>@<domain>  # User Email
↪
#SBATCH --output=mothur-%j.out # Stdout (%j expands to jobId)
↪
```

(continues on next page)

(continued from previous page)

```
#SBATCH --error=mothur-%j.err  # Stderr (%j expands to jobId)
#SBATCH --ntasks=1                  # Number of tasks (processes)
#SBATCH --time=01:00                # Walltime
#SBATCH --partition=longjobs        # Partition


##### ENVIRONMENT CREATION #####

module load mothur/1.42.1_intel-2017_update-1

##### JOB COMMANDS ####

mothur "#read.dist(phylip=98_sq_phylip_amazon.dist, cutoff=0.1);
cluster(); collect.single(); rarefaction.single()"
```

Authors

- Manuela Carrasco Pinzón <mcarras1@eafit.edu.co>

3.3.41 MPICH

MPICH 3.3.2

Table of Contents

Basic information

- **Official Website:** https://www.mpich.org/
- **Downloads page:** https://www.mpich.org/downloads
- **Installed on:** APOLO II

Tested on (Requirements)

- **OS base:** CentOS (x86_64) $\geq$ 8
- **Compiler** Intel 19.0.4

Installation

1. First of all, we need to load the following modules for the compilation

```
$ module load intel/19.0.4
```

2. Then download the tar.gz file and unpack it

```
$ wget http://www.mpich.org/static/downloads/3.3.2/mpich-3.3.2.tar.gz
$ tar -xvf mpich-3.3.2.tar.gz
$ cd mpich-3.3.2
```

3. Then we can continue with the installation

```
$ unset F90
$ unset F90FLAGS
$ ./configure CC=/share/apps/intel/ps_xe/2019_update-4/compilers_and_
↪libraries_2019.4.243/linux/bin/intel64/icc FC=/share/apps/intel/ps_xe/
↪2019_update-4/compilers_and_libraries_2019.4.243/linux/bin/intel64/
↪ifort F77=/share/apps/intel/ps_xe/2019_update-4/compilers_and_libraries_
↪2019.4.243/linux/bin/intel64/ifort CXX=/share/apps/intel/ps_xe/2019_
↪update-4/compilers_and_libraries_2019.4.243/linux/bin/intel64/icpc --
↪prefix=/share/apps/mpich2/3.3.2/intel-19.0.4 --build=x86_64-redhat-
↪linux --enable-cxx --enable-fortran=all --enable-threads=multiple --
↪with-pm=hydra --with-thread-package=posix --with-mxm=/opt/mellanox/mxm -
↪-with-slurm=/opt/slurm/18.08.1
$ make
$ sudo make install
```

Note: The configure can vary on the environment you are working on. In Apolo the only way to make it function is to burn the environmental variables directly in the configure. Also you may need to unset F90 and F90FLAGS variables depending in your enviroment.

Author

- Tomas David Navarro
- Santigo Alzate Cardona

MPICH 3.4.2

Table of Contents

Basic information

- **Official Website:** https://www.mpich.org/
- **Downloads page:** https://www.mpich.org/downloads
- **Installed on:** APOLO II

Tested on (Requirements)

- **OS base:** Rocky Linux 8.5 (x86_64) $\geq$ 8
- **Compiler** GCC 9.3.0

Note: This version of MPICH is recommended for WRFCMAQ

Installation

1. First of all, we need to load the following modules for the compilation

```
$ module load intel/2022_oneAPI-update1
```

2. Then download the tar.gz file and unpack it

```
$ wget https://www.mpich.org/static/downloads/3.4.2/mpich-3.4.2.tar.gz
$ tar -xvf mpich-3.4.2.tar.gz
$ cd mpich-3.4.2
```

3. Then we can continue with the installation

```
$ unset F90
$ unset F90FLAGS
$ ./configure CC=/share/apps/gcc/9.3.0/bin/gcc FC=/share/apps/gcc/9.3.0/
↪bin/gfortran F77=/share/apps/gcc/9.3.0/bin/gfortran --prefix=/share/
↪apps/mpich/3.4.2/gcc-9.3.0 --with-device=ch3
$ make
$ sudo make install
```

Note: The configure can vary on the environment you are working on. In Apolo the only way to make it function is to burn the environmental variables directly in the configure. Also you may need to unset F90 and F90FLAGS variables depending in your enviroment.

Author

- Bryan López Parra <blopezp@eafit.edu.co>

3.3.42 mrBayes

Description

MrBayes is a program for Bayesian inference and model choice across a wide range of phylogenetic and evolutionary models. MrBayes uses Markov chain Monte Carlo (MCMC) methods to estimate the posterior distribution of model parameters.

Program features include:

A common command-line interface across Macintosh, Windows, and UNIX operating systems; Extensive help available from the command line; Analysis of nucleotide, amino acid, restriction site, and morphological data; Mixing of data types, such as molecular and morphological characters, in a single analysis; Easy linking and unlinking of parameters across data partitions; An abundance of evolutionary models, including 4x4, doublet, and codon models for nucleotide data and many of the standard rate matrices for amino acid data; Estimation of positively selected sites in a fully hierarchical Bayesian framework; Full integration of the BEST algorithms for the multi-species coalescent. Support for complex combinations of positive, negative, and backbone constraints on topologies; Model jumping across the GTR model space and across fixed rate matrices for amino acid data; Monitoring of convergence during the analysis, and access to a wide range of convergence diagnostics tools after the analysis has finished; Rich summaries of posterior samples of branch and node parameters printed to majority rule consensus trees in FigTree format; Implementation of the stepping-stone method for accurate estimation of model likelihoods for Bayesian model choice using Bayes factors; The ability to spread jobs over a cluster of computers using MPI (for Macintosh (OS X) and UNIX environments only); Support for the BEAGLE library, resulting in dramatic speedups for codon and amino acid models on compatible hardware (NVIDIA graphics cards); Checkpointing across all models, allowing the user to seemlessly extend a previous analysis or recover from a system crash;

Table of Contents

MrBayes 3.2.6

- **Installation date:** 08/02/2017
- **URL:** http://mrbayes.sourceforge.net/index.php
- **Apolo version:** Apolo II
- **License:** GNU GENERAL PUBLIC LICENSE Version 3

Dependencies

- Beagle (CUDA or CPU version)

- Intel compiler (C y C++)
- Intel MPI (C y C++)

Installation

1. First download the tar from the main page

```
wget https://downloads.sourceforge.net/project/mrbayes/mrbayes/3.2.6/mrbayes-3.2.6.
↪tar.gz?r=http%3A%2F%2Fmrbayes.sourceforge.net%2Fdownload.php&ts=1486584181&use_
↪mirror=superb-dca2
tar -zxvf mrbayes-3.2.6.tar.gz
```

1. Makefile config

```
cd mrbayes-3.2.6
module load beagle-lib/2.1.2_cuda-7.0_intel-2017_update-1
module load impi/2017_update-1
./configure --prefix=/share/apps/mrbayes/3.2.6/cuda/7.0/intel_impi/2017_update-1 --
↪enable-mpi --enable-sse --with-beagle=/share/apps/beagle-lib/2.1.2/cuda/7.0/intel/
↪2017_update-1 2>&1 | tee mrbayes-conf.log
make 2>&1 | tee mrbayes-make.log
```

then

```
sudo mkdir -p /share/apps/mrbayes/3.2.6/cuda/7.0/intel_impi/2017_update-1
make install 2>&1 | tee mrbayes-make-install.log
```

1. CUDA SUPPORT

```
cd /home/mgomezzul/apps/garli/intel
wget https://downloads.sourceforge.net/project/mrbayes/mrbayes/3.2.6/mrbayes-3.2.6.
↪tar.gz?r=http%3A%2F%2Fmrbayes.sourceforge.net%2Fdownload.php&ts=1486584181&use_
↪mirror=superb-dca2
tar -zxvf mrbayes-3.2.6.tar.gz
```

1. edit makefile

```
cd mrbayes-3.2.6
module load beagle-lib/2.1.2_intel-2017_update-1
module load impi/2017_update-1
./configure --prefix=/share/apps/mrbayes/3.2.6/intel_impi/2017_update-1 --enable-mpi -
↪-enable-sse --with-beagle=/share/apps/beagle-lib/2.1.2/intel/2017_update-1 2>&1 |␣
↪tee mrbayes-conf.log
make 2>&1 | tee mrbayes-make.log
```

Module

CUDA

```
#%Module1.0#####################################################################
##
## module mrbayes/3.2.6_cuda-7.0_intel-2017_update-1
##
## /share/apps/modules/mrbayes/3.2.6_cuda-7.0_intel-2017_update-1     Written by␣
↪Mateo Gomez-Zuluaga
```

(continues on next page)

(continued from previous page)

```
##

proc ModulesHelp { } {
    puts stderr "\tzlib/1.2.11 - sets the Environment for MrBayes 3.2.6 in \
    \n\tthe share directory /share/apps/mrbayes/3.2.6/cuda/7.0/intel/2017_update-1\n"
}

module-whatis "\n\n\tSets the environment for using MrBayes 3.2.6 \
               \n\tbuilded with Intel Parallel Studio XE 2017 and CUDA 7.0\n"

# for Tcl script use only
set       topdir     /share/apps/mrbayes/3.2.6/cuda/7.0/intel_impi/2017_update-1
set       version    3.2.6
set       sys        x86_64-redhat-linux

module load beagle-lib/2.1.2_cuda-7.0_intel-2017_update-1
module load impi/2017_update-1

prepend-path PATH    $topdir/bin
```

CPU

```
#%Module1.0#####################################################################
##
## module mrbayes/3.2.6_intel-2017_update-1
##
## /share/apps/modules/mrbayes/3.2.6_intel-2017_update-1     Written by Mateo Gomez-
↪Zuluaga
##

proc ModulesHelp { } {
    puts stderr "\tmrbayes/3.2.6_intel-2017_update-1 - sets the Environment for␣
↪MrBayes in \
    \n\tthe share directory /share/apps/mrbayes/3.2.6/intel/2017_update-1\n"
}

module-whatis "\n\n\tSets the environment for using MrBayes 3.2.6 \
               \n\tbuilded with Intel Parallel Studio XE 2017\n"

# for Tcl script use only
set       topdir     /share/apps/mrbayes/3.2.6/cuda/7.0/intel_impi/2017_update-1
set       version    3.2.6
set       sys        x86_64-redhat-linux

module load beagle-lib/2.1.2_intel-2017_update-1
module load impi/2017_update-1

prepend-path PATH    $topdir/bin
```

usage mode

CUDA

```
#!/bin/bash
#SBATCH --partition=accel
```

(continues on next page)

(continued from previous page)

```
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=1
#SBATCH --gres=gpu:2
#SBATCH --time=1:00:00
#SBATCH --job-name=mrbayes_gpu
#SBATCH -o result_%N_%j.out
#SBATCH -e result_%N_%j.err

export SBATCH_EXPORT=NONE
export OMP_NUM_THREADS=1

module load mrbayes/3.2.6_cuda-7.0_intel_impi-2017_update-1

mb concat_prot_corrected.nexus
```

CPU

```
#!/bin/bash
#SBATCH --partition=bigmem
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=24
#SBATCH --time=1:00:00
#SBATCH --job-name=mrbayes_cpu
#SBATCH -o result_%N_%j.out
#SBATCH -e result_%N_%j.err

export SBATCH_EXPORT=NONE
export OMP_NUM_THREADS=1

module load garli/2.01_intel_impi-2017_update-1

srun -n $SLURM_NTASKS mb concat_prot_corrected.nexus
```

References

- manual

Author

- Mateo Gómez Zuluaga

MrBayes 3.2.7

Table of Contents

- **Installation date:** 23/02/2021
- **URL:** http://mrbayes.sourceforge.net/index.php
- **Apolo version:** Apolo II
- **License:** GNU GENERAL PUBLIC LICENSE Version 3

Dependencies

- Beagle (CUDA or CPU version)
- Intel compiler (C y C++)
- Intel OPENMPI (C y C++)

Installation

1. First download the tar from the main page

```
mkdir mrbayes
cd /home/blopezp/Desktop/mrbayes
wget https://github.com/NBISweden/MrBayes/releases/download/v3.2.7/mrbayes-3.2.7.tar.
↪gz
tar -xzvf mrbayes-3.2.7.tar.gz
cd mrbayes-3.2.7
```

2. Makefile config and CUDA support

```
mkdir build
cd build
module load openmpi/3.1.5_intel-19.0.4
module load beagle-lib/2.1.2_cuda-7.0_intel-2017_update-1
module load gcc/7.4.0
../configure --prefix=/share/apps/mrbayes/3.2.7/cuda/7.0/gcc/7.4.0 --with-mpi=/share/
↪apps/openmpi/3.1.5/intel-19.0.4 --enable-sse --with-beagle=/share/apps/beagle-lib/2.
↪1.2/cuda/7.0/intel2017_update-1 2>&1 | tee mrbayes-conf.log
make 2>&1 | tee mrbayes-make.log
sudo make install
```

3. Makefile config and CUDA no support

```
module unload beagle-lib/2.1.2_cuda-7.0_intel-2017_update-1
module unload gcc/7.4.0

module load openmpi/3.1.5_intel-19.0.4
module load beagle-lib/2.1.2_intel-2017_update-1
module load gcc/7.4.0
../configure --prefix=/share/apps/mrbayes/3.2.7/gcc/7.4.0 --with-mpi=/share/apps/
↪openmpi/3.1.5/intel-19.0.4 --enable-sse --with-beagle=/share/apps/beagle-lib/2.1.2/
↪intel/2017_update-1 2>&1 | tee mrbayes-conf.log
make 2>&1 | tee mrbayes-make.log
sudo make install
```

Module

CUDA

```
##%Module1.0#####################################################################
##
## module mrbayes/3.2.7_cuda-7.0_gcc-7.4.0
##
## /share/apps/modules/mrbayes/3.2.7_cuda-7.0_gcc-7.4.0    Written by Bryan López␣
↪Parra
##

proc ModulesHelp { } {
    puts stderr "\tzlib/1.2.11 - sets the Environment for MrBayes 3.2.7 in \
    \n\tthe share directory /share/apps/mrbayes/3.2.7_cuda-7.0_gcc-7.4.0\n"
}

module-whatis "\n\n\tSets the environment for using MrBayes 3.2.7 \
                \n\tbuilded with CUDA 7.0 AND gcc-7.4.0\n"

# for Tcl script use only
set       topdir     /share/apps/mrbayes/3.2.7_cuda-7.0_gcc-7.4.0
set       version    3.2.7
set       sys        x86_64-redhat-linux

module load beagle-lib/2.1.2_cuda-7.0_intel-2017_update-1
module load openmpi/3.1.5_intel-19.0.4

prepend-path PATH    $topdir/bin
```

CPU

```
##%Module1.0#####################################################################
##
## module mrbayes/3.2.7_gcc-7.4.0
##
## /share/apps/modules/mrbayes/3.2.7_gcc-7.4.0    Written by Bryan López Parra
##

proc ModulesHelp { } {
    puts stderr "\tmrbayes/3.2.7_gcc-7.4.0 - sets the Eviroment for MrBayes in \
    \n\tthe share directory /share/apps/mrbayes/3.2.7_gcc-7.4.0\n"
}

module-whatis "\n\n\tSets the environment for using MrBayes 3.2.7 \
               \n\tbuilded with gcc-7.4.0\n"

# for Tcl script use only
set       topdir     /share/apps/mrbayes/3.2.7_gcc-7.4.0
set       version    3.2.7
set       sys        x86_64-redhat-linux

module load beagle-lib/2.1.2_intel-2017_update-1
module load openmpi/3.1.5_intel-19.0.4

prepend-path PATH    $topdir/bin
```

Slurm template

CUDA

```
#!/bin/bash
#SBATCH --partition=accel
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=1
#SBATCH --gres=gpu:2
#SBATCH --time=1:00:00
#SBATCH --job-name=mrbayes_gpu
#SBATCH -o result_%N_%j.out
#SBATCH -e result_%N_%j.err

export SBATCH_EXPORT=NONE
export OMP_NUM_THREADS=1

module load mrbayes/3.2.7_cuda-7.0_gcc-7.4.0

mpirun -np 1 mb primates-gtr-gamma.nex
```

CPU

```
#!/bin/bash
#SBATCH --partition=bigmem
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=24
#SBATCH --time=1:00:00
#SBATCH --job-name=mrbayes_cpu
#SBATCH -o result_%N_%j.out
#SBATCH -e result_%N_%j.err

export SBATCH_EXPORT=NONE
export OMP_NUM_THREADS=1

module load mrbayes/3.2.7_gcc-7.4.0

mpirun -np $SLURM_NTASKS mb primates-gtr-gamma.nex
```

References

- https://nbisweden.github.io/MrBayes/download.html

Author

- Bryan López Parra <blopezp@eafit.edu.co>

3.3.43 MUSCLE

MUSCLE is one of the best-performing multiple alignment programs according to published benchmark tests, with accuracy and speed that are consistently better than CLUSTALW. MUSCLE can align hundreds of sequences in seconds. Most users learn everything they need to know about MUSCLE in a few minutes—only a handful of command-line options are needed to perform common alignment tasks.

MUSCLE 3.8.1551

Table of Contents

Basic information

- **Official Website:** http://www.drive5.com/muscle/
- **License:** Open source
- **Installed on:** Apolo II and Cronos
- **Installation date:** 20/02/2017

Tested on (Requirements)

- **OS base:** CentOS (x86_64) $\geq$ 6.6 (Rocks 6.2)
- **Dependencies:**
  - g++ >= 4.9.4

Installation

1. Download the desired version of the software (Source code - tar.gz)

```
cd /home/mgomezzul/apps/muscle/gcc
http://www.drive5.com/muscle/downloads3.8.31/muscle3.8.31_src.tar.gz
tar -zxvf muscle3.8.31_src.tar.gz
```

2. Do the following changes in the Makefile inside the unpacked directory:

```
cd muscle3.8.31/src/
emacs mk
...
# mateo
```

(continues on next page)

(continued from previous page)

```
g++ $ENV_GCC_OPTS -c -O3 -march=native -mfpmath=sse -D_FILE_OFFSET_
↪BITS=64 -DNDEBUG=1 $CPPName.cpp -o $CPPName.o  >> muscle.make.stdout.
↪txt 2>> muscle.make.stderr.txt
...
```

3. Compilation:

```
module load gcc/4.9.4
make
```

4. Install MUSCLE:

```
sudo mkdir -p /share/apps/muscle/gcc/4.9.4/bin
sudo cp  muscle /share/apps/muscle/gcc/4.9.4/bin
```

Module

```
##%Module1.0#####################################################################
↪###
##
## module muscle/3.8.31_gcc-4.9.4
##
## /share/apps/modules/muscle/3.8.31_gcc-4.9.4     Written by Mateo Gomez-
↪Zuluaga
##

proc ModulesHelp { } {
    puts stderr "\tmuscle/3.8.31_gcc-4.9.4 - sets the Environment for MUSCLE␣
↪3.8.31 in \
    \n\tthe share directory /share/apps/muscle/3.8.31/gcc/4.9.4\n"
}

module-whatis "\n\n\tSets the environment for using MUSCLE 3.8.31 \
            \n\tbuilded with GNU GCC 4.9.4\n"

# for Tcl script use only
set       topdir     /share/apps/muscle/3.8.31/gcc/4.9.4
set       version    3.8.31
set       sys        x86_64-redhat-linux

module load gcc/4.9.4

prepend-path PATH    $topdir/bin
```

Use

Slurm template

```
#!/bin/bash
#SBATCH --partition=longjobs
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=32
```

(continues on next page)

(continued from previous page)

```
#SBATCH --time=1:00:00
#SBATCH --job-name=muscle
#SBATCH -o result_%N_%j.out
#SBATCH -e result_%N_%j.err

export SBATCH_EXPORT=NONE
export OMP_NUM_THREADS=1

module load muscle/3.8.31

muscle xxx
```

Resources

- http://www.drive5.com/muscle/

Author

- Mateo Gómez Zuluaga

3.3.44 NAMD

Nanoscale Molecular Dynamics[1] (formerly Not Another Molecular Dynamics Program) is computer software for molecular dynamics simulation, written using the Charm++ parallel programming model. It is noted for its parallel efficiency and is often used to simulate large systems (millions of atoms). It has been developed by the collaboration of the Theoretical and Computational Biophysics Group (TCB) and the Parallel Programming Laboratory (PPL) at the University of Illinois at Urbana–Champaign.

NAMD has an interface to quantum chemistry packages ORCA and MOPAC, as well as a scripted interface to many other quantum packages. Together with Visual Molecular Dynamics (VMD) and QwikMD, NAMD's interface provides access to hybrid QM/MM simulations in an integrated, comprehensive, customizable, and easy-to-use suite.

NAMD 2.13

Basic Information

- **Deploy date:** November 9th 2018
- **Official Website:** https://www.ks.uiuc.edu/Research/namd/
- **License:** University of Illinois NAMD Mlecular Dynamics software license.
- **Installed on:** *Apolo II*

Installation

This entry covers the entire process performed for the installation and configuration of NAMD 2.13 on a cluster.

[1] Wikipedia contributors. (2019, November 30). NAMD. Retrieved December 11, 2019, from https://en.wikipedia.org/wiki/NAMD

Tested on (Requirements)

- **OS base:** CentOS (x86_64) $\geq$ 6.6
- **Compiler:** GCC $\geq$ 7.4.
- **Requirements:**
  - CHARM++ $\geq$ 6.8.2 with GCC 7.4.
  - CUDA 10.1 (Optional).

Build process

1. Download NAMD from the official website, you should create an account and agree the license term.
2. Decompress the tar file.

```
tar xfz NAMD_2.13_Source.tar.gz
cd NAMD_2.13_Source
```

3. Compile CHARM++.

   Charm++[1] is a parallel object-oriented programming language based on C++ and developed in the Parallel Programming Laboratory at the University of Illinois at Urbana–Champaign. Charm++ is designed with the goal of enhancing programmer productivity by providing a high-level abstraction of a parallel program while at the same time delivering good performance on a wide variety of underlying hardware platforms.

   1. Decompress it (the namd source files contains the needed source files of CHARM).

```
tar xf charm-6.8.2.tar
cd charm-6.8.2
```

   2. Compile it. The command structure is `build <target> <version> <options> [charmc-options ...]`

```
## To compile NAMD with CUDA
./build charm++ multicore-linux-x86_64 smp gcc --with-production

## To compile NAMD with MPI
./build charm++ mpi-linux-x86_64 mpicxx gcc --with-production
```

4. Configure the compilation.

```
## To compile NAMD with CUDA
./config Linux-x86_64-g++ --charm-arch multicore-linux-x86_64 --with-cuda --cuda-
↪prefix /path/to/cuda/10.1/

## To compile NAMD with MPI
./config Linux-x86_64-g++ --charm-arch mpi-linux-x86_64-mpicxx
```

5. Compile NAMD.

```
cd Linux-x86_64-g++
make -j 4
```

[1] Wikipedia contributors. (2019a, September 11). Charm++. Retrieved December 11, 2019, from https://en.wikipedia.org/wiki/Charm%2B%2B

Module files

- **NAMD with CUDA**

Listing 48: `2.13-gcc_CUDA`

```
##%Module1.0#####################################################################
↪#####
##
## module load namd/2.13-gcc_CUDA
##
## /share/apps/modules/namd/2.13-gcc_CUDA
## Written by Manuela Carrasco Pinzon
##

proc ModulesHelp {} {
     global version modroot
     puts stderr "Sets the environment for using NAMD 2.13 with CUDA\
                  \nin the shared directory /share/apps/namd/2.13/2.13-gcc_
↪CUDA\
                       \nbuilded with GCC 7.4 and CUDA 10.1"
}

module-whatis "(Name________) namd"
module-whatis "(Version_____) 2.13-gcc_CUDA"
module-whatis "(Compilers___) gcc 7.4 and cuda 10.1"
module-whatis "(System______) x86_64-redhat-linux"
module-whatis "(Libraries___) "

# for Tcl script use only
set         topdir        /share/apps/namd/2.13/2.13-gcc_CUDA/
set         version       2.13-gcc_CUDA
set         sys           x86_64-redhat-linux

conflict mafft
module load gcc/7.4.0
module load cuda/10.1


prepend-path        PATH                        $topdir/bin
```

- **NAMD with MPI**

Listing 49: `2.13-gcc_MPI`

```
##%Module1.0#####################################################################
↪#####
##
## module load namd/2.13-gcc_MPI
##
## /share/apps/modules/namd/2.13-gcc_MPI
## Written by Manuela Carrasco Pinzon
##

proc ModulesHelp {} {
     global version modroot
     puts stderr "Sets the environment for using NAMD 2.13 with MPI\
```

(continues on next page)

(continued from previous page)

```
                \nin the shared directory /share/apps/namd/2.13/2.13-gcc_
↪MPI\
                        \nbuilded with GCC 7.4 and OpenMPI 1.10.7"
}

module-whatis "(Name_________) namd"
module-whatis "(Version______) 2.13-gcc_MPI"
module-whatis "(Compilers____) gcc 7.4 and OpenMPI 1.10.7"
module-whatis "(System_______) x86_64-redhat-linux"
module-whatis "(Libraries____) "

# for Tcl script use only
set         topdir        /share/apps/namd/2.13/2.13-gcc_MPI/
set         version       2.13-gcc_MPI
set         sys           x86_64-redhat-linux

conflict mafft
module load gcc/5.4.0
module load openmpi/1.10.7_gcc-5.4.0


prepend-path        PATH                        $topdir/bin
```

Usage

This section describes the method to submit jobs with the resource manager SLURM.

1. Run WRF from a SLURM bash script, for this example we will use a test case from NAMD official page namd-tutorial-files

```
sbatch example.sh
```

The following code is an example for running NAMD using SLURM:

Listing 50: `example.sh`

```
#!/bin/bash
#SBATCH --job-name=wps-wrf                          # Job name
#SBATCH --mail-type=ALL                             # Mail notification
#SBATCH --mail-user=<user>@<domain>                 # User Email
#SBATCH --error=%x-%j.err                           # Stderr (%j expands to jobId)
#SBATCH --output=%x-%j.out                          # Stdout (%j expands to jobId)
#SBATCH --ntasks=2                                  # Number of tasks (processes)
#SBATCH --nodes=1                                   # Number of nodes
#SBATCH --time=3:00:00                              # Walltime
#SBATCH --partition=longjobs                        # Partition

##### MODULES #####

module load namd/2.13-gcc_CUDA

namd2 ubq_ws_eq.conf
```

Authors

- Vincent Alejandro Arcila Larrea <vaarcilal@eafit.edu.co>.
- Manuela Carrasco Pinzón <mcarras1@eafit.edu.co>

3.3.45 ncl

Description

The Nexus Class Library (NCL) is a C++ library for interpreting data files created according to the NEXUS file format used in phylogenetic systematics and molecular evolution.

Table of Contents

NCL 2.1.18

- **Installation date:** 06/02/2017
- **URL:** https://sourceforge.net/projects/ncl/
- **Apolo version:** Apolo II
- **License:** Simplified BSD license

Dependencies

- Intel Parallel Studio XE 2017 Update 1 (C y C++)

Installation

1. First download the tar from the main page

```
wget https://downloads.sourceforge.net/project/ncl/NCL/ncl-2.1.18/ncl-2.1.18.tar.gz?
↪r=https%3A%2F%2Fsourceforge.net%2Fprojects%2Fncl%2F&ts=1486420255&use_mirror=ufpr
tar -zxvf ncl-2.1.18.tar.gz
```

1. compilation config

```
cd ncl-2.1.18
module load intel/2017_update-1
env CPPFLAGS=-DNCL_CONST_FUNCS ./configure --prefix=/share/apps/ncl/2.1.18/intel/2017_
↪update-1  --build=x86_64-redhat-linux  2>&1 | tee ncl-conf.log
make 2>&1 | tee ncl-make.log
```

Module

```
#%Module1.0#####################################################################
##
## module ncl/2.1.18_intel-2017_update-1
##
## /share/apps/ncl/2.1.18/intel/2017_update-1     Written by Mateo Gomez-Zuluaga
##

proc ModulesHelp { } {
    puts stderr "\tncl/2.1.18_intel-2017_update-1 - sets the Environment for NCL in \
    \n\tthe share directory /share/apps/ncl/2.1.18/intel/2017_update-1\n"
}

module-whatis "\n\n\tSets the environment for using NCL 2.1.18 \
               \n\tbuilded with Intel Parallel Studio XE 2017\n"

# for Tcl script use only
set       topdir     /share/apps/ncl/2.1.18/intel/2017_update-1
set       version    2.1.18
set       sys        x86_64-redhat-linux

module load intel/2017_update-1

prepend-path PATH           $topdir/bin

prepend-path LD_LIBRARY_PATH    $topdir/lib/ncl
prepend-path LIBRARY_PATH       $topdir/lib/ncl
prepend-path LD_RUN_PATH        $topdir/lib/ncl

prepend-path C_INCLUDE_PATH     $topdir/include/ncl
prepend-path CXX_INCLUDE_PATH   $topdir/include/ncl
prepend-path CPLUS_INCLUDE_PATH $topdir/include/ncl

prepend-path PKG_CONFIG_PATH    $topdir/lib/pkgconfig
```

usage mode

```
module load ncl/2.1.18_intel-2017_update-1
```

References

- manual

Author

- Mateo Gómez Zuluaga

3.3.46 OpenFOAM

OpenFOAM is the free, open source CFD software released and developed primarily by OpenCFD Ltd since 2004. It has a large user base across most areas of engineering and science, from both commercial and academic organisations. OpenFOAM has an extensive range of features to solve anything from complex fluid flows involving chemical reactions, turbulence and heat transfer, to acoustics, solid mechanics and electromagnetics. More...

OpenFOAM is professionally released every six months to include customer sponsored developments and contributions from the community, including the OpenFOAM Foundation. Releases designated OpenFOAM+ contain several man years of client-sponsored developments of which much has been transferred to, but not released in the OpenFOAM Foundation branch.

It is independently tested by ESI-OpenCFD's Application Specialists, Development Partners and selected customers, and supported by ESI's worldwide infrastructure, values and commitment.

Quality assurance is based on rigorous testing. The process of code evaluation, verification and validation includes several hundred daily unit tests, a medium-sized test battery run on a weekly basis, and large industry-based test battery run prior to new version releases. Tests are designed to assess regression behaviour, memory usage, code performance and scalability.

Table of Contents

OpenFOAM/v1612+

- **Installation date:** 27/02/2017
- **URL:** http://openfoam.com/
- **Apolo version:** Apolo II
- **License:** General Public Licence

Dependencies

- gcc >= 5.4.0
- python >= 2.7.11
- fftw >= 3.3.5
- zlib >= 1.2.11

- qt >= 4.8.7
- boost >= 1.62.0
- cmake >= 3.7.1
- Intel Parallel Studio XE Cluster Edition 2017 Update 1 (Intel MPI)

Installation

1. First download the tar from the main page

```
wget https://sourceforge.net/projects/openfoamplus/files/v1612+/OpenFOAM-v1612+.tgz
wget https://sourceforge.net/projects/openfoamplus/files/v1612+/ThirdParty-v1612+.tgz
```

#. For the configuration and installation of OpenFOAM it is necessary to build the following dependencies of the software following these steps:

CGAL

```
source /share/apps/openfoam/v1612+/intel_impi/2017_update-1/OpenFOAM-v1612+/etc/bashrc
cd ThirdParty-v1612+
module load fftw/3.3.5_intel_impi-2017_update-1
module load zlib/1.2.11_intel-2017_update-1
module load qt/4.8.7_intel-2017_update-1
module load boost/1.62.0_intel_mkl_impi_2017_update-1
module load python/2.7.11_intel-2017_update-1
module load cmake/3.71
emacs ../OpenFOAM-v1612+/etc/config.sh/CGAL
...
# Modification
boost_version=boost-system
cgal_version=CGAL-4.9

# Modification
export BOOST_ARCH_PATH=/share/apps/boost/1.62.0/intel_impi/2017_update-1
export CGAL_ARCH_PATH=$WM_THIRD_PARTY_DIR/platforms/$WM_ARCH$WM_COMPILER/$cgal_version
...
emacs makeCGAL
...
# Enable/disable gmp/mpfr together
if _foamIsNone $gmpPACKAGE || _foamIsNone $mpfrPACKAGE
then
    GMP_ARCH_PATH=none
    MPFR_ARCH_PATH=none
elif _foamIsSystem $GMP_ARCH_PATH || _foamIsSystem $MPFR_ARCH_PATH
then
    # Modification
    GMP_ARCH_PATH=/share/apps/gmp/6.1.1    # for an accurate record
    MPFR_ARCH_PATH=/share/apps/mpfr/3.1.5
fi
...

sudo ln -s /share/apps/gmp/6.11/lib /share/apps/gmp/6.11/lib64
sudo ln -s /share/apps/mpfr/3.1.5/lib /share/apps/mpfr/3.1.5/lib64
./makeCGAL
```

MESA

```
wget ftp://ftp.freedesktop.org/pub/mesa/13.0.4/mesa-13.0.4.tar.gz
./makeMESA mesa-13.0.4
```

ParaView

```
# Modificar la versión de MESA para que corresponda
emacs makeParaview.example
...
# Modification
mesa=mesa-13.0.4
...
# Modificar el siguiente archivo
emacs etc/tools/ParaViewFunctions
...
# Modification
pythonInclude=/share/apps/python/2.7.12/intel/2017_update-1/intelpython27/include/
↪python$pythonMajor
...
# Comentar las siguientes líneas en el archivo indicado:
emacs ParaView-5.0.1/Qt/Components/pqPresetToPixmap.cxx
...
    // Now determine best value for Nh in [Nh/2,Nh-1]
    double bestQ = vtkMath::Inf();
    int best = -1;
    // mateo
    //for (int i = Nh / 2; i < Nh; ++i)
    //  {
    double ar = Nv * wmp / static_cast<double>(hmp * Nh);
    double q = ( ar >= 1.0 ) ? ar : 1. / ar;
    if ( q < bestQ )
      {
        bestQ = q;
        best = Nh-1;//i;
      }
    //}
    Nh = best;
    }
...
./makeParaView.example -python
```

VTK

```
ln -s ParaView-5.0.1/VTK  VTK-7.1.0
# Agregar versión de MESA
emacs makeVTK.example
...
mesa=mesa-13.0.4
...
./makeVTK.example -mpi=0
```

FFTW

```
emacs ../OpenFOAM-v1612+/etc/config.sh/FFTW
...
# Modification
fftw_version=fftw-system
../OpenFOAM-v1612+/etc/config.sh/FFTW
```

(continues on next page)

(continued from previous page)

```
# Modifictation
export FFTW_ARCH_PATH=/share/apps/fftw/3.3.5/intel_impi/2017_update-1
...
```

1. Edit the makefile

```
emacs etc/wmakeFiles/scotch/Makefile.inc.i686_pc_linux2.shlib-OpenFOAM
...
MAKE         = make
AR           = icc
ARFLAGS      = $(WM_CFLAGS) -shared -o
CAT          = cat
CCS          = icc
CCP          = mpiicc
CCD          = mpiicc
...
module unload python
./Allwmake
```

Slurm template

```
#!/bin/bash
#SBATCH --partition=longjobs
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=32
#SBATCH --time=1:00:00
#SBATCH --job-name=vsearch
#SBATCH -o result_%N_%j.out
#SBATCH -e result_%N_%j.err

export SBATCH_EXPORT=NONE
export OMP_NUM_THREADS=???

xxx
```

References

- https://openfoamwiki.net/index.php/Installation/Linux/OpenFOAM-4.0/CentOS_SL_RHEL
- http://openfoam.com/documentation/system-requirements.php
- http://openfoam.com/download/install-source.php
- http://openfoam.com/code/build-guide.php
- https://software.intel.com/en-us/forums/intel-c-compiler/topic/702934

Author

- Mateo Gómez Zuluaga

Table of Contents

OpenFOAM/v1712+

- **Installation date:** 16/02/2018
- **URL:** http://openfoam.com/
- **Apolo version:** Cronos
- **License:** General Public Licence

Dependencies

-GNU GCC >= 5.4.0 - Python2 (Intel version) >= 2.7.14 - Boost >= 1.67.0 - cmake >= 3.7.1 - OpenMPI >= 1.10.7

Installation

1. First download the tar from the main page

```
cd /share/apps/openfoam/v1712/gcc-5.4.0/OpenFOAM-v1712
wget https://sourceforge.net/projects/openfoamplus/files/v1712/OpenFOAM-v1712.tgz
wget https://sourceforge.net/projects/openfoamplus/files/v1712/ThirdParty-v1712.tgz
tar xf OpenFOAM-v1712.tgz
tar xf ThirdParty-v1712.tgz
```

1. Edit the following archives:

/share/apps/openfoam/v1712+/gcc-5.5.0/OpenFOAM-v1712/etc/bashrc:

```
...
FOAM_INST_DIR=/share/apps/openfoam/v1712/gcc-5.4.0/OpenFOAM-v1712
# FOAM_INST_DIR=/opt/$WM_PROJECT
# FOAM_INST_DIR=/usr/local/$WM_PROJECT
...
export WM_COMPILER_TYPE=system
...
export WM_COMPILER=Gcc54
...
export WM_MPLIB=SYSTEMOPENMPI
...
```

** /share/apps/openfoam/v1712+/gcc-5.5.0/OpenFOAM-v1712/etc/config.sh/CGAL:**

```
...
boost_version=boost-system
...
export BOOST_ARCH_PATH=/share/apps/boost/1.67.0/gcc-5.4.0
...
```

1. Load compilation environment:

```
source /share/apps/openfoam/v1712+/gcc-5.5.0/OpenFOAM-v1712/etc/bashrc
cd ThirdParty-v1712
module load python/2.7.12_intel-2017_update-1
module load boost/1.67.0_gcc-5.4.0_openmpi-1.10.7
module load cmake/3.7.1
```

1. Edit archives:

makeCGAL

```
 ...
# Enable/disable gmp/mpfr together
if _foamIsNone $gmpPACKAGE || _foamIsNone $mpfrPACKAGE
then
    GMP_ARCH_PATH=none
    MPFR_ARCH_PATH=none
elif _foamIsSystem $GMP_ARCH_PATH || _foamIsSystem $MPFR_ARCH_PATH
then
    # Modification
   GMP_ARCH_PATH=/share/apps/gmp/6.1.1     # for an accurate record
   MPFR_ARCH_PATH=/share/apps/mpfr/3.1.5
fi
...
```

```
sudo ln -s /share/apps/gmp/6.1.1/lib /share/apps/gmp/6.1.1/lib64
sudo ln -s /share/apps/mpfr/3.1.5/lib /share/apps/mpfr/3.1.5/lib64
```

1. compile cgal

```
./makeCGAL
```

1. OpenFoam installation

```
cd ../OpenFOAM-v1712
./Allwmake
```

Slurm template

```
#!/bin/sh

#SBATCH --partition=bigmem
#SBATCH --nodes=2
#SBATCH --ntasks-per-node=24
#SBATCH --time=14-00
#SBATCH --job-name=OpenFOAM_1
#SBATCH -o result_%N_%j.out
#SBATCH -e result_%N_%j.err
```

(continues on next page)

(continued from previous page)

```
#SBATCH --mail-type=ALL
#SBATCH --mail-user=dtobone@eafit.edu.co

# Don't share environment variables
export SBATCH_EXPORT=NONE
export OMP_NUM_THREADS=1
# Debug OpenFOAM
#export FOAM_ABORT=1

# R U in cronos or apolo2?
if [[ "${SLURM_SUBMIT_HOST}" != "apolo.eafit.edu.co" ]]; then
## OpenFOAM-v1712 - Cronos Configuration
    echo "No estoy en apolo"
    module load python/2.7.14_intel-18_u1
    module load openmpi/1.10.7_gcc-5.5.0
    module load fftw/3.3.7_gcc-5.5.0
    module load boost/1.66.0_gcc-5.5.0
    source /share/apps/openfoam/v1712+/gcc-5.5.0/OpenFOAM-v1712/etc/bashrc
else
## OpenFOAM-v1612 - Apolo Configuration
    echo "Estoy en Apolo"
    module load boost/1.67.0_gcc-5.4.0_openmpi-1.10.7
    source /share/apps/openfoam/v1712/gcc-5.4.0/OpenFOAM-v1712/etc/bashrc > /dev/null␣
↪2>&1
fi


# Source tutorial run functions
. $WM_PROJECT_DIR/bin/tools/RunFunctions

#------------------------------------------------------------------------------
#     BORRAR Y ORGANIZAR LOS ARCHIVOS INICILES PARA LA SIMULACIÓN
#------------------------------------------------------------------------------

rm -rf processor*                                #Borra carpetas procesadores
rm -rf file log.pimpleFoam                       #Borra archivos de simulaciones␣
↪pasadas
rm -rf file log.snappyHexMesh                    #Borra archivos de simulaciones␣
↪pasadas
rm -rf file log.renumberMesh                     #Borra archivos de simulaciones␣
↪pasadas

#------------------------------------------------------------------------------
#     PASOS PREVIOS - MALLADO - VERIFICACIÓN
#------------------------------------------------------------------------------

mv 0 0.org                                  #Mueve los datos para que no se dañe al␣
↪hacer la malla
mkdir 0                                     #Crea una carpeta de 0 falsa
cd constant/triSurface                      #Entra donde estan los archivos stl para␣
↪la malla
surfaceTransformPoints -scale '(0.001 0.001 0.001)' vane_mm.stl vane_m.stl #Escala la␣
↪malla
cd ..                                       #Se devuelve a la carpeta constant
cd ..                                       #Se devuelve a la carpeta de la simulación
blockMesh                                   #Crea la malla base, o geometria de␣
↪referencia
```

(continues on next page)

(continued from previous page)

```
surfaceFeatureExtract                          #Extrae las superficies de los stl para␣
↪la malla
decomposePar                                   #Parte las instancias para los procesos␣
↪en paralelo
srun snappyHexMesh -parallel -overwrite        #Crea la malla en paralelo
reconstructParMesh -constant                   #Unifica la malla en la carpeta constant
rm -rf processor*                              #Borra carpetas procesadores
rm -rf 0                                       #Borra la carpeta 0 falsa
mv 0.org 0                                     #Trae de vuela la carpeta original 0
checkMesh -allTopology -allGeometry            #Verifica la calidad de la malla

#------------------------------------------------------------------------------
#     PROCESO DE SIMULACIÓN Y UNIFICACIÓN DE RESULTADOS
#------------------------------------------------------------------------------

decomposePar                                   #Parte las instancias para los procesos␣
↪en paralelo
srun renumberMesh -overwrite                   #Reescribe la malla de forma que sea mas␣
↪estable a la hora de la simulación
srun `getApplication` -parallel                #Inicia el proceso de cálculo
reconstructPar                                 #Reconstruye los resultados en las␣
↪carpetas del tiempo

#------------------------------------------------------------------------------
#     BORRADO Y LIMPIEZA DE ARCHIVOS QUE NO SON NECESARIOS
#------------------------------------------------------------------------------

rm -rf processor*                              #Borra carpetas procesadores
. $WM_PROJECT_DIR/bin/tools/CleanFunctions  # Source tutorial clean functions

#------------------------------------------------------------------------------
#     FIN DE LA SIMULACIÓN
#------------------------------------------------------------------------------
```

References

- https://openfoamwiki.net/index.php/Installation/Linux/OpenFOAM-4.0/CentOS_SL_RHEL
- http://openfoam.com/documentation/system-requirements.php
- http://openfoam.com/download/install-source.php
- http://openfoam.com/code/build-guide.php
- https://software.intel.com/en-us/forums/intel-c-compiler/topic/702934

Author

- Mateo Gómez Zuluaga

OpenFOAM/v2006+

Table of Contents

- **Installation date:** 16/02/2021
- **URL:** http://openfoam.com/
- **Apolo version:** Apolo
- **License:** General Public Licence

Dependencies

- GNU GCC >= 7.4.0
- Boost >= 1.67.0
- cmake >= 3.7.1
- OpenMPI >= 3.1.5

Installation

1. First download the tar from the main page

```
cd /share/apps/openfoam/v2006/gcc-7.4.0/OpenFOAM-v2006
wget https://sourceforge.net/projects/openfoamplus/files/v2006/OpenFOAM-
↪v2006.tgz
wget https://sourceforge.net/projects/openfoamplus/files/v2006/ThirdParty-
↪v2006.tgz
tar xf OpenFOAM-v2006.tgz
tar xf ThirdParty-v2006.tgz
```

2. Load the dependencies' modules:

```
module load gcc/7.4.0 openmpi/3.1.5_intel-19.0.4 boost/1.67.0_gcc-5.4.0␣
↪cmake/3.7.1
```

3. Edit the following archives:

   `/share/apps/openfoam/v2006/gcc-7.4.0/OpenFOAM-v2006/etc/bashrc`

```
...
export WM_COMPILER_TYPE=system
...
export WM_COMPILER=Gcc74
...
export WM_MPLIB=SYSTEMOPENMPI
...
```

(continues on next page)

(continued from previous page)

```
projectDir=/share/apps/openfoam/v2006/gcc-7.4.0/OpenFOAM-v2006
...
```

/share/apps/openfoam/v2006/gcc-7.4.0/OpenFOAM-v2006/etc/config.sh/CGAL

```
...
boost_version=boost-system
...
export BOOST_ARCH_PATH=/share/apps/boost/1.67.0/gcc-5.4.0
...
```

4. Load compilation environment:

```
source /share/apps/openfoam/v2006/gcc-7.4.0/OpenFOAM-v2006/etc/bashrc
cd ThirdParty-v2006
module load boost/1.67.0_gcc-5.4.0_openmpi-1.10.7
module load cmake/3.7.1
```

5. Edit archives:

/share/apps/openfoam/v2006/gcc-7.4.0/ThirdParty-v2006/makeCGAL

```
...
        if _foamIsNone "$gmpPACKAGE" || _foamIsNone "$mpfrPACKAGE"
    then
        GMP_ARCH_PATH=none
        MPFR_ARCH_PATH=none
    elif _foamIsSystem "$gmpPACKAGE" || _foamIsSystem "$mpfrPACKAGE"
    then
        # May really be system, but could also by a central installation
        # Ensure everything is accurately recorded. Resolve paths etc.

        if [ -d "$GMP_ARCH_PATH" ]
        then
            if GMP_ARCH_PATH=$(cd "$GMP_ARCH_PATH" 2>/dev/null && pwd -P)
            then
                gmpPACKAGE="${GMP_ARCH_PATH##*/}"
            else
                echo "ERROR: bad path for GMP_ARCH_PATH"
                echo "stopping build"
                exit 1
            fi
        else
            GMP_ARCH_PATH=/share/apps/gmp/6.1.1/
        fi

        if [ -d "$MPFR_ARCH_PATH" ]
        then
            if MPFR_ARCH_PATH=$(cd "$MPFR_ARCH_PATH" 2>/dev/null && pwd -
↪P)
            then
                mpfrPACKAGE="${MPFR_ARCH_PATH##*/}"
            else
                echo "ERROR: bad path for MPFR_ARCH_PATH"
                echo "stopping build"
                exit 1
```

(continues on next page)

(continued from previous page)

```
            fi
        else
            MPFR_ARCH_PATH=/share/apps/mpfr/3.1.5/
        fi
    else
        GMP_ARCH_PATH=/share/apps/gmp/6.1.1/
        MPFR_ARCH_PATH=/share/apps/mpfr/3.1.5/
    fi

...
```

```
sudo ln -s /share/apps/gmp/6.1.1/lib /share/apps/gmp/6.1.1/lib64
sudo ln -s /share/apps/mpfr/3.1.5/lib /share/apps/mpfr/3.1.5/lib64
```

6. compile cgal

```
./makeCGAL
```

7. OpenFoam installation

```
cd ../OpenFOAM-v2006
./Allwmake
```

Slurm template

```
#!/bin/sh

#SBATCH --partition=bigmem
#SBATCH --nodes=2
#SBATCH --ntasks-per-node=24
#SBATCH --time=14-00
#SBATCH --job-name=OpenFOAM_1
#SBATCH -o result_%N_%j.out
#SBATCH -e result_%N_%j.err
#SBATCH --mail-type=ALL
#SBATCH --mail-user=dtobone@eafit.edu.co

# Don't share environment variables
export SBATCH_EXPORT=NONE
export OMP_NUM_THREADS=1
# Debug OpenFOAM
#export FOAM_ABORT=1

# R U in cronos or apolo2?
if [[ "${SLURM_SUBMIT_HOST}" != "apolo.eafit.edu.co" ]]; then
## OpenFOAM-v1712 - Cronos Configuration
    echo "No estoy en apolo"
    module load openmpi/3.1.5_intel-19.0.4
    module load fftw/3.3.7_gcc-5.5.0
    module load boost/1.66.0_gcc-5.5.0
    source /share/apps/openfoam/v2006/gcc-7.4.0/OpenFOAM-v2006/etc/bashrc
else
## OpenFOAM-v1612 - Apolo Configuration
    echo "Estoy en Apolo"
```

(continues on next page)

(continued from previous page)

```
    module load boost/1.67.0_gcc-5.4.0_openmpi-1.10.7
    source /share/apps/openfoam/v1712/gcc-5.4.0/OpenFOAM-v1712/etc/bashrc > /dev/null␣
↪2>&1
fi


# Source tutorial run functions
. $WM_PROJECT_DIR/bin/tools/RunFunctions

#------------------------------------------------------------------------------
#     BORRAR Y ORGANIZAR LOS ARCHIVOS INICILES PARA LA SIMULACIÓN
#------------------------------------------------------------------------------

rm -rf processor*                                #Borra carpetas procesadores
rm -rf file log.pimpleFoam                       #Borra archivos de simulaciones␣
↪pasadas
rm -rf file log.snappyHexMesh                    #Borra archivos de simulaciones␣
↪pasadas
rm -rf file log.renumberMesh                     #Borra archivos de simulaciones␣
↪pasadas

#------------------------------------------------------------------------------
#     PASOS PREVIOS - MALLADO - VERIFICACIÓN
#------------------------------------------------------------------------------

mv 0 0.org                                  #Mueve los datos para que no se dañe al␣
↪hacer la malla
mkdir 0                                     #Crea una carpeta de 0 falsa
cd constant/triSurface                      #Entra donde estan los archivos stl para␣
↪la malla
surfaceTransformPoints -scale '(0.001 0.001 0.001)' vane_mm.stl vane_m.stl #Escala la␣
↪malla
cd ..                                       #Se devuelve a la carpeta constant
cd ..                                       #Se devuelve a la carpeta de la simulación
blockMesh                                   #Crea la malla base, o geometria de␣
↪referencia
surfaceFeatureExtract                       #Extrae las superficies de los stl para␣
↪la malla
decomposePar                                #Parte las instancias para los procesos␣
↪en paralelo
srun snappyHexMesh -parallel -overwrite     #Crea la malla en paralelo
reconstructParMesh -constant                #Unifica la malla en la carpeta constant
rm -rf processor*                           #Borra carpetas procesadores
rm -rf 0                                    #Borra la carpeta 0 falsa
mv 0.org 0                                  #Trae de vuela la carpeta original 0
checkMesh -allTopology -allGeometry         #Verifica la calidad de la malla

#------------------------------------------------------------------------------
#     PROCESO DE SIMULACIÓN Y UNIFICACIÓN DE RESULTADOS
#------------------------------------------------------------------------------

decomposePar                                #Parte las instancias para los procesos␣
↪en paralelo
srun renumberMesh -overwrite                #Reescribe la malla de forma que sea mas␣
↪estable a la hora de la simulación
srun `getApplication` -parallel             #Inicia el proceso de cálculo
reconstructPar                              #Reconstruye los resultados en las␣
↪carpetas del tiempo
```

(continues on next page)

(continued from previous page)

```
#------------------------------------------------------------------------------
#     BORRADO Y LIMPIEZA DE ARCHIVOS QUE NO SON NECESARIOS
#------------------------------------------------------------------------------

rm -rf processor*                          #Borra carpetas procesadores
. $WM_PROJECT_DIR/bin/tools/CleanFunctions  # Source tutorial clean functions

#------------------------------------------------------------------------------
#     FIN DE LA SIMULACIÓN
#------------------------------------------------------------------------------
```

References

- https://openfoamwiki.net/index.php/Installation/Linux/OpenFOAM-4.0/CentOS_SL_RHEL
- http://openfoam.com/documentation/system-requirements.php
- http://openfoam.com/download/install-source.php
- http://openfoam.com/code/build-guide.php
- https://software.intel.com/en-us/forums/intel-c-compiler/topic/702934

Author

- Tomás Navarro <tdnavarrom@eafit.edu.co>

3.3.47 OpenQuake

The OpenQuake Engine is the Global Earthquake Model Foundation's (GEM) state-of-the-art, open-source software collaboratively developed for earthquake hazard and risk modelling. It runs on operating systems such as Linux, macOS and Windows; and can be deployed on laptops, desktops, standalone servers and multi-node clusters. The functionality to analyze hazard and risks at specific site, city, country or regional level makes the OpenQuake Engine a powerful and dynamic tool for assessing the potential impacts of earthquakes at any location in the world.[1]

OpenQuake 3.9

Table of Contents

[1] https://www.globalquakemodel.org/oq-get-started

Basic information

- **Official Website:** https://www.globalquakemodel.org/oq-get-started
- **Download Website:** https://github.com/gem/oq-engine
- **License:** GNU Affero General Public License v3.0
- **Installed on:** Apolo II
- **Installation date:** 12/06/2020

Tested on (Requirements)

- **OS base:** CentOS (x86_64) $\geq$ 6.6 (Rocks 6.2)
- **Dependencies to run OpenQuake:**
  - Python 3.X (tested on Python 3.6.5)

Installation

The following procedure is the easiest way to install OpenQuake 3.9 in a cluster.

> **Note:** OpenQuake at the moment only runs via `.py` files, therefore is not posible to install it properly in a cluster. Only the user that clones the repo can run OpenQuake.

Now, for being able to install and run OpenQuake please follow these instructions:

1. Load the dependencies so OpenQuake will be able to run.

```
$ module load python/3.6.5_miniconda-4.5.1
```

2. We create a conda environment, so we can run OpenQuake in a secure environment (it is not necessary but we recommend it).

```
$ conda create --name openquake3.9
$ conda activate openquake3.9
```

3. Clone the repository

```
$ mkdir src && cd src
$ git clone https://github.com/gem/oq-engine.git --depth=1
```

4. Install the dependencies

```
$ pip install -r oq-engine/requirements-py36-linux64.txt -r oq-engine/
↪requirements-extra-py36-linux64.txt
$ pip install -e oq-engine/[dev]
```

5. To verify that the installation was done correctly, run the following command:

```
$ oq engine
```

The output should be similar to this:

```
usage: oq engine [-h] [--log-file LOG_FILE] [--no-distribute] [-y]
         [-c CONFIG_FILE] [--make-html-report YYYY-MM-DD|today] [-u]
         [-d] [-w] [--run JOB_INI [JOB_INI ...]]
         [--list-hazard-calculations] [--list-risk-calculations]
         [--delete-calculation CALCULATION_ID]
         [--delete-uncompleted-calculations]
         [--hazard-calculation-id HAZARD_CALCULATION_ID]
         [--list-outputs CALCULATION_ID] [--show-log CALCULATION_ID]
         [--export-output OUTPUT_ID TARGET_DIR]
         [--export-outputs CALCULATION_ID TARGET_DIR] [-e]
         [-l {debug, info, warn, error, critical}] [-r]
         [--param PARAM]
```

Warning: In case of multiple installations:

If any other installation of the Engine exists on the same machine, like a system-wide installation made with packages, you must change the DbServer port from the default one (1908) to any other unused port. Change it in the file: `oq-engine/openquake/engine/openquake.cfg`

Usage

An OpenQuake-engine seismic source input model contains a list of sources belonging to a finite set of possible typologies. Each source type is defined by a set of parameters - called source data - which are used to specify the source geometry and the properties of seismicity occurrence.

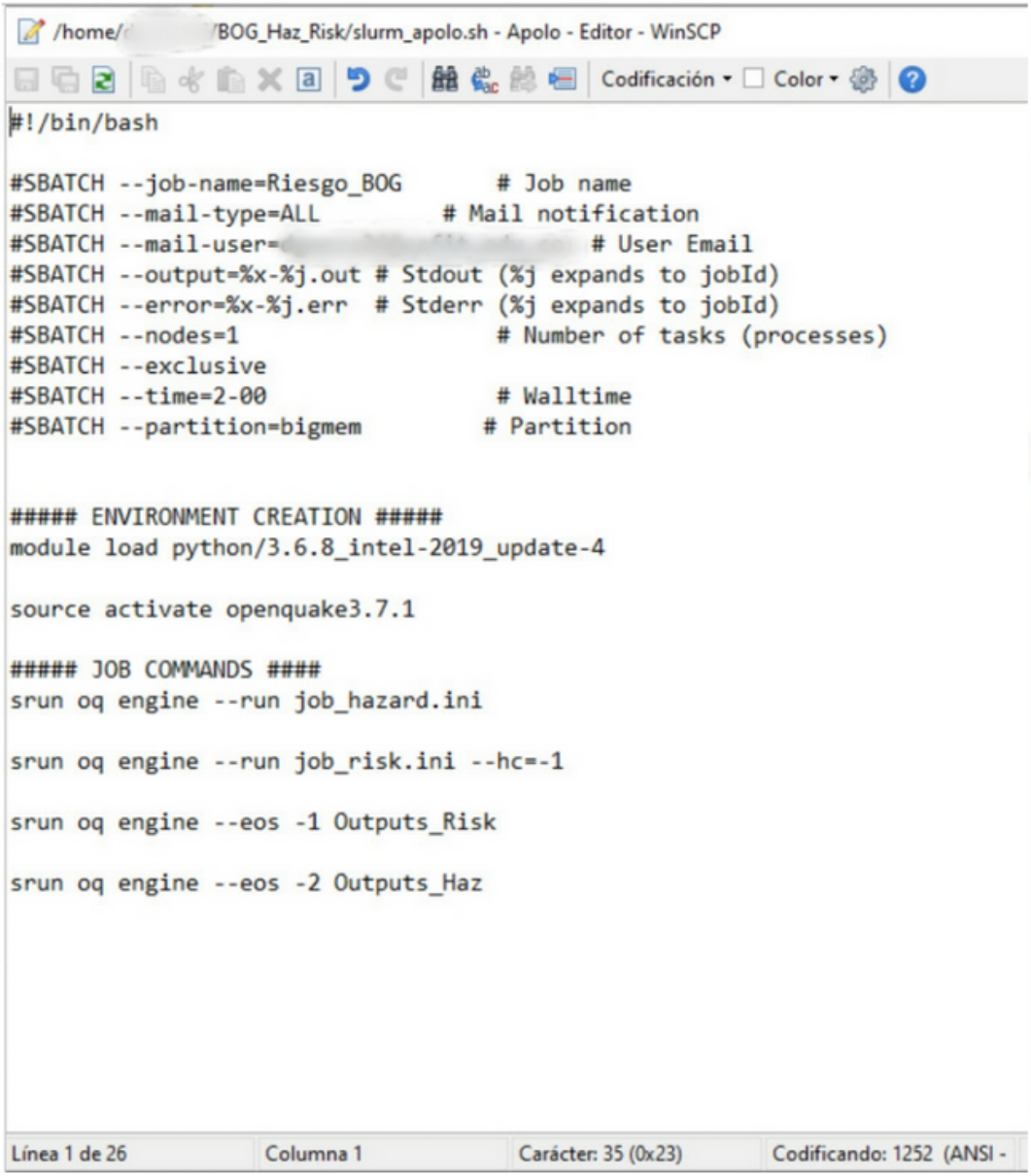
To measure the impacts of an earthquake, OpenQuake relies on two calculations, hazard and risk calculations, which must be performed through a configuration file called job.ini See an example of a job.ini for a hazard calculation:

job.ini

```
[general]
description = Scenario Hazard Config File
calculation_mode = scenario

[sites]
sites_csv = sites.csv

[rupture]
rupture_model_file = rupture_model.xml
rupture_mesh_spacing = 2.0

[site_params]
site_model_file = site_model.xml

[correlation]
ground_motion_correlation_model = JB2009
ground_motion_correlation_params = {"vs30_clustering": True}

[hazard_calculation]
intensity_measure_types = PGA, SA(0.3), SA(1.0)
random_seed = 42
truncation_level = 3.0
maximum_distance = 200.0
gsim = BooreAtkinson2008
number_of_ground_motion_fields = 1000
```

Running Example

1. This is an example for Apolo in SLURM.



2. Use this command to run the job:

```
$ sbatch slurm_Apolo.sh
```

Basic Commands

1. To see the results of running the calculations, use the following command (the first one in case of risk calculations, the second one in case of hazard calculations).

```
$ oq engine --lrc

$ oq engine --lhc
```

2. To see the specific outputs of a calculation, identify the calculation id and use the following command. You can see the calculation after running the previous command and then identify the number at the beginning of the

required calculation line.

```
$ oq engine --lo <calculation_id>
```

3. To export a specific output of a calculation, identify the output id (you can see it after running the previous command) and use this command to get the output and save it in the directory of your preference, if it doesn't exist, it will be created.

```
$ oq engine --eo <output_id> <directory_path>
```

4. Always remember to end OpenQuake after finishing the job using this command.

```
$ oq dbserver stop
```

See also:

1. For a detailed manual of the use of the application go to https://docs.openquake.org/manuals/OpenQuake%20Manual%20%28latest%29.pdf
2. To directly generate the job.ini file, OpenQuake has a tool called ipt so the user only has to set the necessary parameters. More information at: https://github.com/gem/oq-engine/blob/engine-3.9/doc/running/server.md

---

Note:

If it fails when you run the job, use this command to see error details.

```
$ less <job_name>-<job_id>.err
```

The job id is the one specified in the slurm file as job-name, and the id is the one sent by email when the job started running.

For more information of how to use OpenQuake, please visit the official website.

Change it in the file: oq-engine/openquake/engine/openquake.cfg

For more information on how to use OpenQuake, please visit the official website.

References

OpenQuake - OpenQuake Official website. https://www.globalquakemodel.org/oq-get-started

Installation - OpenQuake Official Website. https://github.com/gem/oq-engine

Usage - OpenQuake User Manual https://docs.openquake.org/manuals/OpenQuake%20Manual%20%28latest%29.pdf

Author

- Laura Sánchez Córdoba <lsanchezc@eafit.edu.co>

3.3.48 OpenSees

OpenSees (the Open System for Earthquake Engineering Simulation) is a proprietary object-oriented, software framework created at the National Science Foundation-sponsored Pacific Earthquake Engineering (PEER) Center. It allows users to create finite element applications for simulating the response of structural and geotechnical systems subjected

to earthquakes. This framework was developed by Frank McKenna and Gregory L. Fenves with significant contributions from Michael H. Scott, Terje Haukaas, Armen Der Kiureghian, Remo M. de Souza, Filip C. Filippou, Silvia Mazzoni, and Boris Jeremic. OpenSees is primarily written in C++ and uses several Fortran numerical libraries for linear equation solving.[1]

OpenSees 3.0.0

Table of Contents

Basic information

- **Official Website:** http://opensees.berkeley.edu/
- **License:** http://opensees.berkeley.edu/OpenSees/copyright.php
- **Installed on:** *Apolo II* , *Cronos*

Tested on (Requirements)

- **OS base:** CentOS (x86_64) $\geq$ 6.6 (Rocks 6.2)
- **Compiler:** Intel MPI Library $\geq$ 17.0.1 (Apolo)
- **Math Library:** ScaLAPACK $\geq$ 2.0.2 and MUMPS $\geq$ 5.0.2

Installation

The following procedure present the way to compile **OpenSeesMP** and its dependencies for distributed computing using Intel MPI.[1]

[1] Wikipedia contributors. (2019, March 9). OpenSees. In Wikipedia, The Free Encyclopedia. Retrieved April 10, 2019, from https://en.wikipedia.org/w/index.php?title=OpenSees&oldid=886922043

[1] OpenSees Parallel - OpenSees official site. Retrieved April 12, 2019, from http://opensees.berkeley.edu/OpenSees/parallel/parallel.php

OpenSees

1. Download the latest version of OpenSees from Github

```
$ wget https://github.com/OpenSees/OpenSees/archive/v3.0.0.tar.gz
$ tar xf v3.0.0.tar.gz
```

2. Inside the folder, on the top create a `Makefile.def` file. You can copy one of the examples from the MAKES folder depending on your architecture and adapt it.

```
$ cp MAKES/Makefile.def.INTEL2 ./Makefile.def
```

Note: We recommend to use our `Makefile.def` which has been simplified for OpenSeesMP and Intel MPI. `Makefile.def`

---

3. Modify the following lines in the `Makefile.def`

```
# The location where the OpenSees folder is placed. It is expected for the sake␣
↪of simplicity that the OpenSees'
# folder is named just **OpenSees** because of the Makefile.def will look for␣
↪this name in the HOME folder in all
# defined paths.
HOME   = /home/jyepesr1/source

# The location of the final binary. NOTE: create the bin folder previously.
OpenSees_PROGRAM = $(HOME)/OpenSees/bin/OpenSeesMP

# Ensure you have the Tcl library installed and check its version, in that case,␣
↪8.5. libtcl8.5 is located in
# an standard location /usr/lib64/libtcl8.5.so, after the package installation.
TCL_LIBRARY = -ltcl8.5

# MUMPS dir where it was compiled.
MUMPS_DIR = /home/jyepesr1/source/mumps

# MUMPS has some dependencies scotch, pt-scotch, METIS and ParMETIS which are the␣
↪serial and parallel versions
# scotch and pt-scotch are in the same folder because they are compiled together.
SCOTCHDIR  = /home/jyepesr1/apps/scotch/6.0.6
METISDIR = /home/jyepesr1/apps/metis/5.1.0
PARMETISDIR = /home/jyepesr1/apps/parmetis/4.0.3

# Parallel lib, we can use ScaLAPACK or MKL, in that case, we will the first one␣
↪because there are some routines
# in OpenSees NOT well supported with MKL and your code could fail.
PARALLEL_LIB = -L/home/jyepesr1/apps/scalapack/2.0.2-impi_18.0.2/lib -lscalapack -
↪lreflapack -lrefblas -ltmg
```

4. Create the `lib/` and `bin/` directories in the OpenSees top folder. The compilation will place the libraries and the final binary in that routes if you did not change the paths in the Makefile.

```
$ mkdir OpenSees/{bin,lib}
```

5. Run the make command and wait for the compilation.

```
$ make -j10
```

Note: Remember to load the Intel MPI module for all compilations. `module load impi/2017_update-1`

ScaLAPACK

ScaLAPACK is a library of high-performance linear algebra routines for parallel distributed memory machines. This solves dense and banded linear systems, least squares problems, eigenvalue problems, and singular value problems.[2]

ScaLAPACK integrates a python script which can configure and install in a quick way all the requirements and the library itself, so we strongly recommend using this method.

1. Download the installer

```
$ wget http://www.netlib.org/scalapack/scalapack_installer.tgz
$ tar xf scalapack_installer.tgz
$ cd scalapack_installer/
```

2. Edit the `netlib.py` file changing the `cc` and `fc` variables to use the Intel compiler.

```
cc  = "icc"          # the C compiler for plasma
fc  = "ifort"        # the Fortran compiler for core_lapack
```

3. Create the folder where the build will be placed and execute the `setup.py` command. Check what options are the best choice for your architecture.

```
$  mkdir -p /home/jyepesr1/source/apolo/scalapack/2.0.2-impi_17.0.1
$ ./setup.py --prefix=/home/jyepesr1/source/apolo/scalapack/2.0.2-impi_17.0.1 \
  --mpibindir=/share/apps/intel/ps_xe/2017_update-1/compilers_and_libraries/linux/
↪mpi/bin64 \
  --mpicc=mpiicc --mpif90=mpiifort \
  --mpiincdir=/share/apps/intel/ps_xe/2017_update-1/compilers_and_libraries/linux/
↪mpi/include64 \
  --ccflags="-xHost -O3" --fcflags="-xHost -O3" --downall --ldflags_fc="-nofor_
↪main"
```

Note: When compiling with Intel the configuration will require the `-nofor_main` flag in the fortran linker because the compiler will try to look for the main function in the Fortran files by default.

Note: The program will try to execute some examples to test MPI in C and Fortran. In our case these examples will fail because in our architecture MPI cannot run without `srun --mpi=pmi2` command

Warning: The following steps are optional and will be executed due to the restriction of our current architecture

[2] ScaLAPACK—Scalable Linear Algebra PACKage - ScaLAPACK official site. Retrieved April 12, 2019, from http://www.netlib.org/scalapack/

4. Edit the `scripts/framework.py` file to avoid execution halt due to mpirun restrictions. Go to the functions `def check_mpicc()` and `def check_mpif90()`, and comment out the lines that checks the mpirun execution. Finally, run again the `setup.py`

```
def check_mpicc(self):
    .
    .
    .
    # run
    # comm = self.config.mpirun + ' ./tmpc'
    # (output, error, retz) = runShellCommand(comm)
    # if retz:
    #     print '\n\nCOMMON: mpirun not working! aborting...'
    #     print 'error is:\n','*'*40,'\n',error,'\n','*'*40
    #     sys.exit()

    .
    .
    .

def check_mpif90(self):
    .
    .
    .
    # run
    # comm = self.config.mpirun + ' ./tmpf'
    # (output, error, retz) = runShellCommand(comm)
    # if retz:
    #     print '\n\nCOMMON: mpif90 not working! aborting...'
    #     print 'error is:\n','*'*40,'\n',error,'\n','*'*40
    #     sys.exit()
```

Warning: Sometimes depending on your architecture the different tests could fail, in such case, you can ignore them and continue checking that all libraries have been placed in the destination folder.

5. The final step is checking that the libraries are placed in the destination folder.

```
$ tree /home/jyepesr1/source/apolo/scalapack/2.0.2-impi_17.0.1/lib/

/home/jyepesr1/source/apolo/scalapack/2.0.2-impi_17.0.1/lib/
├── librefblas.a
├── libreflapack.a
├── libscalapack.a
└── libtmg.a
```

MUMPS

MUMPS (MUltifrontal Massively Parallel Sparse direct Solver) can solve very large linear systems through in/out-of-core LDLt or LU factorisation.[3]

Before compile MUMPS its dependencies have to be installed.

1. Go to the MUMPS folder and copy an example of a Makefile from the `Make.inc/` folder to edit its content

[3] MUMPS: a parallel sparse direct solver - MUMPS official site. Retrieved April 12, 2019, from http://mumps.enseeiht.fr/

```
$ wget http://mumps.enseeiht.fr/MUMPS_5.0.2.tar.gz
$ tar xf MUMPS_5.0.2.tar.gz
$ cd MUMPS_5.0.2
$ ln -s Make.inc/Makefile.INTEL.PAR Makefile.inc
```

2. Edit the following lines in the `Makefile.inc`.

Listing 51: `Makefile.inc`

```
  # Change and uncomment the location of the Scotch installation folder and its␣
↪include dir
  SCOTCHDIR  = /home/jyepesr1/source/apolo/opensees-3.0.0_install/scotch_6.0.6
  ISCOTCH    = -I$(SCOTCHDIR)/include

  # Uncomment the parallel scotch libraries
  LSCOTCH    = -L$(SCOTCHDIR)/lib -lptesmumps -lptscotch -lptscotcherr -lscotch

  # Change and uncomment the location of the METIS installation folder and its␣
↪include dir
  LMETISDIR = /home/jyepesr1/source/apolo/opensees-3.0.0_install/parmetis-4.0.3/
↪metis
  IMETIS    = $(LMETISDIR)/include

  # Add the location of the ParMETIS folder
  LPARMETISDIR = /home/jyepesr1/source/apolo/opensees-3.0.0_install/parmetis-4.0.
↪3/
  IPARMETIS    = $(LMETISDIR)/include

  # Uncomment the METIS and ParMETIS libraries
  LMETIS    = -L$(LMETISDIR)/lib -lmetis
  LPARMETIS = -L$(LPARMETISDIR)/lib -lparmetis

  # Uncomment the following line and delete the next one
  ORDERINGSF = -Dscotch -Dmetis -Dpord -Dptscotch -Dparmetis

  # Modify the following variables adding the ParMETIS option
  LORDERINGS = $(LPARMETIS) $(LMETIS) $(LPORD) $(LSCOTCH)
  IORDERINGSC = $(IPARMETIS) $(IMETIS) $(IPORD) $(ISCOTCH)

  # Edit the LIBPAR variable to link against Intel MKL.
  # REMEMBER to load the module. module load mkl/2017_update-1
  # You can delete the other variables in that section, we will just need LIBPAR.
  LIBPAR =  $(MKLROOT)/lib/intel64/libmkl_blas95_ilp64.a $(MKLROOT)/lib/intel64/
↪libmkl_lapack95_ilp64.a \
  -L$(MKLROOT)/lib/intel64 -lmkl_scalapack_ilp64 -lmkl_intel_ilp64 -lmkl_
↪sequential -lmkl_core \
  -lmkl_blacs_intelmpi_ilp64 -lpthread -lm -ldl

  # At the end in the compiler flags for C and Fortran change -openmp for -qopenmp
  OPTF    = -O -DALLOW_NON_INIT -nofor_main -qopenmp
  OPTL    = -O -nofor_main -qopenmp
  OPTC    = -O -qopenmp
```

Note: If you want to use ScaLAPACK instead of Intel MKL, set the `LIBPAR` variable as:

```
-L/home/jyepesr1/source/apolo/scalapack/2.0.2-impi_17.0.1/lib -lscalapack -
↪lreflapack -lrefblas -ltmg
```

3. Compile and wait

```
$ make -j10
```

Scotch and PT-Scotch

Scotch and PT-Scotch are software packages and libraries for sequential and parallel graph partitioning, static mapping and clustering, sequential mesh and hypergraph partitioning, and sequential and parallel sparse matrix block ordering.[4]

1. Download and build **scotch** and **PT-Scotch**:

```
$ wget https://gforge.inria.fr/frs/download.php/file/37622/scotch_6.0.6.tar.gz
$ tar xf scotch_6.0.6.tar.gz
$ cd scotch_6.0.6/src
$ ln -s Make.inc/Makefile.inc.x86-64_pc_linux2.icc.impi Makefile.inc
```

2. Edit the `Makefile.inc` adding the directive `-DINTSIZE64` at the end of the `CFLAGS` variable to support integers of 64 bits.

```
CFLAGS  = -O3 -DCOMMON_FILE_COMPRESS_GZ -DCOMMON_PTHREAD -DCOMMON_RANDOM_FIXED_
↪SEED -DSCOTCH_RENAME -DSCOTCH_PTHREAD -restrict -DIDXSIZE64 -DINTSIZE64
```

3. Finally, compile the lib **ptesmumps**:

```
$ make -j10 ptesmumps
```

Note: The built libraries will be located in the `lib/` folder under the `scotch_6.0.6` folder

METIS and ParMETIS

METIS is a set of serial programs for partitioning graphs, partitioning finite element meshes, and producing fill reducing orderings for sparse matrices.[5]

ParMETIS is an MPI-based parallel library that implements a variety of algorithms for partitioning unstructured graphs, meshes, and for computing fill-reducing orderings of sparse matrices. ParMETIS extends the functionality provided by METIS and includes routines that are especially suited for parallel AMR computations and large scale numerical simulations.[6]

1. Download **ParMETIS** which include **METIS** and build both of them.

```
$ wget http://glaros.dtc.umn.edu/gkhome/fetch/sw/parmetis/parmetis-4.0.3.tar.gz
$ tar xf parmetis-4.0.3.tar.gz
$ cd parmetis-4.0.3
```

2. Edit the file `metis/include/metis.h` and specify 64 bits integers in the **IDXTYPEWIDTH** and **REALTYPEWIDTH** constants.

[4] Scotch & PT-Scotch - Scotch official site. Retrieved April 12, 2019, from http://www.labri.fr/perso/pelegrin/scotch/

[5] METIS - Serial Graph Partitioning and Fill-reducing Matrix Ordering - Karypis LAB. Retrieved April 12, 2019, from http://glaros.dtc.umn.edu/gkhome/metis/metis/overview

[6] ParMETIS - Parallel Graph Partitioning and Fill-reducing Matrix Ordering- Karypis LAB. Retrieved April 12, 2019, from http://glaros.dtc.umn.edu/gkhome/metis/parmetis/overview

```
#define IDXTYPEWIDTH 64
#define REALTYPEWIDTH 64
```

3. Load the CMake module to be able to build the source files.

```
$ module load cmake/3.7.1
```

4. Configure the ParMETIS installation as follows:

```
$ make config openmp=-qopenmp cc=mpiicc cxx=mpiicpc prefix=<install folder>
$ make -j10
$ make install
```

5. To build METIS, go to the `metis/` folder in the ParMETIS top directory and execute the following:

```
$ make config openmp=-qopenmp cc=mpiicc prefix=<install folder>
$ make -j10
$ make install
```

References

Authors

- Johan Sebastián Yepes Ríos <jyepesr1@eafit.edu.co>

3.3.49 Openstructure

Openstructure aims to provide an open-source, modular, flexible, molecular modelling and visualization environment. It is targeted at interested method developers in the field of structural bioinformatics. Open-Source Computational Structural Biology Framework.[1]

Openstructure 1.10

Table of Contents

Basic information

- **Official Website:** https://openstructure.org
- **Installed on:** *Apolo II*

[1] Retrieved 17:57, December 09, 2019, from https://openstructure.org/

Dependencies

- Cmake 3.7.1
- Boost 1.62.0
- Zlib 1.2.11
- FFTW 3.3.5
- Python 2.7.15
- Sqlite 3.30.1
- Libtiff 4.1.0
- Libpng 1.6.37
- Eigen 3.3.7
- GCC 5.4.0

Installation

1. Get source code.

```
$ git clone https://git.scicore.unibas.ch/schwede/openstructure.git
```

2. Load the dependences of Openstructure (It varies depending on the user's needs).

```
$ module load cmake/3.7.1
$ module load boost/1.62.0_gcc-5.4.0_openmpi-1.8.8-x86_64
$ module load zlib/1.2.11_gcc-5.4.0
$ module load fftw/3.3.5_gcc-5.4.0_openmpi-1.8.8-x86_64
$ module load python/2.7.15_miniconda-4.5.4
$ module load sqlite/3.30.1
$ module load libtiff/4.1.0_intel-19.0.4
$ module load libpng/1.6.37
$ module load eigen/3.3.7_intel-19.0.4
```

3. Run the cmake according to the dependencies needed during the compilation.

```
$ CXXFLAGS="-fPIC -O3" cmake .. -DENABLE_GFX=OFF -DENABLE_INFO=OFF -DENABLE_
↪IMG=ON -DPYTHON_ROOT=/share/apps/python/2.7_miniconda-4.5.4 -DPYTHON_LIBRARIES=/
↪share/apps/python/2.7_miniconda-4.5.4/lib -DFFTW_LIBRARY=$FFTW_LIBRARY/
↪libfftw3f.so -DFFTW_INCLUDE_DIR=$FFTW_INCLUDE_DIR -DBOOST_ROOT=$BOOST_ROOT -
↪DEIGEN3_INCLUDE_DIR=$EIGEN_HOME/include/eigen3 -DSQLITE3_LIBRARY=$SQLITE_HOME/
↪lib/libsqlite3.so.0.8.6 -DSQLITE3_INCLUDE_DIR=$SQLITE_HOME/include -DTIFF_
↪LIBRARY=$LIBTIFF_HOME/lib/libtiff.so -DTIFF_INCLUDE_DIR=$LIBTIFF_HOME/include -
↪DPNG_LIBRARY=$LIBPNG_HOME/lib/libpng.so -DPNG_INCLUDE_DIR=$LIBPNG_HOME/include -
↪DZLIB_LIBRARY=$ZLIB_HOME/lib/libz.so -DZLIB_INCLUDE_DIR=$ZLIB_HOME/include -
↪DPREFIX=/share/apps/openstructure/1.10/gcc-5.4.0
```

4. Compile and install openstructure

```
$ make
$ make install
```

5. Create the corresponding module of Openstructure 1.10.

```
$ mkdir /share/apps/modules/openstructure
$ vim /share/apps/modules/openstructure/1.10_gcc-5.4.0

#%Module1.0#####################################################################
↪######
##
## modulefile /share/apps/openstructure/1.10/gcc-5.4.0/
##

proc ModulesHelp { } {
     global version modroot
          puts stderr "\t Openstructure 1.10"
}

module-whatis "\n\n\tSets the environment for using Openstructure 1.10 \n"


set     topdir          /share/apps/openstructure/1.10/gcc-5.4.0
set     version         1.10
set     sys             x86_64-redhat-linux

module load cmake/3.7.1
module load boost/1.62.0_gcc-5.4.0_openmpi-1.8.8-x86_64
module load zlib/1.2.11_gcc-5.4.0
module load fftw/3.3.5_gcc-5.4.0_openmpi-1.8.8-x86_64
module load python/2.7.15_miniconda-4.5.4
module load sqlite/3.30.1
module load libtiff/4.1.0_intel-19.0.4
module load libpng/1.6.37
module load eigen/3.3.7_intel-19.0.4

prepend-path PATH                       $topdir/bin
prepend-path PYTHONPATH                 $topdir/lib64/python2.7/site-
↪packages

prepend-path C_INCLUDE_PATH             $topdir/include
prepend-path CXX_INCLUDE_PATH           $topdir/include
prepend-path CPLUS_INCLUDE_PATH         $topdir/include

prepend-path LD_LIBRARY_PATH            $topdir/lib64
prepend-path LIBRARY_PATH               $topdir/lib64
prepend-path LD_RUN_PATH                $topdir/lib64
```

Authors

- Juan Diego Ocampo García <jocamp18@eafit.edu.co>

3.3.50 ParGenes

ParGenes[1] a massively parallel tool for model selection and tree inference on thousands of genes,[2] aimed to make the iference of gene tree easier.[3][4]

ParGenes master

Table of Contents

Basic information

- **Official Website:** https://github.com/BenoitMorel/ParGenes
- **Download Website:** https://github.com/BenoitMorel/ParGenes.git
- **License:** GNU General Public License v3.0
- **Installed on:** Apolo II

Tested on (Requirements)

- **OS base:** CentOS (x86_64) $\geq$ 6.6 (Rocks 6.2)
- **Dependencies to run ParGenes:**
  - GCC 5.0 > (tested on version 5.4.0)
  - Python 2.7 or 3.X (tested on Python 3.6.5)
  - CMAKE > 3.6 (tested on version 3.7.1)
  - mpich2 (tested on version 3.2)

[1] Morel, B. (2019, October 18). BenoitMorel/ParGenes. Retrieved from https://github.com/BenoitMorel/ParGenes.
[2] Morel, B. (n.d.). BenoitMorel/ParGenes. Retrieved from https://github.com/BenoitMorel/ParGenes/wiki.
[3] Benoit Morel, Alexey M. Kozlov, Alexandros Stamatakis (2018) ParGenes: a tool for massively parallel model selection and phylogenetic tree inference on thousands of genes. Bioinformatics. https://doi.org/10.1093/bioinformatics/bty839
[4] Alexey M. Kozlov, Diego Darriba, Tomáš Flouri, Benoit Morel, and Alexandros Stamatakis (2018) RAxML-NG: A fast, scalable, and user-friendly tool for maximum likelihood phylogenetic inference. bioRxiv. https://doi.org/10.1101/447110

Installation

The following procedure is the easiest way to install ParGenes (master git version) in a cluster.

1. Clone the git repository of ParGenes in a cluster location (we are going to use the `$HOME` directory).

```
$ git clone --recursive https://github.com/BenoitMorel/ParGenes.git
```

Note: ParGenes at the moment only runs via `.py` files, therefore is not posible to install it properly in a cluster. Only the user that clones the repo can run ParGenes.

2. Load the dependencies so ParGenes will be able to run, and check the GCC version.

```
$ module load mpich2/3.2_gcc-5.4.0 python/3.6.5_miniconda-4.5.1 cmake/3.7.
↪1
$ gcc --version
gcc (GCC) 5.4.0
```

3. Now, for being able to run ParGenes please follow these commands:

```
$ cd ParGenes
$ ./install.sh
```

Now you have ParGenes installed inside your `$HOME` directory.

Running Example

In this section, there is an example run that ParGenes already has.

1. First, we create a conda environment, so we can run ParGenes in a secure environment (it is not necessary but we recommend it).

```
$ conda create --name pargenes
$ conda activate pargenes
$ cd examples/data/small/
$ python ../../../pargenes/pargenes-hpc.py -a fasta_files/ -o output_dir -c 32 -d␣
↪nt -R "--model GTR"
```

Note: Make sure to load every module from the beginning, specially mpich2.

If everything runs without errors, you have installed ParGenes successfully in your `$HOME` directory

2. **This is an example for Apolo in SLURM.**

```
#!/bin/bash

#SBATCH --partition=longjobs
#SBATCH --nodes=2
#SBATCH --ntasks-per-node=1
#SBATCH --cpus-per-task=16
#SBATCH --time=1:00:00
#SBATCH --job-name=ParGenes
#SBATCH --partition=longjobs
```

(continues on next page)

(continued from previous page)

```
#SBATCH -o result_%N_%j.out
#SBATCH -e result_%N_%j.err

module load mpich2/3.2_gcc-5.4.0 python/3.6.5_miniconda-4.5.1 cmake/3.7.1

python ~/scripts/ParGenes/pargenes/pargenes-hpc.py -a ~/Bacillus_subtilis/
↪ParGenes_data/mix_msa -o ~/output_dir_slurm3 -c 32 -d nt -b 10000 -R "--
↪model GTR"
```

For more information on how to use ParGenes, please visit the official website.

References

ParGenes - ParGenes Official website. Retrieved Octubre 4, 2019, from https://github.com/BenoitMorel/ParGenes/wiki

Installation - ParGenes Official Website. Retrieved Octubre 4, 2019, from https://github.com/BenoitMorel/ParGenes#installation

Authors

- Tomas David Navarro Munera <tdnavarrom@eafit.edu.co>

3.3.51 Partition Finder

PartitionFinder2 is a program for selecting best-fit partitioning schemes and models of evolution for nucleotide, amino acid, and morphology alignments. The user provides an alignment, and optionally some pre-defined data blocks (e.g. 9 data blocks defining the 1st, 2nd and 3rd codon positions of 3 protein-coding genes, see Figure 1). The program then finds the best partitioning scheme for this dataset, at the same time as selecting best-fit models for each subset of sites/columns. Here are a few things you can do with the program:

1. Find the best-fit partitioning scheme nucleotide, amino acid, or morphology datasets
2. Compare any number of user-defined partitioning schemes
3. Find best-fit models of evolution for each subset in any partitioned dataset (much like you might do with ModelTest or ProtTest).

Partition Finder 2.1.1

Basic Information

- **Deploy date:** 5 December 2016
- **Official Website:** http://www.robertlanfear.com/partitionfinder/
- **License:** GNU General Public License
- **Installed on:** *Apolo II*

Installation

This entry covers the entire process performed for the installation and configuration of Partition Finder in a Cluster with Conda.

1. Create a Conda Environment for Partition Finder

```
conda create -n partitionFinder
```

1. Install the following dependencies (numpy, pandas, pytables, pyparsing, scipy, and sklearn).

```
conda install numpy pandas pytables pyparsing scipy scikit-learn
```

1. Download the latest version of Partition Finder and decompress it.

```
wget https://github.com/brettc/partitionfinder/archive/v2.1.1.tar.gz
tar xfz v2.1.1.tar.gz
```

1. Move it to wherever you want to store and give it execution permissions.

```
mv partitionfinder-2.1.1 /path/to/partition/finder
chmod +x partitionfinder-2.1.1/PartitionFinder.py
```

Usage

This section describes the method to submit jobs with the resource manager SLURM.

Note: If it is the first time you need Partition Finder or you want to use it locally, you should create and load the environment.

```
conda env create -f partitionFinder.yml
```

Listing 52: `Partition Finder environment`

```
name: partitionFinder
channels:
- file:///share/apps/intel/ps_xe/2019_update-4/conda_channel
- /share/apps/intel/ps_xe/2019_update-4/conda_channel
- intel
- defaults
dependencies:
- _libgcc_mutex=0.1=main
- atomicwrites=1.3.0=py27_1
- attrs=19.3.0=py_0
- blosc=1.16.3=hd408876_0
- contextlib2=0.6.0.post1=py_0
- hdf5=1.10.4=hb1b8bf9_0
- importlib_metadata=0.23=py27_0
- libgcc-ng=9.1.0=hdf63c60_0
- libgfortran-ng=7.3.0=hdf63c60_0
- libstdcxx-ng=9.1.0=hdf63c60_0
- lz4-c=1.8.1.2=h14c3975_0
- lzo=2.10=h49e0be7_2
- mock=1.0.1=py27_0
- more-itertools=5.0.0=py27_0
```

(continues on next page)

(continued from previous page)

```
- packaging=19.2=py_0
- pluggy=0.13.0=py27_0
- py=1.8.0=py_0
- pytables=3.5.2=py27h71ec239_1
- pytest=4.6.2=py27_0
- snappy=1.1.7=hbae5bb6_3
- zipp=0.6.0=py_0
- zstd=1.3.7=h0b5b093_0
- backports=1.0=py27_9
- backports.functools_lru_cache=1.5=py27_2
- bzip2=1.0.6=17
- certifi=2018.1.18=py27_2
- cycler=0.10.0=py27_7
- daal=2019.4=intel_243
- daal4py=2019.4=py27h7b7c402_0
- freetype=2.9=3
- funcsigs=1.0.2=py27_7
- functools32=3.2.3.2=py27_7
- icc_rt=2019.4=intel_243
- impi_rt=2019.4=intel_243
- intel-openmp=2019.4=intel_243
- intelpython=2019.4=0
- kiwisolver=1.0.1=py27_2
- libpng=1.6.36=2
- matplotlib=2.2.4=py27_1
- mkl=2019.4=intel_243
- mkl_fft=1.0.11=py27h7b7c402_2
- mkl_random=1.0.2=py27h7b7c402_4
- numexpr=2.6.8=py27_2
- numpy=1.16.2=py27h7b7c402_0
- numpy-base=1.16.2=py27_0
- openssl=1.0.2r=2
- pandas=0.24.1=py27_3
- pip=10.0.1=py27_0
- pyparsing=2.2.0=py27_2
- python=2.7.16=3
- python-dateutil=2.6.0=py27_12
- pytz=2018.4=py27_3
- scikit-learn=0.20.3=py27h7b7c402_5
- scipy=1.2.1=py27h7b7c402_3
- six=1.11.0=py27_3
- sqlite=3.27.2=4
```

1. Run SLURM with the following bash file.

```
sbatch partitionFinder.sh
```

Listing 53: `Partition Finder SLURM`

```
#!/bin/bash
↪

#SBATCH --job-name=serial_test        # Job name
↪
#SBATCH --mail-type=ALL          # Mail notification
↪
```

(continues on next page)

(continued from previous page)

```
#SBATCH --mail-user=<user>@<domain>  # User Email
#SBATCH --output=mothur-%j.out # Stdout (%j expands to jobId)
#SBATCH --error=mothur-%j.err  # Stderr (%j expands to jobId)
#SBATCH --ntasks=1                   # Number of tasks (processes)
#SBATCH --time=01:00                 # Walltime
#SBATCH --partition=longjobs         # Partition


##### ENVIRONMENT CREATION #####

module load python
source activate partitionFinder

##### JOB COMMANDS ####
/path/to/partition/finder/PartitionFinder.py /path/to/partitionFinder/input/files
```

Authors

- Manuela Carrasco Pinzón <mcarras1@eafit.edu.co>

3.3.52 PhyloNet

PhyloNet is a tool designed mainly for analyzing, reconstructing, and evaluating reticulate (or non-treelike) evolutionary relationships, generally known as phylogenetic networks. Various methods that we have developed make use of techniques and tools from the domain of phylogenetic networks, and hence the PhyloNet package includes several tools for phylogenetic network analysis. PhyloNet is released under the GNU General Public License. For the full license, see the file GPL.txt included with this distribution.

PhyloNet is designed, implemented, and maintained by Rice's BioInformatics Group, which is lead by Professor Luay Nakhleh (nakhleh@cs.rice.edu). For more details related to this group please visit http://bioinfo.cs.rice.edu.

PhyloNet3.8.0

Table of Contents

Basic information

- **Official Website:** https://bioinfocs.rice.edu/phylonet
- **License:** GNU General Public License
- **Installed on:** *Apolo II*
- **Installation date:** 22/07/2021

Tested on (Requirements)

- **Dependencies:**
  - Java JDK - 1.8.0 u112

Installation

1. Download the jar file from the official website

```
cd /share/apps/phylonet/3.8.0
wget https://bioinfocs.rice.edu/sites/g/files/bxs266/f/kcfinder/files/
↪PhyloNet_3.8.0.jar
```

2. After downloading, rename the file to `PhyloNet.jar`:

```
mv PhyloNet_3.8.0.jar PhyloNet.jar
```

3. Install the corresponding module file:

Module

```
##%Module1.0#####################################################################
↪###
##
## module /share/apps/modules/phylonet/3.8.0
##
## /share/apps/phylonet/3.8.0 Written by Juan Pablo Ossa Zapata
##

proc ModulesHelp { } {
    puts stderr "\tphylonet/3.8.0 - sets the Environment for Phylonet \
    \n\tin the share directory /share/apps/phylonet/3.8.0 \n"
}

module-whatis "\n\n\tSets the environment for using phylonet 3.8.0\n"
```

(continues on next page)

(continued from previous page)

```
# for Tcl script use only
set     topdir          /share/apps/phylonet/3.8.0
set     version         3.8.0
set     sys             linux-x86_64

conflict phylonet

module load java/jdk-1.8.0_112

prepend-path PHYLONET_DIRECTORY               $topdir
```

Use

```
module load phylonet/3.8.0
```

Example slurm job file:

```
#!/bin/sh
#SBATCH --partition=longjobs
#SBATCH --nodes=1
#SBATCH --cpus-per-task=4
#SBATCH --ntasks=1
#SBATCH --time=2:00:00
#SBATCH --job-name=testPhylonet
#SBATCH -o %x_%j.out      # File to which STDOUT will be written
#SBATCH -e %x_%j.err      # File to which STDERR will be written
#SBATCH --mail-type=ALL
#SBATCH --mail-user=jpossaz@eafit.edu.co

module load phylonet/3.8.0

export OMP_NUM_THREADS=4

srun java -jar $PHYLONET_DIRECTORY/PhyloNet.jar mynexusfile
```

Make sure that your nexus file is able to use all of the allocated cpu cores.

Resources

- https://wiki.rice.edu/confluence/pages/viewpage.action?pageId=39500205#PhylonetTutorial%28SSB2020%29-2.Installation
- https://bioinfocs.rice.edu/phylonet

Author

- Juan Pablo Ossa Zapata

3.3.53 PICRUSt2

PICRUSt:[1] Phylogenetic Investigation of Communities by Reconstruction of Unobserved States. PICRUSt (pronounced "pie crust") is a bioinformatics software package designed to predict metagenome functional content from marker gene (e.g., 16S rRNA) surveys and full genomes.

Versions

PICRUSt 2.0.4-BETA

Table of Contents

Basic information

- **Official Website:** https://github.com/picrust/picrust2/wiki
- **License:** GNU General Public License v3.0
- **Tested on** CentOS (x86_64) 6.6 (Rocks 6.2)

Installation

Note: Official installation instructions can be found in https://github.com/picrust/picrust2/blob/master/INSTALL.md.

You should follow the following procedure in order to install this software on a conda environment, without root privileges.

1. Load conda module. On *Apolo II* is python/3.6.5_miniconda-4.5.1, on *Cronos* python/3.6.5_miniconda-4.5.4.

```
$ module load <conda_module>
```

Note: Be aware you can also try out the intel optimized conda modules, which are proved to increase performance in several applications: on *Apolo II* is python/3.5.2_intel-2017_update-1, on *Cronos* is python/3.6.2_intel-18_u1.

2. Clone PICRUSt2 repository and cd into it:

```
$ git clone https://github.com/picrust/picrust2.git
$ cd picrust2
```

[1] PYCRUSt. Retrieved January 24th, 2019, from http://picrust.github.io/picrust/.

Note: All the following commands should be executed inside the "picrust2" directory!.

3. Install the required dependencies into a conda environment. This will create a conda environment named picrust2 in which the dependencies will be installed:

```
$ conda env create -f picrust2-env.yaml
```

4. Activate the conda environment and finish installing PICRUSt2:

```
$ source activate picrust2
$ pip install --no-deps --editable .
```

5. Finally, test the installation.

```
$ pytest
```

Authors

- Vincent Alejandro Arcila Larrea (vaarcilal@eafit.edu.co).

3.3.54 Pigz

Pigz[1] A parallel implementation of gzip for modern multi-processor, multi-core machines.

Pigz 2.4

Basic information

- **Deploy date:** 22 October 2019
- **Official Website:** https://zlib.net/pigz/
- **License:** `Custom license`
- **Installed on:** *Apolo II*

Installation

1. Run:

```
$ wget https://zlib.net/pigz/pigz-2.4.tar.gz
$ tar xvf pigz-2.4.tar.gz
$ cd pigz-2.4
```

The compiler used here is Intel 2019, therefore, modify the makefile like this:

```
CC=icc
CXX=icpc
CFLAGS=-O3 -xHost -ipo -Wall -Wextra -Wno-unknown-pragmas
```

[1] Pigz. (2019, Dec 3). Retrieved December 3, 2019, from https://zlib.net/pigz/

Then compile:

```
$ module load intel/19.0.4
$ make -j4
```

Install it:

```
$ sudo mkdir -p /share/apps/pigz/2.4/intel/19.0.4/bin
$ sudo cp pigz unpigz /share/apps/pigz/2.4/intel/19.0.4/bin
```

2. Create and place the needed module file. Create a file with the following content:

Listing 54: `2.4_intel-19.0.4`

```
##%Module1.0#####################################################################
↪#####
##
## module load pigz/2.4_intel-19.0.4
##
## /share/apps/modules/pigz/2.4_intel-19.0.4
## Written by Hamilton Tobon Mosquera.
##

proc ModulesHelp {} {
        global version modroot
        puts stderr "Sets the environment for using Parallel Gzip 2.4, \
        \nin the shared directory /share/apps/pigz/2.4/intel/19.0.4, \
        \nbuilt with Intel 19.0.4.\n"
}

module-whatis "(Name________) pigz"
module-whatis "(Version_____) 2.4"
module-whatis "(Compilers___) intel-19.0.4"
module-whatis "(System______) x86_64-redhat-linux"
module-whatis "(Libraries___) "

# for Tcl script use only
set         topdir        /share/apps/pigz/2.4/intel/19.0.4
set         version       2.4
set         sys           x86_64-redhat-linux

conflict pigz

prepend-path   PATH            $topdir/bin
```

Create the needed folder and place it:

```
$ sudo mkdir /share/apps/modules/pigz
$ sudo mv 2.4_intel-19.0.4 /share/apps/modules/pigz/
```

Authors

- Hamilton Tobon-Mosquera <htobonm@eafit.edu.co>

3.3.55 Plumed

PLUMED is an open source library for free energy calculations in molecular systems which works together with some of the most popular molecular dynamics engines. Free energy calculations can be performed as a function of many order parameters with a particular focus on biological problems, using state of the art methods such as metadynamics, umbrella sampling and Jarzynski-equation based steered MD. The software, written in C++, can be easily interfaced with both fortran and C/C++ codes.

PLUMED 2.3.5

Table of Contents

Basic information

- **Installation Date:** 22/02/2017
- **URL:** http://www.plumed.org/
- **License:** GNU LESSER GENERAL PUBLIC LICENSE Version 3
- **Installed on:** Apolo II

Tested on (Requirements)

- **Dependencies to run PLUMED:**
  - GNU GCC >= 5.4.0
  - Mpich2 >= 3.2
  - OpenBLAS >= 0.2.19
  - GSL >= 2.4
  - libmatheval >= 1.1.11

Installation

After solving the previously mentioned dependencies, you can proceed with the installation of **Plumed**.

1. Download the latest version of the software, in this case, the latest version is 2.3.5 of line 2.3.x (Source code - tgz) (http://www.plumed.org/get-it):

```
cd /home/mgomezzul/apps/plumed/src/gcc-5.4.0
# Descargar el .tgz en este directorio
tar -xf plumed-2.3.5.tgz
```

2. After decompressing **Plumed**, we continue with the following steps for its configuration and compilation:

```
module load libmatheval/1.1.11 gsl/2.4_gcc-5.4.0 openblas/0.2.19_gcc-5.4.
↪0 mpich2/3.2_gcc-5.4.0
cd plumed-2.3.5
sudo mkdir -p /share/apps/plumed/2.3.5/
sudo chown -R mgomezzul.apolo /share/apps/plumed/2.3.5
./configure CC=mpicc CXX=mpic++ FC=mpif90 --prefix=/share/apps/plumed/2.3.
↪5 LIBS="-lopenblas" 2>&1 | tee plumed-conf.log
make -j 16 2>&1 | tee plumed-make.log
make check 2>&1 | tee plumed-make-check.log
make install 2>&1 | tee plumed-make-install.log
sudo chown -R root.root /share/apps/plumed/2.3.5
```

Module

```
##%Module1.0#####################################################################
↪##
##
## module load plumed/2.3.5
##
## /share/apps/modules/plumed/2.3.5
## Written by Mateo Gómez-Zuluaga
##

proc ModulesHelp {} {
    global version modroot
    puts stderr "Sets the environment for using plumed 2.3.5\
        \nin the shared directory /share/apps/plumed/2.3.5\
        \nbuilded with gcc-5.4.0, mpich2-3.2, openblas-0.2.19\
        \ngsl-2.4 and libmatheval-1.11.0."
}

module-whatis "(Name________) plumed"
module-whatis "(Version_____) 2.3.5"
module-whatis "(Compilers___) gcc-5.4.0_mpich2-3.2"
module-whatis "(System______) x86_64-redhat-linux"
module-whatis "(Libraries___) opeblas-0.2.19, libmatheval-1.11.0, gsl-2.4"

# for Tcl script use only
set         topdir        /share/apps/plumed/2.3.5
set         version       2.3.5
set         sys           x86_64-redhat-linux

conflict plumed

module load mpich2/3.2_gcc-5.4.0
module load openblas/0.2.19_gcc-5.4.0
module load gsl/2.4_gcc-5.4.0
```

(continues on next page)

(continued from previous page)

```
module load libmatheval/1.1.11


prepend-path    PATH                    $topdir/bin

prepend-path    LD_LIBRARY_PATH         $topdir/lib
prepend-path    LIBRARY_PATH            $topdir/lib
prepend-path    LD_RUN_PATH             $topdir/lib
prepend-path    DYLD_LIBRARY_PATH       $topdir/lib

prepend-path    C_INCLUDE_PATH          $topdir/include
prepend-path    CXX_INCLLUDE_PATH       $topdir/include
prepend-path    CPLUS_INCLUDE_PATH      $topdir/include

setenv          PLUMED_KERNEL           $topdir/lib/libplumedKernel.so
```

Mode of Use

- The use of **Plumed** in this case is limited to patching the **Gromacs** source code for this MD to use **Plumed** for physical handling.

```
module load wrf/3.7.1_gcc-5.4.0
```

References

- http://www.plumed.org
- https://plumed.github.io/doc-v2.3/user-doc/html/_installation.html
- https://plumed.github.io/doc-v2.3/user-doc/html/gromacs-5-1-4.html
- http://www.jyhuang.idv.tw/JYH_ComputingPackages.html
- http://pdc-software-web.readthedocs.io/en/latest/software/plumed/centos7/2.3b/
- https://plumed.github.io/doc-v2.4/user-doc/html/_g_m_x_g_p_u.html

Authors

- Mateo Gómez Zuluaga

3.3.56 Prank

PRANK is a probabilistic multiple alignment program for DNA, codon and amino-acid sequences. It's based on a novel algorithm that treats insertions correctly and avoids over-estimation of the number of deletion events. In addition, PRANK borrows ideas from maximum likelihood methods used in phylogenetics and correctly takes into account the evolutionary distances between sequences. Lastly, PRANK allows for defining a potential structure for sequences to be aligned and then, simultaneously with the alignment, predicts the locations of structural units in the sequences.[1]

[1] http://wasabiapp.org/software/prank/

Prank 170427

Table of Contents

Basic information

- **Official Website:** http://wasabiapp.org/software/prank/
- **Downloads page:** http://wasabiapp.org/download/prank/
- **Installed on:** APOLO II

Tested on (Requirements)

- **OS base:** CentOS (x86_64) $\geq$ 6
- **Compiler** Intel 19.0.4

Installation

1. First of all, you must load the following modules for the compilation.

```
$ module load intel/19.0.4
```

2. After that, download the source code from github and move into the src directory generated.

```
$ git clone https://github.com/ariloytynoja/prank-msa.git
$ cd prank-msa/src
```

3. Before starting the installation, you should make some changes, in the Makefile in order to be able to compile Prank with ICC and ICPC support, so open your trusted text editor and edit the file.

```
$ # For this case I'll use vim
$ vim Makefile
```

4. At the top of the Makefile, you'll find a section titled "Compiler, tools and options". There, there are some variables such as CC, CXX, CFLAGS, CXXFLAGS, etc. Those variables should look like the following after the changes.

```
CC            = icc
CXX           = icpc
DEFINES       = -xHost -Ofast -pipe
CFLAGS        = $(DEFINES)
```

(continues on next page)

(continued from previous page)

```
CXXFLAGS        = $(DEFINES)
LINK            = icpc
AR              = xiar cqs
```

5. Then you can continue with the installation.

```
$ make -j
```

6. If the installation was successful then you should:

```
$ ./prank -version
```

7. Optional: If you want, you can add the Prank binary to your $PATH. For this, is highly recommended to move just the binary file to another location and add it to your $PATH

```
$ mkdir -p $HOME/prank-bin
$ cp prank $HOME/prank-bin
$ cd $HOME/prank-bin
$ ./prank -version
$ # From here you can follow some tutorial about how to add something to␣
↪your $PATH
```

Authors

- Laura Sanchez Cordoba
- Samuel Palacios Bernate

3.3.57 Prinseq-lite

PR eprocessing and **IN** formation of **SEQ** uence data. **PRINSEQ** will help you to preprocess your genomic or metagenomic sequence data in FASTA or FASTQ format.

The lite version is a standalone perl script (prinseq-lite.pl) that does not require any non-core perl modules for processing.

The used modules are:

```
Getopt::Long Pod::Usage
File::Temp qw(tempfile)
Fcntl qw(:flock SEEK_END)
Digest::MD5 qw(md5_hex)
Cwd List::Util qw(sum min max)
```

Also PRINSEQ is available in web and graphics version.

For additional information you can open those links:

- **Main page:** http://prinseq.sourceforge.net/
- **More information** https://sourceforge.net/projects/prinseq/files/?source=navbar
- **Download** https://sourceforge.net/projects/prinseq/files/standalone/prinseq-lite-0.20.4.tar.gz/download

Prinseq-lite 0.20.4

Table of Contents

Basic information

- **Installation Date:** 11/07/2018
- **URL:** http://prinseq.sourceforge.net
- **License:** GNU GENERAL PUBLIC LICENSE Version 3
- **Installed on:** Apolo II & Cronos

Tested on (Requirements)

- **Dependencies to run Prinseq-lite:**
  - Perl >= 5.26.1

Installation

1. We get the **Prinseq-lite** binary from the official website.

```
wget https://sourceforge.net/projects/prinseq/files/standalone/prinseq-
↪lite-0.20.4.tar.gz/download
mv download prinseq-lite-0.20.4.tar.gz
tar -xzvf prinseq-lite-0.20.4.tar.gz
cd prinseq-lite-0.20.4
```

2. We create the installation folder and copy the script, changing its name

```
mkdir -p /share/apps/prinseq-lite/0.20.4/bin
cp prinseq-lite.pl /share/apps/prinseq-lite/0.20.4/bin/prinseq-lite
```

3. We assign the execution permissions

```
chmod +x /share/apps/prinseq-lite/0.20.4/bin/prinseq-lite
```

Module

```
#%Module1.0#####################################################################
↪##
##
## module load prinseq-lite/0.20.4
##
## /share/apps/modules/prinseq-lite/0.20.4
## Written by Juan David Arcila-Moreno
##

proc ModulesHelp {} {
    global version modroot
    puts stderr "Sets the environment for using prinseq-lite 0.20.4\
        \nin the shared directory \
        \n/share/apps/prinseq-lite/0.20.4\
        \nperl script"
}

module-whatis "(Name________) prinseq-lite"
module-whatis "(Version_____) 0.20.4"
module-whatis "(Compilers___) perl-5.26.1"
module-whatis "(System______) x86_64-redhat-linux"
module-whatis "(Libraries___) "

# for Tcl script use only
set         topdir        /share/apps/prinseq-lite/0.20.4
set         version       0.20.4
set         sys           x86_64-redhat-linux

conflict prinseq-lite
module load perl/5.26.1_gcc-5.5.0


prepend-path    PATH                    $topdir/bin
```

Mode of Use

```
module load prinseq-lite
prinseq-lite -h
```

References

- https://sourceforge.net/projects/prinseq/

Authors

- Juan David Arcila Moreno

pteros:

3.3.58 Pteros

Pteros[1] is a C++ library for molecular modeling. It is designed to simplify the development of custom programs and scripts for molecular modeling, analysis of molecular dynamics trajectories and implementing new simulation and analysis algorithms. Pteros provides facilities, which are routinely used in all molecular analysis programs, namely input/output of popular file formats, powerful and flexible atom selections, geometry transformations, RMSD fitting and alignment, etc. Pteros also contains powerful facilities for parsing command-line arguments in custom programs and for running several analysis tasks in parallel, utilizing the power of modern multi-core processors.

Pteros supports writing analysis programs in either C++ or Python programming languages.

Pteros 2.0

Table of Contents

Basic information

- **Official Website:** https://yesint.github.io/pteros/index.html
- **Installed on:** *Apolo II*

Tested on (Requirements)

- **OS base:** CentOS (x86_64) $\geq$ 6.6 (Rocks 6.2)
- **Dependencies:**
  - Standard-conforming C and C++ compilers (tested with gcc= 5.4, g++= 5.4).
  - cmake 3.3.7 build system
  - Eigen 3.3.7
  - Boost 1.62.0

[1]

- Semen O. Yesylevskyy, "Pteros 2.0: Evolution of the fast parallel molecular analysis library for C++ and python", Journal of Computational Chemistry, 2015, 36(19), 1480–1488, doi: 10.1002/jcc.23943. (https://onlinelibrary.wiley.com/doi/abs/10.1002/jcc.23943)
- Semen O. Yesylevskyy, "Pteros: Fast and easy to use open-source C++ library for molecular analysis", Journal of Computational Chemistry, 2012, 33(19), 1632–1636. doi: 10.1002/jcc.22989. (https://onlinelibrary.wiley.com/doi/full/10.1002/jcc.22989)

- Pybind11
- Git for getting the source code

Installation

1. Download the latest version of Pteros

```
$ git clone https://github.com/yesint/pteros.git pteros
```

2. Inside the folder, on the top create a `build` directory where the installation binaries will be put by cmake.

```
$ cd pteros
$ mkdir build
$ cd build
```

3. Load the necessary modules for the building.

```
$ module load cmake/3.7.1
$ module load gcc/5.4.0
$ module load boost/1.62.0_gcc-5.4.0_openmpi-1.8.8-x86_64
$ module load eigen/3.3.7_intel-2017_update-1
$ module load pybind11/11_gcc-5.4.0
```

4. Execute the cmake command with the desired directives.

```
$ cmake .. -DCMAKE_INSTALL_PREFIX=/share/apps/pteros/2.0/gcc-5.4.0/ -
↪DEIGEN3_INCLUDE_DIR=/share/apps/eigen/3.3.7/intel-19.0.4/include/eigen3/
↪ -DCMAKE_C_COMPILER=gcc -DCMAKE_CXX_COMPILER=g++ -DCMAKE_BUILD_
↪TYPE=Release -DBoost_NO_SYSTEM_PATHS=OFF -DWITH_OPENBABEL=OFF -DWITH_
↪GROMACS=OFF -DWITH_PYTHON=ON -DPYTHON_EXECUTABLE:FILEPATH=/share/apps/
↪python/3.6_miniconda-4.5.1/bin/python -Dpybind11_DIR=/share/apps/
↪pybind11/gcc-5.4.0/share/cmake/pybind11
```

5. Execute the make commands sequence.

```
$ make -j <N>
$ make -j install
```

Module

```
#%Module1.0#####################################################################
↪###
##
## modulefile /share/apps/modules/pteros/2.0_gcc-5.4.0
## Written by Juan Diego Ocampo and Santiago Hidalgo Ocampo
##

proc ModulesHelp { } {
    global version modroot
        puts stderr "\t Pteros 2.0."
}

module-whatis "(Name________) Pteros"
```

(continues on next page)

(continued from previous page)

```
module-whatis "(Version_____) 2.0"
module-whatis "(Compilers___) gcc-5.4.0"
module-whatis "(System______) x86_64-redhat-linux"

set     topdir          /share/apps/pteros/2.0/gcc-5.4.0
set     version         2.0
set     sys             x86_64-redhat-linux

module load boost/1.62.0_gcc-5.4.0_openmpi-1.8.8-x86_64
module load python/3.6.5_miniconda-4.5.1
module load eigen/3.3.7_intel-19.0.4
module load pybind11/11_gcc-5.4.0

prepend-path PATH                       $topdir/bin
prepend-path PYTHONPATH                 $topdir/python

prepend-path C_INCLUDE_PATH             $topdir/include
prepend-path CXX_INCLUDE_PATH           $topdir/include
prepend-path CPLUS_INCLUDE_PATH         $topdir/include

prepend-path LD_LIBRARY_PATH            $topdir/lib
prepend-path LIBRARY_PATH               $topdir/lib
prepend-path LD_RUN_PATH                $topdir/lib
prepend-path LD_LIBRARY_PATH            $topdir/lib64
prepend-path LIBRARY_PATH               $topdir/lib64
prepend-path LD_RUN_PATH                $topdir/lib64
```

Testing Installation

Run the following command:

```
$ pteros_analysis.py --help all
```

See also:

To use pteros you must have the numpy library, so we suggest following the next steps:

```
$ conda create -n pteros # Create a virtual environment
$ conda activate pteros
$ conda install numpy
```

Warning: Some commands may fail, however, the application may work with the features you need

Troubleshooting

See also:

If you have this problem: ModuleNotFoundError: No module named '_pteros', probably you must rename this file: <path to Pteros>/python/pteros/_pteros.cpython-37m-x86_64-linux-gnu.so to _pteros.so

Authors

- Santiago Hidalgo Ocampo <shidalgoo1@eafit.edu.co>

3.3.59 pyrad

Description

This program is designed to analyze data of the Radseq type with the use of an alignment-grouping method that allows you to include "indel" variation which improves the identification of homology through highly divergent samples.

Table of Contents

Pyrad

- **Installation date:** 06/02/2012
- **URL:** http://dereneaton.com/software/pyrad/
- **Apolo version:** Apolo II
- **License:** GPLv3

Dependencies

- python
- numpy
- scipy

Installation

1. First download the tar from the main page

```
$ wget https://github.com/dereneaton/pyrad/archive/3.0.66.tar.gz
$ tar -zxvf 3.0.66.tar.gz
```

1. Software config

```
cd pyrad-3.0.66
module load python/2.7.12_intel-2017_update-1
conda create -m -n <nombre proyecto> intelpython2_core python=2
source activate <nombre proyecto>
python setup.py install
```

Use mode

```
module load python/2.7.12_intel-2017_update-1
module load vsearch/2.4.0_gcc-4.9.4
module load muscle/3.8.31_gcc-4.9.4
source activate <nombre proyecto>
python
>>> import pyrad
```

References

- http://dereneaton.com/software/pyrad/

Author

- Alejandro Salgado Gómez

3.3.60 QE

Quantum ESPRESSO is an integrated suite of Open-Source computer codes for electronic-structure calculations and materials modeling at the nanoscale. It is based on density-functional theory, plane waves, and pseudopotentials.

Quantum ESPRESSO has evolved into a distribution of independent and inter-operable codes in the spirit of an open-source project. The Quantum ESPRESSO distribution consists of a "historical" core set of components, and a set of plug-ins that perform more advanced tasks, plus a number of third-party packages designed to be inter-operable with the core components. Researchers active in the field of electronic-structure calculations are encouraged to participate in the project by contributing their own codes or by implementing their own ideas into existing codes.

Quantum ESPRESSO is an open initiative, in collaboration with many groups world-wide, coordinated by the Quantum ESPRESSO Foundation. Present members of the latter include Scuola Internazionale Superiore di Studi Avanzati, the Abdus Salam International Centre for Theoretical Physics (Trieste), the CINECA National Supercomputing Center (Bologna), the Ecole Polytechnique Fédérale de Lausanne, the University of North Texas (Dallas), the Duke University (Durham). Courses on modern electronic-structure theory with hands-on tutorials on the Quantum ESPRESSO codes are offered on a regular basis in collaboration with the Abdus Salam International Centre for Theoretical Physics in Trieste.

6.1

Table of contents

- **Installation date:** 16/03/2017
- **URL:** http://www.quantum-espresso.org
- **Apolo version:** Apolo II
- **License:** GNU GENERAL PUBLIC LICENSE Version 2

Pre requirements

- Intel Parell Studio XE Cluster Edition 2017 - Update 1
- Intel Compilers Fortran and C
- Intel MPI (Fortran and C)
- Intel MKL

Installation

1. Download the latest version of the software (Repository) (https://github.com/QEF/q-e):

```
cd /home/mgomezzul/apps/qe/src/intel
wget https://github.com/QEF/q-e/archive/qe-6.1.0.tar.gz
tar xf qe-6.1.tar.gz
cd qe-6.1
```

2. To proceed with the configuration and compilation we must continue with the following steps:

```
module load intel/2017_update-1 impi/2017_update-1 mkl/2017_update-1
unset LD
export CPP="$CC -E"
export LD=$MPIF90

./configure --prefix=/share/apps/qe/6.1/intel_17.0.1_impi_17.0.1_mkl_17.0.1 --
↪build=x86_64-redhat-linux --enable-openmp --enable-parallel --with-scalapack=yes␣
↪FC=ifort F90=ifort F77=ifort CC=icc CXX=icpc CPP="icc -E" LD=mpiifort BLAS_LIBS="-
↪lmkl_intel_lp64 -lmkl_intel_thread -lmkl_core -liomp5 -lpthread -lm -ldl" LAPACK_
↪LIBS="" SCALAPACK_LIBS="-lmkl_scalapack_lp64 -lmkl_blacs_intelmpi_lp64" FFT_LIBS="-
↪lmkl_intel_lp64 -lmkl_intel_thread -lmkl_core -liomp5 -lpthread -lm -ldl" 2>&1 |␣
↪tee conf.log
sudo mkdir -p /share/apps/qe/6.1/intel_17.0.1_impi_17.0.1_mkl_17.0.1
sudo chown -R mgomezzul.apolo /share/apps/qe/6.1/intel_17.0.1_impi_17.0.1_mkl_17.0.1
```

(continues on next page)

(continued from previous page)

```
emacs make.inc
...
DFLAGS         =  -D__INTEL -D__SCALAPACK -D__OPENMP -D__DFTI -D__MPI
...
make all -j 8 2>&1 | tee qe-make.log
make install 2>&1 | tee qe-make-install.log
sudo chown -R root.root /share/apps/qe/6.1/intel_17.0.1_impi_17.0.1_mkl_17.0.1
```

3. Add the potential pseudos

```
sudo mkdir -p /share/apps/qe/6.1/intel_17.0.1_impi_17.0.1_mkl_17.0.1/pseudos
cd /share/apps/qe/6.1/intel_17.0.1_impi_17.0.1_mkl_17.0.1/pseudos
# Verificar la última versión de los pseudos - http://www.quantum-espresso.org/
↪pseudopotentials/
wget http://www.quantum-espresso.org/wp-content/uploads/upf_files/upf_files.tar
tar xf upf_files.tar
rm upf_files.tar
```

Module

```
##%Module1.0#####################################################################
##
## modules qe/6.1_intel-17.0.1 Written by Mateo Gómez Zuluaga
##

proc ModulesHelp { } {
        puts stderr "\tSets the environment for Quantum Espresso 6.1 in the \
                    \n\tfollowing shared directory \
            \n\t/share/apps/qe/6.1/intel_17.0.1_impi_17.0.1_mkl_17.0.1\n"
}

module-whatis "\n\n\tQuatum Espresso-6.1 is  an integrated suite of \
        \n\tOpen-Source computer codes for electronic-structure \
        \n\tcalculations and materials modeling at the nanoscale \
            \n\tbuilded with Intel Parallel Studio XE Cluster Edition \
        \n\t2017 Update 1 (Intel MPI and Intel MKL)\n"


# for Tcl script use only
set           topdir        /share/apps/qe/6.1/intel_17.0.1_impi_17.0.1_mkl_17.0.1
set   version       6.1
set       sys               x86_64-redhat-linux
set       user              [exec bash -c "echo \$USER"]

conflict qe

module load intel/2017_update-1
module load impi/2017_update-1
module load mkl/2017_update-1

prepend-path PATH           $topdir/bin
setenv           BIN_DIR            $topdir/bin

setenv       OMP_NUM_THREADS    1
```

(continues on next page)

(continued from previous page)

```
setenv            ESPRESSO_PSEUDO    $topdir/pseudo
setenv            PSEUDO_DIR         $topdir/pseudo
setenv            ESPRESSO_TMPDIR    /scratch-local/$user/qe
setenv        TMP_DIR                /scratch-local/$user/qe
setenv            NETWORK_PSEUDO     http://www.quantum-espresso.org/wp-content/
↪uploads/upf_files/
```

Mode of use

Load the necessary environment through the **module**:

..code-block:: bash

    module load qe/6.1_intel-17.0.1

Slurm template

```
#!/bin/sh
#SBATCH --partition=longjobs
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=32
#SBATCH --time=1-00
#SBATCH --job-name=qe_test
#SBATCH -o result_%N_%j.out
#SBATCH -e result_%N_%j.err
#SBATCH --mail-type=ALL
#SBATCH --mail-user=jrendon8@eafit.edu.co


# Don't share environment variables
export SBATCH_EXPORT=NONE
export OMP_NUM_THREADS=1

module load qe/6.1_intel-17.0.1

srun pw.x < test_1.in
```

Input file

```
&CONTROL
calculation = "scf", ! single point calculation (default, could be omitted)
prefix = "CO", ! all auxiliary files will have filename beginning by prefix
tprnfor = .true.
/
&SYSTEM
ibrav = 0, ! Bravais lattice defined by user in CELL_PARAMETERS card
celldm(1)= 1.88972687, ! define length unit as 1 AA= 1/0.529177 bohr
ntyp = 2, ! number of atomic species (see later ATOMIC_SPECIES)
nat = 2, ! number of atoms in the unit cell (see later ATOMIC_POSITIONS)
ecutwfc = 24.D0,
ecutrho = 144.D0,
```

(continues on next page)

(continued from previous page)

```
/
&ELECTRONS
conv_thr = 1.D-7, ! convergence threshold on total energy , in Rydberg
/
CELL_PARAMETERS cubic
10.0  0.0  0.0
0.0 10.0  0.0
0.0  0.0 10.0
ATOMIC_SPECIES
O 1.00 O.pbe-rrkjus.UPF
C 1.00 C.pbe-rrkjus.UPF
ATOMIC_POSITIONS angstrom
C 1.152 0.0 0.0
O 0.000 0.0 0.0
K_POINTS gamma
```

References

- http://www.archer.ac.uk/documentation/software/espresso/compiling_5.0.3_mkl-phase1.php
- https://glennklockwood.blogspot.com.co/2014/02/quantum-espresso-compiling-and-choice.html
- https://proteusmaster.urcf.drexel.edu/urcfwiki/index.php/Compiling_Quantum_Espresso
- https://www.hpc.ntnu.no/ntnu-hpc-group/vilje/user-guide/software/quantum-espresso
- https://nishaagrawal.wordpress.com/2013/03/21/quantum-espresso-5-0-2qe-64-bit-installation-with-intel-compser-xe-2013-and-
- https://software.intel.com/en-us/articles/quantum-espresso-for-intel-xeon-phi-coprocessor
- http://www.quantum-espresso.org/pseudopotentials/

Author

- Mateo Gómez Zuluaga

6.2.1

Table of contents

– *Author*

- **Installation date:** 30/05/2018
- **URL:** http://www.quantum-espresso.org
- **Apolo version:** Apolo II and Cronos
- **License:** GNU GENERAL PUBLIC LICENSE Version 2

Pre requirements

- Intel Parell Studio XE Cluster Edition 2018 - Update 2
- Intel Compilers (Fortran and C)
- Intel MPI (Fortran and C)
- Intel MKL

Installation

1. Download the latest version of the software (Git Repository):

```
cd /home/$USER/apps/qe/src/intel
wget https://github.com/QEF/q-e/archive/qe-6.2.1.tar.gz
tar xf qe-6.2.1.tar.gz
cd qe-6.2.1
```

2. To proceed with the configuration and compilation we must follow these steps:

```
module load mkl/18.0.2 impi/18.0.2
module load intel/18.0.2 # Sobreescribe las variables CC, CXX, FC, etc..
unset LD
export LD=$MPIF90
export CPP="$CC -E"
```

3. After establishing the compilation environment we must do the following steps for configuration:

```
sudo mkdir -p /share/apps/qe/6.2.1/intel-18.0.2
sudo chown -R $USER.apolo /share/apps/qe/6.2.1/intel-18.0.2
../configure --prefix=/share/apps/qe/6.2.1/intel-18.0.2 --build=x86_64-redhat-linux --
↪enable-openmp --enable-parallel --with-scalapack=intel 2>&1 | tee conf.log
```

4. Now we must edit the **make.inc** file and it should look like the one found here (make.inc)

```
emacs make.inc
...
DFLAGS         =  -D__DFTI -D__MPI -D__SCALAPACK -D__INTEL -D__OPENM
...
CPP            = "icc -E"
...
CFLAGS         = -O3 -xHost -fno-alias -ansi-alias $(DFLAGS) $(IFLAGS)
FFLAGS         = -O3 -xHost -fno-alias -ansi-alias -assume byterecl -g -traceback -
↪qopenmp
...
```

(continues on next page)

(continued from previous page)

```
LD              = mpiifort
...
# External Libraries (if any) : blas, lapack, fft, MPI
BLAS_LIBS       =   -lmkl_intel_lp64  -lmkl_intel_thread -lmkl_core
BLAS_LIBS_SWITCH = external

# If you have nothing better, use the local copy via "--with-netlib" :
# LAPACK_LIBS = /your/path/to/espresso/LAPACK/lapack.a
# LAPACK_LIBS_SWITCH = internal
# For IBM machines with essl (-D__ESSL): load essl BEFORE lapack!
# remember that LAPACK_LIBS precedes BLAS_LIBS in loading order

LAPACK_LIBS     =
LAPACK_LIBS_SWITCH = external

SCALAPACK_LIBS = -lmkl_scalapack_lp64 -lmkl_blacs_intelmpi_lp64
...
```

- **Note:** review in detail the make.inc present in GitLab.

5. After the modifications in the previous file we can continue with the compilation

```
make all -j 8 2>&1 | tee qe-make.log
make install 2>&1 | tee qe-make-install.log
sudo chown -R root.root /share/apps/qe/6.2.1/intel-18.0.2
```

- **Note:** sometimes you have to run the **make -j 8** twice

6. Add the potential pseudo

- **Note:** Check if they are already present in any previous Quatum-Espresso installation

1. If they are already present in a previous installation.

```
sudo ln -s /share/apps/qe/6.2.1/gcc-5.5.0/pseudo /share/apps/qe/6.2.1/intel-18.0.2/
↪pseudo
```

2. If they are not present.

```
sudo mkdir -p /share/apps/qe/6.2.1/intel-18.0.2/pseudo
cd /share/apps/qe/6.2.1/intel-18.0.2/pseudo
# Check the latest version of the pseudos - http://www.quantum-espresso.org/
↪pseudopotentials/
wget http://www.quantum-espresso.org/wp-content/uploads/upf_files/upf_files.tar
tar xf upf_files.tar
rm upf_files.tar
```

Module

```
#%Module1.0#####################################################################
##
## module load qe/6.2.1_intel-18.0.2
##
## /share/apps/modules/qe/6.2.1_intel-18.0.2
## Written by Mateo Gómez-Zuluaga
##
```

(continues on next page)

(continued from previous page)

```
proc ModulesHelp {} {
    global version modroot
    puts stderr "Sets the environment for using qe 6.2.1\
        \nin the shared directory /share/apps/qe/6.2.1/intel-18.0.2\
        \nbuilded with Intel Parallel Studio XE Cluster Edittion 2018 Update 2."
}

module-whatis "(Name________) qe"
module-whatis "(Version_____) 6.2.1"
module-whatis "(Compilers___) intel-18.0.2"
module-whatis "(System______) x86_64-redhat-linux"
module-whatis "(Libraries___) mkl-18.0.2"

# for Tcl script use only
set         topdir        /share/apps/qe/6.2.1/intel-18.0.2
set         version       6.2.1
set         sys           x86_64-redhat-linux
set         user              [exec bash -c "echo \$USER"]

conflict qe
module load mkl/18.0.2
module load impi/18.0.2

setenv              OMP_NUM_THREADS         1
setenv              ESPRESSO_PSEUDO         $topdir/pseudo
setenv              PSEUDO_DIR              $topdir/pseudo
setenv              ESPRESSO_TMPDIR         /scratch-local/$user/qe
setenv              TMP_DIR                 /scratch-local/$user/qe
setenv              NETWORK_PSEUDO          http://www.quantum-espresso.org/wp-
↪content/uploads/upf_files
setenv              BIN_DIR                 $topdir/bin

prepend-path        PATH                    $topdir/bin
```

Mode of use

Load the necessary environment through the **module**:

```
module load qe/6.2.1_intel-18.0.2
```

Slurm template

```
#!/bin/sh
#SBATCH --partition=longjobs
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=16
#SBATCH --time=1-00
#SBATCH --job-name=qe_test
#SBATCH -o result_%N_%j.out
#SBATCH -e result_%N_%j.err
```

(continues on next page)

(continued from previous page)

```
# Don't share environment variables
export SBATCH_EXPORT=NONE
export OMP_NUM_THREADS=1

module load qe/6.2.1_intel-18.0.2

srun --mpi=pmi2 pw.x < test_1.in
```

Input file

```
&CONTROL
calculation = "scf", ! single point calculation (default, could be omitted)
prefix = "CO", ! all auxiliary files will have filename beginning by prefix
tprnfor = .true.
/
&SYSTEM
ibrav = 0, ! Bravais lattice defined by user in CELL_PARAMETERS card
celldm(1)= 1.88972687, ! define length unit as 1 AA= 1/0.529177 bohr
ntyp = 2, ! number of atomic species (see later ATOMIC_SPECIES)
nat = 2, ! number of atoms in the unit cell (see later ATOMIC_POSITIONS)
ecutwfc = 24.D0,
ecutrho = 144.D0,
/
&ELECTRONS
conv_thr = 1.D-7, ! convergence threshold on total energy , in Rydberg
/
CELL_PARAMETERS cubic
10.0  0.0  0.0
0.0 10.0  0.0
0.0  0.0 10.0
ATOMIC_SPECIES
O 1.00 O.pbe-rrkjus.UPF
C 1.00 C.pbe-rrkjus.UPF
ATOMIC_POSITIONS angstrom
C 1.152 0.0 0.0
O 0.000 0.0 0.0
K_POINTS gamma
```

References

- http://www.archer.ac.uk/documentation/software/espresso/compiling_5.0.3_mkl-phase1.php
- https://glennklockwood.blogspot.com.co/2014/02/quantum-espresso-compiling-and-choice.html
- https://proteusmaster.urcf.drexel.edu/urcfwiki/index.php/Compiling_Quantum_Espresso
- https://www.hpc.ntnu.no/ntnu-hpc-group/vilje/user-guide/software/quantum-espresso
- https://nishaagrawal.wordpress.com/2013/03/21/quantum-espresso-5-0-2qe-64-bit-installation-with-intel-compser-xe-2013-and-
- https://software.intel.com/en-us/articles/quantum-espresso-for-intel-xeon-phi-coprocessor
- http://www.quantum-espresso.org/pseudopotentials/

Author

- Mateo Gómez Zuluaga

3.3.61 Qiime2

QIIME 2[1] is a powerful, extensible, and decentralized microbiome analysis package with a focus on data and analysis transparency. QIIME 2 enables researchers to start an analysis with raw DNA sequence data and finish with publication-quality figures and statistical results.

Key features:

- Integrated and automatic tracking of data provenance
- Semantic type system
- Plugin system for extending microbiome analysis functionality
- Support for multiple types of user interfaces (e.g. API, command line, graphical)

QIIME 2 is a complete redesign and rewrite of the QIIME 1 microbiome analysis pipeline. QIIME 2 will address many of the limitations of QIIME 1, while retaining the features that makes QIIME 1 a powerful and widely-used analysis pipeline.

QIIME 2 currently supports an initial end-to-end microbiome analysis pipeline. New functionality will regularly become available through QIIME 2 plugins. You can view a list of plugins that are currently available on the QIIME 2 plugin availability page. The future plugins page lists plugins that are being developed.

Qiime2 2018.4

Table of Contents

Basic information

- **Installation Date:** 15/06/2018
- **URL:** https://docs.qiime2.org/2018.4

[1] QIIME 2 development team. (n.d.). What is QIIME 2? Retrieved from https://docs.qiime2.org/2019.7/about/

- **License:** BSD 3-Clause License
- **Installed on:** Apolo II & Cronos

Tested on (Requirements)

- **Dependencies to run Qiime2:**
  - Python >= 3.6.5 (Miniconda)

Installation

1. Charge the environment and perform the following steps for installation:

```
module load python/3.6.5_miniconda-4.5.1
cd /home/mgomezzul/apps/qiime2
wget https://data.qiime2.org/distro/core/qiime2-2018.4-py35-linux-conda.
↪yml
conda env create -n qiime2-2018.4 --file qiime2-2018.4-py35-linux-conda.
↪yml
```

Mode of Use

Load the necessary environment through the modules:

```
module load python/3.6.5_miniconda-4.5.1
source activate qiime2-2018.4
```

SLURM Template

```
#!/bin/sh

#SBATCH --partition=longjobs
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=16
#SBATCH --time=10:00
#SBATCH --job-name=qiime2_test
#SBATCH -o result_%N_%j.out
#SBATCH -e result_%N_%j.err

# Don't share environment variables
export SBATCH_EXPORT=NONE
export OMP_NUM_THREADS=$SLURM_NTASKS

module load python/3.6.5_miniconda-4.5.1

source activate qiime2-2018.4

qiime tools import --type EMPPairedEndSequences \
    --input-path /home/mgomezzul/test/qiime2/source \
    --output-path /home/mgomezzul/test/qiime2/result.qza
```

Input files

- Remember to create the **source** directory in the place where the **slurm.sh** file is located
- Have the files (**barcodes.fastq.gz, forward.fastq.gz, reverse.fastq.gz**)

References

- https://docs.qiime2.org/2018.4/install/native/#install-qiime-2-within-a-conda-environment
- https://anaconda.org/qiime2/qiime2
- https://wiki.hpcc.msu.edu/display/ITH/QIIME+2
- https://hpc.research.uts.edu.au/software_specific/software_qiime2/

Authors

- Mateo Gómez Zuluaga

Qiime2 2019.7

Basic Information

- **Deploy date:** 30 July 2019
- **Official Website:** https://docs.qiime2.org
- **License:** BSD 3-Clause License
- **Installed on:** *Apolo II*, *Cronos*

Installation

Tested on (Requirements)

- **OS base:** CentOS (x86_64) $\geq$ 6.6
- Python $\geq$ 2.7

Build process

This entry described the installation process of Qiime2 in a Conda environment.

1. Download the dependencies file for your Qiime version, in this example we download the 2019.7 version.

```
wget https://data.qiime2.org/distro/core/qiime2-2019.7-py36-linux-conda.yml
```

2. Create the environment.

> **Note:** It's highly recommend creating a new environment for each version of QIIME 2 release being installed.

```
conda env create -n qiime2-2019.7 --file qiime2-2019.7-py36-linux-conda.yml
```

Testing

1. Load the python module with conda.

```
module load python
```

2. Activate the conda environment `source activate <environment-name>`, this example we use the qiimw2-2019.7 environment.

```
source activate qiime2-2019.7
```

3. Run a simple command.

```
qiime --help
```

Usage

This section describes the method to submit jobs with the resource manager SLURM.

Listing 55: `slurm.sh`

```
#!/bin/bash

#SBATCH --job-name=<job_name>       # Job name
#SBATCH --mail-type=ALL         # Mail notification
#SBATCH --mail-user=test@eafit.edu.co  # User Email
#SBATCH --output=%x.%j.out # Stdout (%j expands to jobId, %x expands to jobName)
#SBATCH --error=%x.%j.err  # Stderr (%j expands to jobId, %x expands to jobNam0e)
#SBATCH --ntasks=1                   # Number of tasks (processes)
#SBATCH --cpus-per-task=32           # Number of threads
#SBATCH --time=1-00:00               # Walltime
#SBATCH --partition=longjobs         # Partition


##### ENVIRONMENT CREATION #####
module load python
source activate qiime2-2019.7

##### JOB COMMANDS ####
#3. Sequence quality control and feature table construction
qiime <plugin> <flags>
```

Note: For more information about plugins, read the qiime2 documentation

Authors

- Manuela Carrasco Pinzón <mcarras1@eafit.edu.co>

3.3.62 RAxML

RAxML[1] (Randomized Axelerated Maximum Likelihood) is a popular program for phylogenetic analysis of large datasets under maximum likelihood. Its major strength is a fast maximum likelihood tree search algorithm that returns trees with good likelihood scores. Since the last RAxML paper (Stamatakis, 2006), it has been continuously maintained and extended to accommodate the increasingly growing input datasets and to serve the needs of the user community. In the following, I will present some of the most notable new features and extensions

RAxML - 8.2.12

Basic information

- **Deploy date:** 2 August 2018
- **Official Website:** https://sco.h-its.org/exelixis/web/software/raxml/
- **License:** GNU GENERAL PUBLIC LICENSE - Version 3 (GPL 3.0), 29 June 2007
- **Installed on:** *Apolo II*, *Cronos*
- **Available versions:** Hybrid (MPI and Threads), MPI

Installation

This entry covers the entire process performed for the installation and configuration of RAxML on a cluster with the conditions described below.

Contents

Tested on (Requirements)

- **OS base:** CentOS (x86_64) $\geq$ 6.6 (Rocks 6.2)
- **Compiler:** Intel Parallel Studio XE Cluster Edition $\geq$ 17.0.1
- **MPI:** Intel MPI $\geq$ 17.0.1
- **Scheduler:** SLURM $\geq$ 16.05.6

Build process

1. Get source code from the github repository (RAxML Releases).

[1] Alexandros Stamatakis; RAxML version 8: a tool for phylogenetic analysis and post-analysis of large phylogenies, Bioinformatics, Volume 30, Issue 9, 1 May 2014, Pages 1312–1313, https://doi.org/10.1093/bioinformatics/btu033

```
cd ~/apps/raxml/src/intel
wget https://github.com/stamatak/standard-RAxML/archive/v8.2.12.tar.gz
tar -zxvf v8.2.12.tar.gz
```

2. To build RAxML follow next steps:

Apolo II

1. Create a new Makefile for your particular environment

   (i.e. *AVX2* instruction set and the supported MPI implementation)

```
# Go to source directory
cd standard-RAxML-8.2.12
# Load the necessary environment
module load impi/2017_update-1
# Create a new Makefile
cp Makefile.AVX2.HYBRID.gcc Makefile.AVX2.HYBRID.icc
```

2. Add Intel MPI support editing the *Makefile* (**Makefile.AVX2.HYBRID.icc**)

```
# From
CC = mpicc
# To
CC = mpiicc
```

3. Build RAxML and deploy it

```
make -f Makefile.AVX2.HYBRID.icc 2>&1 | tee raxml-make.log
sudo mkdir -p /share/apps/raxml/8.2.12/intel-17.0.1/bin
sudo cp raxmlHPC-HYBRID-AVX2 /share/apps/raxml/8.2.12/intel-17.0.1/bin
```

Note: If something goes wrong, check the **raxml-make.log** file to review the build process.

Cronos

1. Create a new Makefile for your particular environment

   (i.e. *AVX* instruction set and the supported MPI implementation)

```
# Go to source directory
cd standard-RAxML-8.2.12
# Load the necessary environment
module load impi/18.0.2
# Create a new Makefile
cp Makefile.AVX.HYBRID.gcc Makefile.AVX.HYBRID.icc
```

2. Add Intel MPI support editing the *Makefile* (**Makefile.AVX.HYBRID.icc**)

```
# From
CC = mpicc
# To
CC = mpiicc
```

3. Build RAxML and deploy it

```
make -f Makefile.AVX.HYBRID.icc 2>&1 | tee raxml-make.log
sudo mkdir -p /share/apps/raxml/8.2.12/intel-18.0.2/bin
sudo cp raxmlHPC-HYBRID-AVX /share/apps/raxml/8.2.12/intel-18.0.2/bin
```

Note: If something goes wrong, check the **raxml-make.log** file to review the build process.

Modulefile

Apolo II

Listing 56: `Module file`

```
##%Module1.0#####################################################################
##
## module load raxml/8.2.12_intel-17.0.1
##
## /share/apps/modules/raxml/8.2.12_intel-17.0.1
## Written by Mateo Gómez Zuluaga
##

proc ModulesHelp {} {
     global version modroot
     puts stderr "Sets the environment for using raxml 8.2.12\
                  \nin the shared directory \
                  \n/share/apps/raxml/8.2.12/intel-17.0.1\
                  \nbuilded with Intel Parallel Studio Cluster Edition XE 17.0.1\
                  \n"
}

module-whatis "(Name________) raxml"
module-whatis "(Version_____) 8.2.12"
module-whatis "(Compilers___) intel-17.0.1"
module-whatis "(System______) x86_64-redhat-linux"
module-whatis "(Libraries___) "

# for Tcl script use only
set         topdir        /share/apps/raxml/8.2.12/intel-17.0.1
set         version       8.2.12
set         sys           x86_64-redhat-linux

conflict raxml
module load impi/2017_update-1

prepend-path        PATH                        $topdir/bin
```

Cronos

Listing 57: `Module file`

```
##%Module1.0#####################################################################
##
## module load raxml/8.2.12_intel-18.02
```

(continues on next page)

(continued from previous page)

```
##
## /share/apps/modules/raxml/8.2.12_intel-18.02
## Written by Mateo Gómez Zuluaga
##

proc ModulesHelp {} {
     global version modroot
     puts stderr "Sets the environment for using raxml 8.2.12\
                  \nin the shared directory \
                  \n/share/apps/raxml/8.2.12/intel-18.02\
                  \nbuilded with Intel Parallel Studio Cluster Edition XE 18.02\
                  \n"
}

module-whatis "(Name________) raxml"
module-whatis "(Version_____) 8.2.12"
module-whatis "(Compilers___) intel-18.02"
module-whatis "(System______) x86_64-redhat-linux"
module-whatis "(Libraries___) "

# for Tcl script use only
set         topdir        /share/apps/raxml/8.2.12/intel-18.02
set         version       8.2.12
set         sys           x86_64-redhat-linux

conflict raxml
module load impi/18.0.2

prepend-path        PATH                        $topdir/bin
```

Usage

This subsection describes how to use RAxML on a cluster and the necessary elements to get a good performance.

1. Before launch **RAxML** you should read next documentation
   - `The RAxML v8.2.X Manual (Mandatory) *`
     - (*When to use which Version?*)
   - `Hybrid Parallelization of the MrBayes & RAxML Phylogenetics Codes`
     - (*Hybrid MPI/Pthreads*)

   Note: It is really important to understand how the *HYBRID* version works, since this is the only available version for HPC scenarios. Additionally, understanding the behavior of the *HYBRID* version is the key to properly use the computational resources and achieve better performance.

2. In the following example we will run 100 bootstrap replicates (MPI parallelization) and independent tree searches (PThreads - shared memory) for each bootstrap replicate, all of this using SLURM (Resource Manager) to spawn properly the processes across the nodes.

Listing 58: `slurm.sh`

```
#!/bin/bash

#SBATCH --partition=longjobs
#SBATCH --nodes=3
#SBATCH --ntasks-per-node=2
#SBATCH --cpus-per-task=16
#SBATCH --time=48:00:00
#SBATCH --job-name=RAxML_test
#SBATCH -o result_%N_%j.out
#SBATCH -e result_%N_%j.err

# Default variables
export SBATCH_EXPORT=NONE
export OMP_NUM_THREADS=1

# Load RAxML module file
module load raxml/8.2.12_intel-17.0.1

# Launch RAxML (MPI with srun (pmi2) and PThreads using '-T' argument
# and SLURM_CPUS_PER_TASK environment variable.
srun --mpi=pmi2 raxmlHPC-HYBRID-AVX2 -s funiji.fasta -T $SLURM_CPUS_PER_TASK \
     -X -f a -n out_1 -m GTRGAMMA -x 6669 -p 2539 -N 100
```

Note: Node quick specs (Apolo II): 32 Cores, 64 GB RAM

- - *-ntasks-per-node* MPI process per node
- - *-cpus-per-task* PThreads per MPI process
- - *-nodes* Number of nodes

In this case, we will use 2 MPI process per node and each MPI process has 16 PThreads; for a total of 32 processes per node. Also we will use 3 nodes.

References

- The RAxML v8.2.X Manual, Alexandros Stamatakis Heidelberg Institute for Theoretical Studies, July 20, 2016

Authors

- Mateo Gómez-Zuluaga <mgomezz@eafit.edu.co>

Table of Contents

RAxML 8.2.9

- **Installation date:** 07/02/2017
- **URL:** http://sco.h-its.org/exelixis/web/software/raxml/
- **Apolo version:** Apolo II
- **License:** GNU GENERAL PUBLIC LICENSE Version 3

Dependencies

-Intel Parallel Studio XE 2017 Update 1 (Intel C Compiler and Intel MPI)

Installation

1. First download the tar from the main page

```
wget https://github.com/stamatak/standard-RAxML/archive/v8.2.9.tar.gz
tar -zxvf v8.2.9.tar.gz
```

1. compilation config and editing the makefile

```
cd standard-RAxML-8.2.9
module load impi/2017_update-1
cp Makefile.AVX2.HYBRID.gcc Makefile.AVX2.HYBRID.icc
emacs Makefile.AVX2.HYBRID.icc (Cambiar CC = mpicc por CC = mpiicc)
make -f Makefile.AVX2.HYBRID.icc 2>&1 | tee raxml-make.log
```

Module

```
##%Module1.0#####################################################################
##
## module raxml/8.2.9_intel_impi-2017_update-1
##
## /share/apps/modules/raxml/8.2.9_intel_impi-2017_update-1     Written by Mateo␣
↪Gomez-Zuluaga
##

proc ModulesHelp { } {
    puts stderr "\traxml/8.2.9_intel-2017_update-1 - sets the Environment for Raxml␣
↪in \
    \n\tthe share directory /share/apps/raxml/8.2.9/intel_impi/2017_update-1\n"
}

module-whatis "\n\n\tSets the environment for using Raxml 8.2.9 \
```

(continues on next page)

(continued from previous page)

```
                \n\tbuilded with Intel Parallel Studio XE 2017\n"

# for Tcl script use only
set       topdir     /share/apps/raxml/8.2.9/intel_impi/2017_update-1
set       version    8.2.9
set       sys        x86_64-redhat-linux

module load impi/2017_update-1

prepend-path PATH    $topdir/bin
```

Usage mode

```
module load raxml/8.2.9_intel_impi-2017_update-1
```

References

- manual
- http://sco.h-its.org/exelixis/web/software/raxml/cluster.html
- http://sco.h-its.org/exelixis/resource/doc/Phylo100225.pdf

Author

- Mateo Gómez Zuluaga

3.3.63 rDock

Description

rDock is a fast and versatile Open Source docking program that can be used to dock small molecules against proteins and nucleic acids. It is designed for High Throughput Virtual Screening (HTVS) campaigns and Binding Mode prediction studies.

For additional information you can open those links:

- http://rdock.sourceforge.net/getting-started/
- https://sourceforge.net/projects/rdock/files/?source=navbar
- http://rdock.sourceforge.net/about/

Table of Contents

rDock 2013.1

- **Installation date:** 18/10/2017
- **URL:** http://rdock.sourceforge.net/
- **Apolo version:** Apolo II
- **License:** GNU-LGPLv3.0

Dependencies

- GNU GCC >= 3.3.4
- cppunit >= 1.12.1
- cppunit-devel >= 1.12.1
- popt-devel >= 1.7-190
- popt >= 1.7-190

Installation

Load the needed modules or add the to the path

1. Download the binaries

```
$ wget https://downloads.sourceforge.net/project/rdock/rDock_2013.1_src.tar.gz?r=https
↪%3A%2F%2Fsourceforge.net%2Fprojects%2Frdock%2F%3Fsource%3Dtyp_redirect&
↪ts=1508430181&use_mirror=superb-dca2
$ mv rDock_2013.1_src.tar.gz\?r\=https\:%2F%2Fsourceforge.net%2Fprojects%2Frdock%2F\?
↪source\=typ_redirect rDock_2013.1_src.tar.gz
```

2. unzip and compilation

```
$ tar -xvzf rDock_2013.1_src.tar.gz
$ cd rDock_2013.1_src/build
$ make linux-g++-64 (usar 'make linux-g++' para 32-bits)
```

3. Test installation

```
$ make test
```

Module

```
##%Module1.0#####################################################################
##
## module load rdock/2013.1
##
## /share/apps/modules/rdock/2013.1
## Written by Juan David Arcila Moreno
##

proc ModulesHelp {} {
     global version modroot
     puts stderr "Sets the environment for using rdock 2013.1\
                  \nin the shared directory \
                  \n/share/apps/rdock/2013.1/\
                  \nbuilded with gcc-5.4.0"
}

module-whatis "(Name________) rdock"
module-whatis "(Version_____) 2013.1"
module-whatis "(Compilers___) g++-4.4.7"
module-whatis "(System______) x86_64-redhat-linux"
module-whatis "(Libraries___) popt-1.7-190"

# for Tcl script use only
set         topdir        /share/apps/rdock/2013.1/
set         version       2013.1
set         sys           x86_64-redhat-linux

conflict rdock


prepend-path    PATH                    $topdir/bin

prepend-path    LD_LIBRARY_PATH         $topdir/lib
prepend-path    LIBRARY_PATH            $topdir/lib
prepend-path    LD_RUN_PATH             $topdir/lib
```

Use mode

```
$ module load rdock/2013.1
```

References

- https://sourceforge.net/projects/rdock/files/?source=navbar
- http://rdock.sourceforge.net/

Author

- Juan David Arcila-Moreno

3.3.64 REPET

REPET orchestrates bioinformatics applications in order to perform genomic tasks. "Its two main pipelines are dedicated to detecte, annotate and analyse repeats in genomic sequences, specifically designed for transposable elements (TEs)"[1]

Versions

REPET 2.5

Table of Contents

Basic information

- **Official Website:** https://urgi.versailles.inra.fr/Tools/REPET
- **License:** CeCILL License 2.0
- **Tested on** CentOS (x86_64) 6.6 (Rocks 6.2)

Installation

Note: Official installation instructions can be found in https://urgi.versailles.inra.fr/Tools/REPET/INSTALL.

A detailed list of dependencies can be found at https://urgi.versailles.inra.fr/Tools/REPET/INSTALL#dependencies. Notice that every executable that is needed by REPET should be reachable from your PATH. Be aware that the cause of most of the errors when running REPET is due to the malfunction of a dependency.

The tar file containing REPET can be found here. The compressed package already contains the binaries so there is no need to compile anything. Once you have uncompressed REPET and installed all the dependencies you should configure your environment so that every program needed is in your PATH. Also, you should configure the PYTHONPATH environment variable, so that every REPET python module is reachable.

```
$ export PYTHONPATH=$REPET_HOME
```

[1] REPET. Retrieved April 24th, 2019, from https://urgi.versailles.inra.fr/Tools/REPET

For more information on setting up your REPET environment, you should follow the guide provided at the beginning of the following web page: https://urgi.versailles.inra.fr/Tools/REPET/TEdenovo-tuto.

REPET needs a MySQL server, even though we chose MariaDB. Instructions to configure a MySQL server can be found on this link. You also need to create a database and a user for REPET:

```
# tables engine must be MyISAM
mysql> SET default_storage_engine=MYISAM;

# in order to avoid errors regarding character encoding
mysql> CREATE DATABASE database_name CHARACTER  set utf8 COLLATE  utf8_bin;

# create user and grant privileges on database
mysql> CREATE USER 'username'@'hostname' IDENTIFIED BY 'password';
mysql> GRANT ALL ON database_name.* TO 'username'@'hostname';
mysql> FLUSH PRIVILEGES;
```

Repet needs a working environment with some python modules installed:

```
# load REPET module
$ module load REPET

# create a conda environment named env_name and install mysql-python, pyyaml as␣
↪needed for REPET
$ conda create -n env_name mysql-python pyyaml
```

Note: REPET module will load python/2.7.15_miniconda-4.5.4 automatically and will copy a RepeatMasker directory into your home which you will have to configure as follow.

If you want to use REPET with RepeatMasker:

- The first time you run `module load REPET` you will have to move into RepeatMasker directory and run `perl ./configure`.

```
$ cd $HOME/RepeatMasker
$ perl ./configure
```

It will prompt you to enter the path for some applications. You should enter the following:

```
# perl path
Enter path: env

# RepeatMasker path
Enter path: /home/<your_username>/RepeatMasker

# TRF path
Enter path: /share/apps/REPET/2.5/third_party/others/bin/

# RMblast path
Enter path: /share/apps/REPET/2.5/third_party/rmblast/2.9.0/bin/
```

- Be aware that RepeatMasker comes by default with the open Dfam database. If you want to use RepBase library you should copy the compressed version to RepeatMasker's top directory and uncompress it from there. Then reconfigure RepeatMasker:

```
$ module load REPET
$ cp RepBaseRepeatMaskerEdition-XXXXXXXX.tar.gz $HOME/RepeatMasker/
$ cd $HOME/RepeatMasker
$ gunzip RepBaseRepeatMaskerEdition-XXXXXXXX.tar.gz
$ tar xvf RepBaseRepeatMaskerEdition-XXXXXXXX.tar
$ rm RepBaseRepeatMaskerEdition-XXXXXXXX.tar
$ perl ./configure
```

- When you load REPET module the following script will be executed. It will loads the environment variables for every REPET dependency, and REPET itself. It also load the Python 2.7 module and GCC module.

```
#%Module1.0#####################################################################
##
## module load REPET 2.5
##
## /share/apps/modules/REPET/2.5
## Written by Vincent Arcila
##

proc ModulesHelp {} {
    global version modroot
    puts stderr "Sets the environment for using REPET\
          \nin the shared directory /share/apps/REPET/2.5"
}

module-whatis "(Name________) REPET"
module-whatis "(Version_____) 2.5"
module-whatis "(System______) x86_64-redhat-linux"
module-whatis "(Interpreter_____) Python 2.7"

# for Tcl script use only
set         topdir        /share/apps/REPET/2.5
set         version       2.5
set         sys           x86_64-redhat-linux
set         sourceDir     /share/apps/REPET/2.5/third_party/RepeatMasker/1.332
set         targetDir     $::env(HOME)/RepeatMasker

conflict glibc

module load gcc/5.4.0
module load python/2.7.15_miniconda-4.5.4

if {! [file exist $targetDir]} {
    file mkdir $targetDir
    foreach f [glob -directory $sourceDir -nocomplain *] {
    file copy -force $f $targetDir
    }
}

puts stderr "NOTE: If this is the first time you load the module remember to move into
RepeatMasker's directory \nand execute the configuration script for RepeatMasker:
\n\n\t$ cd ::env(HOME)/RepeatMasker\n\t$ perl ./configure \n\nIf you want to configure
RepBase follow the instructions provided in: http://apolo-docs.readthedocs.io"

prepend-path    PATH                    $targetDir
prepend-path    PATH                    $topdir/REPET/bin
prepend-path    PATH                    $topdir/third_party/RepeatMasker/1.332/
```

(continues on next page)

(continued from previous page)

```
prepend-path    PATH                    $topdir/third_party/others/bin
prepend-path    PATH                    $topdir/third_party/genometools/1.5.9/bin
append-path     PATH                    $topdir/third_party/blast/2.2.9/bin
append-path     PATH                    $topdir/third_party/rmblast/2.9.0/bin

prepend-path    LD_LIBRARY_PATH         $topdir/lib
prepend-path    LIBRARY_PATH            $topdir/lib
prepend-path    LD_RUN_PATH             $topdir/lib

prepend-path    LD_LIBRARY_PATH         $topdir/third_party/genometools/1.5.9/lib
prepend-path    LIBRARY_PATH            $topdir/third_party/genometools/1.5.9/lib
prepend-path    LD_RUN_PATH             $topdir/third_party/genometools/1.5.9/lib

prepend-path    PERL5LIB                $topdir/third_party/RepeatMasker/1.332
prepend-path    PERL5LIB                $topdir/libexec

prepend-path    INCLUDE_PATH            $topdir/include
prepend-path    C_INCLUDE_PATH          $topdir/include
prepend-path    CXX_INCLUDE_PATH        $topdir/include

setenv          REPET_PATH              $topdir/REPET
setenv          PYTHONPATH              $topdir/REPET
```

Usage

Note: If you don't already have a MariaDB account contact the system administrator. Remember to ask for the database name and hostname for the MariaDB server.

In order to use REPET you should load REPET module and activate your Python environment:

```
$ module load REPET

$ source activate env_name
```

REPET's main pipelines are TEdenovo and TEannot. Each of them has it's specific guidelines and dependencies. REPET provides vast documentation for this pipelines: https://urgi.versailles.inra.fr/Tools/REPET/TEdenovo-tuto, https://urgi.versailles.inra.fr/Tools/REPET/TEannot-tuto.

REPET implements a module for using resource managers such as *SLURM* or TORQUE. It will use this module to send jobs to a queue. In order to manage SBATCH parameters, you will have to edit the configuration file for the pipeline you are using (e.g. TEdenovo.cfg). Each job has its own parameters, which can be specified as follows:

```
resources: longjobs --partition=longjobs --time=03:00:00 --out=out.log --error=err.log
```

This entry will make TEdenovo.py use 'longjobs' as the partition. The job will have 3 hours to finish. The job will redirect stdout to out.log and stderr to err.log.

The first word must be the partition where you want your job to be sent. Even though, you should specify the partition again using `--partition=<partition_name>`. It is mandatory to specify the partition as well as the time for the job to finish.

If for some reason some step did not finish as expected and you do not get an error message, you should erase all data on jobs table, so REPET can use *SLURM* to launch jobs again:

```
# connect to your MariaDB server
$ mysql -u <MariaDB_username> -h <MariaDB_server_hostname> -p

# select your database
mysql> USE <your_database>;

# erase all data in the table
mysql> TRUNCATE TABLE jobs;
```

Note: If getting the following error: **ERROR 1130 (HY000): Host 'not.your.hostname.com' is not allowed to connect to this MariaDB server** you should try creating the user using the ip from which you will connect and then add "skip-name-resolve" to MariaDB configuration:

```
[mariadb]
skip-name-resolve
```

Also, be aware that almost all steps create a directory in which will be the output files from those specific steps. If your step failed, there will be the logs along with the files the step produced.

Examples

SLURM scripts for REPET:

- We provide the scripts and config files needed to run REPET on our cluster, you can download the examples:
  - `TEannot-example`
  - `TEdenovo-example`
- You should modify some values accordingly (e.g. you project name or MariaDB username on .cfg files).
- This scripts are based on https://github.com/stajichlab/REPET-slurm. More information on the usage for this scripts can be found there.

Recommended resources

1. A repository containing bash scripts to use REPET with SLURM: https://github.com/stajichlab/REPET-slurm.
2. A REPET practical course: https://biosphere.france-bioinformatique.fr/wikia2/index.php/REPET_practical_course#Start_TEdenovo_pipeline.
3. README from REPET: https://urgi.versailles.inra.fr/Tools/REPET/README.
4. A extensive guide for our resource manager: *SLURM*.

Authors

- Vincent Alejandro Arcila Larrea (vaarcilal@eafit.edu.co).

3.3.65 SAMtools

Description

BWA is a software package for mapping low-divergent sequences against a large reference genome, such as the human genome. It consists of three algorithms: BWA-backtrack, BWA-SW and BWA-MEM. The first algorithm is designed for Illumina sequence reads up to 100bp, while the rest two for longer sequences ranged from 70bp to 1Mbp. BWA-MEM and BWA-SW share similar features such as long-read support and split alignment, but BWA-MEM, which is the latest, is generally recommended for high-quality queries as it is faster and more accurate. BWA-MEM also has better performance than BWA-backtrack for 70-100bp Illumina reads.s.

Table of Contents

1.3.1

- **Installation date:** 28/02/2017
- **URL:** http://samtools.sourceforge.net/
- **Apolo version:** Apolo II
- **License:** The MIT/Expat License

Dependencies

- Intel Parallel Studio XE Cluster Edition 2017 Update 1
- zlib >= 1.2.11

Installation

After solving the aforementioned dependencies, you can proceed with the installation of **SAMtools.**

1. Download the latest version of the software (Source code - bz2) (https://sourceforge.net/projects/samtools/files/):

```
cd /home/$USER/apps/samtools/src
wget https://downloads.sourceforge.net/project/samtools/samtools/1.3.1/samtools-1.3.1.
↪tar.bz2?r=https%3A%2F%2Fsourceforge.net%2Fprojects%2Fsamtools%2Ffiles%2Fsamtools%2F&
↪ts=1488295734&use_mirror=ufpr
tar xf samtools-1.3.1.tar.bz2
```

2. For configuration and compilation, the following steps must be taken:

```
cd samtools-1.3.1
module load zlib/1.2.11_intel-2017_update-1
./configure --prefix=/share/apps/samtools/1.3.1/intel/2017_update-1 2>&1 conf-
↪samtools.log
make 2>&1 | tee make-samtools.log
make check 2>&1 | tee make-check-samtools.log
```

3. After compiling **SAMtools**, we continue with the following steps:

```
sudo mkdir -p /share/apps/samtools/1.3.1/intel/2017_update-1
sudo mkdir -p /share/apps/samtools/1.3.1/intel/2017_update-1/include/bam
sudo mkdir -p /share/apps/samtools/1.3.1/intel/2017_update-1/lib
sudo chown -R $USER.apolo /share/apps/samtools/1.3.1/intel/2017_update-1
make install 2>&1 | tee make-install-samtools.log
cp libbam.a /share/apps/bwa/0.7.15/intel/2017_update-1/lib
cp *.h /share/apps/samtools/1.3.1/intel/2017_update-1/include/bam
sudo chown -R root.root /share/apps/samtools/1.3.1/intel/2017_update-1
```

Module

```
##%Module1.0#####################################################################
##
## module load samtools/1.3.1_intel-2017_update-1
##
## /share/apps/modules/samtools/1.3.1_intel-2017_update-1     Written by Mateo Gomez-
↪Zuluaga
##

proc ModulesHelp { } {
    puts stderr "\tsamtools/1.3.1 - sets the Environment for SAMTOOLS in \
    \n\tthe share directory /share/apps/samtools/1.3.1/intel-2017_update-1\n"
}

module-whatis "\n\n\tSets the environment for using SAMTOOLS-1.3.1 \
               \n\tbuilded with Intel Parallel Studio XE 2017\n"

# for Tcl script use only
set       topdir     /share/apps/samtools/1.3.1/intel/2017_update-1
set       version    1.3.1
set       sys        x86_64-redhat-linux

module load intel/2017_update-1

prepend-path PATH    $topdir/bin

prepend-path LD_LIBRARY_PATH    $topdir/lib
prepend-path LIBRARY_PATH       $topdir/lib
prepend-path LD_RUN_PATH        $topdir/lib

prepend-path C_INCLUDE_PATH     $topdir/include
prepend-path CXX_INCLUDE_PATH   $topdir/include
prepend-path CPLUS_INCLUDE_PATH $topdir/include

prepend-path MANPATH $topdir/share/ma
```

Mode of use

Load the necessary environment through the **module:**

```
module load samtools/1.3.1_intel-2017_update-1
```

TO-DO

References

- INSTALL (File inside compressed package)
- https://github.com/cole-trapnell-lab/cufflinks

Author

- Mateo Gómez Zuluaga

3.3.66 SMOKE

SMOKE is primarily an emissions processing system designed to create gridded, speciated, hourly emissions for input into a variety of air quality models such as CMAQ, REMSAD, CAMX and UAM. SMOKE supports area, biogenic, mobile (both onroad and nonroad), and point source emissions processing for criteria, particulate, and toxic pollutants.[1]

Contents

SMOKE 4.8.1 Installation

Tested on (Requirements)

- **OS base:** Rocky Linux (x86_64) :math: *boldsymbol{ge}* 8.5
- **Compiler:** GCC 9.3.0
- **Requirements:**
- GCC = 9.3.0
- MPICH = 3.4.2
- Zlib = 1.2.11
- Curl = 7.77.0

[1] https://www.cmascenter.org/smoke/

- Netcdf-c disable netcdf-4 = 4.8.0
- Netcdf-fortran = 4.5.3
- Zstd = 1.5.2
- ioapi = 3.2.1
- tcsh

Installation

This entry covers the entire process performed for the installation and configuration of SMOKE 4.8.1 with a GCC compiler

Create a directory for SMOKE and download the source code

```
mkdir SMOKE
cd SMOKE
wget https://github.com/CEMPD/SMOKE/releases/download/SMOKEv481_Jan2021/
↪smoke_install_v481.csh
wget https://github.com/CEMPD/SMOKE/releases/download/SMOKEv481_Jan2021/
↪smoke_v481.Linux2_x86_64ifort.tar.gz
wget https://github.com/CEMPD/SMOKE/releases/download/SMOKEv481_Jan2021/
↪smoke_v481.nctox.data.tar.gz
```

Install SMOKE

From this poin onwards all the commands given have to be executed with tcsh, since SMOKE only works with tcsh.

Set your SMK_HOME directory for the installation and load libraries.

```
setenv SMK_HOME $cwd

module load gcc/9.3.0
module load mpich/3.4.2_gcc-9.3.0
module load zlib/1.2.11_gcc-9.3.0
module load curl/7.77.0_gcc-9.3.0
module load netcdf-fortran/4.5.3_gcc-9.3.0_disable-netcdf-4
```

unzip all files with the provided script.

```
source smoke_install_v481.csh
```

The script will point the possible problems that may arise.

Create the ioapi folder to make the symbolic link to the already compiled ioapi.

```
cd subsys

mkdir -p ioapi/ioapi

cd ioapi/ioapi

ln -s /home/<user>/ioapi/ioapi/* .
```

After linking the ioapi folder, we need to link all the files in the Linux2_x86_64gfort folder.

```
mkdir $SMK_HOME/subsys/ioapi/Linux2_x86_64gfort/

cd $SMK_HOME/subsys/ioapi/Linux2_x86_64gfort/

ln -s /home/<user>/ioapi/Linux2_x86_64gfort/* .
```

After creating the links change the line 25 in of the file in the following path $SMK_HOME/subsys/smoke/assigns/ASSIGNS.nctox.cmaq.cb05_soa.us12-nc

```
vim $SMK_HOME/subsys/smoke/assigns/ASSIGNS.nctox.cmaq.cb05_soa.us12-nc
```

Change

```
setenv BIN    Linux2_x86_64ifort
```

to

```
setenv BIN    Linux2_x86_64gfort
```

so that the compilation uses the gfort compiler.

```
cd $SMK_HOME/subsys/smoke/assigns/

source ASSIGNS.nctox.cmaq.cb05_soa.us12-nc
```

Now we need to go to the build directory, open the Makeinclude file, comment lines 48 and 53, and uncomment lines 49 and 54

```
vim $SMK_HOME/subsys/smoke/src/Makeinclude
```

After that, compile.

```
cd $SMK_HOME/subsys/smoke/src

make
```

If everything is correctly configured and the necessary modules are loaded you will get two types of errors

1. error: enclosing 'parallel'; With this error you should enter in the file where the error has occurred and before the error line, you should comment the line where it says default none with 'c.....'.
2. error:can't convert CHARACTER(1) TO INTEGER(4) at (1); in the error line where it says LFIP = '' change the quotation marks by a zero.

After correcting the errors all executables should be created, and the compilation should be complete.

References

SMOKE v4.8.1 User's Manual https://www.cmascenter.org/smoke/documentation/4.8.1/manual_smokev481.pdf

Author

- Juan Diego Herrera <jdherrera@eafit.edu.co>

3.3.67 Google Sparsehash

C++ associative containers.

Google Sparsehash 2.0.3

Basic information

- **Deploy date:** 22 October 2019
- **Official Website:** https://github.com/sparsehash/sparsehash
- **License:** BSD 3-Clause
- **Installed on:** *Apolo II*

Installation

Note: Note that the compiler used is *Intel 2019 <https://software.intel.com/en-us/articles/intel-c-compiler-190-for-linux-release-notes-for-intel-parallel-studio-xe-2019>*. Also, if the compiler to be used is different than this one, the compiler flag `-xHost` might be different.

---

1. Follow these steps, run:

```
$ git clone https://github.com/sparsehash/sparsehash
$ cd sparsehash && mkdir build && cd build
$ module load intel/19.0.4
$ CFLAGS="-O3 -xHost" CXXFLAGS="-O3 -xHost" ../configure --prefix=/share/
↪apps/sparsehash/2.0.3/intel/19.0.4
$ make && make check
$ sudo mkdir -p /share/apps/sparsehash/2.0.3/intel/19.0.4
$ sudo make install
```

2. Create and place the needed module file. Create a file with the following content:

Listing 59: `2.0.3_intel-19.0.4`

```
##%Module1.0#####################################################################
↪#####
##
## module load sparsehash/2.0.3_intel-19.0.4
##
## /share/apps/sparsehash/2.0.3/intel/19.0.4
## Written by Vincent A. Arcila L and Hamilton Tobon Mosquera.
##

proc ModulesHelp {} {
     global version modroot
     puts stderr "Sets the environment for using Google's sparsehash 2.0.
↪3\
          \nin the shared directory /share/apps/sparsehash/2.0.3/intel/19.
↪0.4\
          \nbuilt with Intel 19.0.4."
}
```

(continues on next page)

(continued from previous page)

```
module-whatis "(Name________) sparsehash"
module-whatis "(Version_____) 2.0.3"
module-whatis "(Compilers___) intel-19.0.4"
module-whatis "(System______) x86_64-redhat-linux"
module-whatis "(Libraries___) "

# for Tcl script use only
set         topdir        /share/apps/sparsehash/2.0.3/intel/19.0.4
set         version       2.0.3
set         sys           x86_64-redhat-linux

conflict sparsehash

prepend-path        LD_LIBRARY_PATH               $topdir/lib
prepend-path        LD_RUN_PATH                 $topdir/lib
prepend-path        LIBRARY_PATH                 $topdir/lib

prepend-path    C_INCLUDE_PATH              $topdir/include
prepend-path    CXX_INCLUDE_PATH            $topdir/include
prepend-path    CPLUS_INCLUDE_PATH          $topdir/include

prepend-path PKG_CONFIG_PATH             $topdir/lib/pkgconfig
```

Create the needed folder and place it:

```
$ sudo mkdir /share/apps/modules/sparsehash/
$ sudo mv 2.0.3_intel-19.0.4 /share/apps/modules/sparsehash/
```

Usage

Load the module:

```
$ module load sparsehash/2.0.3_intel-19.0.4
```

If it's being used to compile an application, loading the module should be enough. If it's being used to code something, include its header files:

```
#include <sparsehash/sparse_hash_map> // or sparse_hash_set, dense_hash_map .
↪..
```

Authors

- Hamilton Tobon-Mosquera <htobonm@eafit.edu.co>

3.3.68 stacks

Description

Stacks is a program to build "loci" from short reading sequences, such as those generated on the Illumina platform. Among its main functions are the construction of genetic maps and the performance of population genomics and phylogeography.

Table of Contents

Stacks 1.44

- **Installation date:** 06/02/2012
- **URL:** http://catchenlab.life.illinois.edu/stacks/
- **Apolo version:** Apolo II
- **License:** GNU GPL

Dependencies

- GNU GCC 4.7

Installation

1. First download the tar from the main page

```
$ wget http://catchenlab.life.illinois.edu/stacks/source/stacks-1.44.tar.gz
$ tar -zxvf stacks-1.44.tar.gz
```

1. configure the makefile

```
cd stacks-1.44
module load gcc/4.9.4
export CXXFLAGS="-O3 -march=broadwell"
./configure --prefix=/share/apps/stacks/1.44/gcc/4.9.4
make
sudo make install
```

Module

```
#%Module1.0#####################################################################
  ##
  ## module stacks/1.44_gcc_4.9.4
  ##
```

(continues on next page)

(continued from previous page)

```
## /share/apps/stacks/1.44/gcc/4.9.4     Written by Alejandro Salgado-Gomez
##

proc ModulesHelp { } {
    puts stderr "\tncl/2.1.18_intel-2017_update-1 - sets the Environment for stacks␣
↪in \
     \n\tthe share directory /share/apps/stacks/1.44/gcc/4.9.4\n"
}

module-whatis "\n\n\tSets the environment for using stacks 1.44 \
               \n\tbuilded with Gcc 4.9.4\n"

# for Tcl script use only
set       topdir     /share/apps/stacks/1.44/gcc/4.9.4
set       version    1.44
set       sys        x86_64-redhat-linux

module load gcc/4.9.4

prepend-path PATH       $topdir/bin
```

Use mode

```
module load stacks/1.44_gcc_4.9.4
```

References

- http://catchenlab.life.illinois.edu/stacks/manual/#install

Author

- Alejandro Salgado Gómez

Table of Contents

Stacks 1.46

- **Installation date:** 21/04/2012
- **URL:** http://catchenlab.life.illinois.edu/stacks/
- **Apolo version:** Apolo II
- **License:** GNU GPL

Dependencies

- GNU GCC 5.4.0
- SparseHash = 2.0.3

Installation

1. First download the tar from the main page

```
$ wget http://catchenlab.life.illinois.edu/stacks/source/stacks-1.46.tar.gz
$ tar -zxvf stacks-1.46.tar.gz
```

1. configure the makefile

```
cd stacks-1.46
module load gcc/5.4.0
sudo mkdir -p /share/apps/stacks/1.46/gcc-5.4.0
sudo chwon -R mgomezzul.apolo /share/apps/stacks/1.46/gcc-5.4.0
./configure --prefix=/share/apps/stacks/1.46/gcc-5.4.0 --build=redhat-linux-x86_64␣
↪CXXFLAGS="-O3 -march=native" CFLAGS="-O3 -march=native" CPPFLAGS="-O3 -march=native
↪" --enable-sparsehash --with-sparsehash-include-path=/share/apps/sparsehash/2.0.3/
↪gcc-5.4.0/include
make
make install
sudo chown -R root.root /share/apps/stacks/1.46/gcc-5.4.0
```

Module

```
##%Module1.0#####################################################################
##
## module load stacks/1.46_gcc-5.4.0
##
## /share/apps/modules/stacks/1.46_gcc-5.4.0
##
## Written by Mateo Gómez-Zuluaga
##

proc ModulesHelp {} {
     global version modroot
     puts stderr "Sets the environment for using stacks 1.46\
          \n\in the shared directory \
          \n\t\t/share/apps/stacks/1.46/gcc-5.4.0\
          \n\t\tbuilded with gcc-5.4.0\
```

(continues on next page)

(continued from previous page)

```
        \n\t\t"
}

module-whatis "(Name_________)  stacks"
module-whatis "(Version______)  1.46"
module-whatis "(Compilers____)  gcc-5.4.0"
module-whatis "(System_______)  x86_64-redhat-linux"
module-whatis "(Libraries____)  sparsehash"

# for Tcl script use only
set         topdir        /share/apps/stacks/1.46/gcc-5.4.0
set         version       1.46
set         sys           x86_64-redhat-linux

conflict stacks

module load sparsehash/2.0.3_gcc-5.4.0

prepend-path PATH                $topdir/bin
```

Use mode

```
module load stacks/1.46_gcc-5.4.0
```

References

- http://catchenlab.life.illinois.edu/stacks/manual/#install

Author

- Mateo Gómez-Zuluaga
- Alejandro Salgado Gómez

3.3.69 Tigmint

Tigmint[1] Correct misassemblies using linked reads.

Tigmint 1.1.2

Basic information

- **Deploy date:** 3 December 2019
- **Official Website:** https://github.com/bcgsc/tigmint
- **License:** GNU GPL v3
- **Installed on:** *Apolo II*

[1] Tigmint. (2019, Oct 28). Retrieved December 2, 2019, from https://github.com/bcgsc/tigmint.git

- **Dependencies:**
  - Miniconda 4.5.1 or greather
  - Bedtools
  - Bwa
  - Samtools
- **Optional dependencies:**
  - *Pigz*. Recommended for parallel compression

Installation

1. Before installing Tigmint install all its dependencies. After installing all the dependencies run:

```
$ mkdir -p /share/apps/tigmint/1.1.2/miniconda/4.5.1
$ git clone https://github.com/bcgsc/tigmint
$ sudo cp -r tigmint/* /share/apps/tigmint/1.1.2/miniconda/4.5.1/
```

2. Create and place the needed module file. Create a file with the following content:

Listing 60: `1.1.2_miniconda-4.5.1`

```
##%Module1.0#####################################################################
↪######
##
## modulefile tigmint/1.1.2_miniconda-4.5.1
##
## Written by Hamilton Tobon Mosquera.
proc ModulesHelp { } {
     global version modroot
     puts stderr "\ttigmint - Sets the environment for using tigmint 1.1.
↪2. \
     \nin the shared directory /share/apps/tigmint/1.1.2/miniconda/4.5.1."
}

module-whatis "(Name________) tigmint"
module-whatis "(Version_____) 1.1.2 with Conda 4.5.1"
module-whatis "(System______) x86_64-redhat-linux"
module-whatis "(Libraries___) "

set              topdir              /share/apps/tigmint/1.1.2/miniconda/
↪4.5.1

conflict tigmint
module load python/3.6.5_miniconda-4.5.1
module load bwa/0.7.15_intel-2017_update-1
module load bedtools/2.26.0_intel-2017_update-1
module load samtools/1.3.1_intel-2017_update-1
module load pigz/2.4_intel-19.0.4

prepend-path        PATH                $topdir/bin
```

Create the needed folder and place it:

```
$ sudo mkdir /share/apps/modules/tigmint
$ sudo mv 1.1.2_miniconda-4.5.1 /share/apps/modules/tigmint/
```

Authors

- Hamilton Tobon-Mosquera <htobonm@eafit.edu.co>

3.3.70 TopHat

TopHat is a bioinformatic sequence analysis packet tool for fast and high-performance alignment of cDNA sequencing readings generated by transcriptomic technologies (eg, RNA-Seq) using Bowtie first and then mapping to a reference genome for discover de novo RNA splice sites. TopHat aligns RNA-Seq readings with mammal size genomes

TopHat 2.1.1

Table of Contents

Basic information

- **Installation Date:** 07/11/2018
- **URL:** http://ccb.jhu.edu/software/tophat/index.shtml
- **License:** Boost Software License, Version 1.0
- **Installed on:** Apolo II & Cronos

Tested on (Requirements)

- **Dependencies to run TopHat:**
  - Bowtie2
  - Intel compiler (C y C++)

Installation

After solving the previously mentioned dependencies, you can proceed with the installation.

1. Download the latest version of the software (Source code - tar.gz) (ccb.jhu.edu/software/tophat/downloads/tophat-2.1.1.Linux_x86_64.tar.gz):

```
tar -zxvf tophat-2.1.1.Linux_x86_64.tar.gz
sudo mkdir -p /share/apps/tophat/2.1.1/gcc-4.4.7/bin
cp tophat-2.1.1.Linux_x86_64/* /share/apps/tophat/2.1.1/gcc-4.4.7/bin
```

Module

```
##%Module1.0#####################################################################
↪##
##
## module load tophat/2.1.1_gcc-4.4.7
##
## /share/apps/modules/tophat/2.1.1_gcc-4.4.7
## Written by Juan Pablo Alcaraz Florez
##

proc ModulesHelp {} {
    global version modroot
    puts stderr "Sets the environment for using tophat 2.1.1\
                \nin the shared directory \
                \n/share/apps/tophat/2.1.1/gcc-4.4.7/\
                \nbuilded with gcc-4.4.7"
}

module-whatis "(Name________) tophat"
module-whatis "(Version_____) 2.1.1"
module-whatis "(Compilers___) gcc-4.4.7"
module-whatis "(System______) x86_64-redhat-linux"
module-whatis "(Libraries___) bowtie2"

# for Tcl script use only
set         topdir        /share/apps/tophat/2.1.1/gcc-4.4.7/
set         version       2.1.1
set         sys           x86_64-redhat-linux

conflict tophat
module load bowtie2/2.3.4.1_intel-18.0.2

prepend-path PATH    $topdir/bin
```

References

- Manual dentro del paquete del software

Authors

- Juan Pablo Alcaraz Flórez

3.3.71 Trans-ABySS

Trans-ABySS[1] is a De novo assembly of RNAseq data using *ABySS*.

Trans-ABySS 2.0.1

Basic information

- **Deploy date:** 22 October 2019
- **Official Website:** https://github.com/bcgsc/transabyss
- **License:** GNU GPL v3
- **Installed on:** *Apolo II*
- **Dependencies:**
  - *ABySS*
  - Python 3.x
  - python-igraph 0.7.x
  - *BLAT*

Installation

1. Before installing Trans-ABySS install *ABySS* and *BLAT*. Then run:

```
$ module load python/3.6.5_miniconda-4.5.1

# Instead of "biology" any name can be used.
$ conda create -n biology python=3.7
$ conda activate biology
$ pip install python-igraph
```

Test:

```
$ module load blat/36_intel-19.0.4
$ module load abyss/2.2.3_gcc-5.4.0
$ git clone https://github.com/bcgsc/transabyss.git
$ cd transabyss
$ bash sample_dataset/assemble.sh
$ rm -rf sample_dataset/merged.fa  sample_dataset/test.k25  sample_
↪dataset/test.k32
```

Install the scripts. From the Trans-ABySS' directory:

```
$ sudo mkdir -p /share/apps/transabyss/2.0.1/miniconda-4.5.1
$ sudo cp -r * /share/apps/transabyss/2.0.1/miniconda-4.5.1/*
```

2. Create and place the needed module file. Create a file with the following content:

[1] TransAbyss. (2018, Feb 19). Retrieved October 22, 2019, from https://github.com/bcgsc/transabyss

Listing 61: 2.0.1_miniconda-4.5.1

```
#%Module1.0#####################################################################
↪#####
##
## module load transabyss/2018
##
## /share/apps/modules/transabyss/2018
## Written by Hamilton Tobon Mosquera.
##

proc ModulesHelp {} {
        global version modroot
        puts stderr "Sets the environment for using Trans-AByss 2018␣
↪update \
        \nin the shared directory \
        \n/share/apps/transabyss/2.0.1/miniconda-4.5.1"
}

module-whatis "(Name________) transabyss"
module-whatis "(Version_____) 2.0.1 Miniconda 4.5.1"
module-whatis "(System______) x86_64-redhat-linux"
module-whatis "(Libraries___) "

# for Tcl script use only
set         topdir        /share/apps/transabyss/2.0.1/miniconda-4.5.1
set         version       2.0.1
set         sys           x86_64-redhat-linux

conflict transabyss
module load blat/36_intel-19.0.4
module load abyss/2.2.3_gcc-5.4.0

prepend-path   PATH            $topdir
prepend-path   PATH            $topdir/bin
```

Create the needed folder and place it:

```
$ sudo mkdir /share/apps/modules/transabyss
$ sudo mv 2.0.1_miniconda-4.5.1 /share/apps/modules/transabyss/
```

Usage

Always activate the conda environment where python-igraph was installed, in this case:

```
conda activate biology
```

Authors

- Hamilton Tobon-Mosquera <htobonm@eafit.edu.co>

3.3.72 Transrate

Transrate[1] is software for de-novo transcriptome assembly quality analysis. It examines your assembly in detail and compares it to experimental evidence such as the sequencing reads, reporting quality scores for contigs and assemblies. This allows you to choose between assemblers and parameters, filter out the bad contigs from an assembly, and help decide when to stop trying to improve the assembly.

Transrate 1.0.3

Table of Contents

Basic information

- **Official Website:** http://hibberdlab.com/transrate/
- **License:** MIT License
- **Installed on:** *Apolo II* , *Cronos*

Tested on (Requirements)

- **OS base:** CentOS (x86_64) $\geq$ 6.6 (Rocks 6.2)

Installation

The following procedure is the easiest way to install Transrate in a cluster.

1. Download the binaries and unzip the file:

```
$ mkdir build && cd build
$ wget https://bintray.com/artifact/download/blahah/generic/transrate-1.0.3-linux-
↪x86_64.tar.gz
$ tar -xzvf transrate-1.0.3-linux-x86_64.tar.gz
```

2. Then we add the directory path to the PATH environment variable:

[1] TransRate: reference free quality assessment of de-novo transcriptome assemblies (2016). Richard D Smith-Unna, Chris Boursnell, Rob Patro, Julian M Hibberd, Steven Kelly. Genome Research doi: http://dx.doi.org/10.1101/gr.196469.115

```
$ cd transrate-1.0.3-linux-x86_64
$ export PATH=$(pwd):$PATH
```

3. After, the ruby module is loaded (version 2.6.4)

```
$ module load ruby/2.6.4p104_intel-19.0.4
```

4. Now, we proceed to install the dependencies and the application itself:

```
$ transrate --install-deps all
$ gem install transrate
```

5. After the installation is completed you have to create the corresponding module for Transrate 1.0.3

```
##Module1.0#####################################################################
↪######
##
## modulefile transrate/1.0.3_ruby-2.6.4p104_intel-19.0.4
##
## Written by Hamilton Tobon Mosquera.

proc ModulesHelp { } {
    global version modroot
    puts stderr "\ttransrate - Software for de-novo transcriptome␣
↪assembly \
                \n\tquality analysis.\n"
}

module-whatis "\n\n\tSets the environment for using transrate 1.0.3 ruby␣
↪gem."

conflict transrate

# for Tcl script use only
set     topdir       ~/.local
set     version      1.0.3
set     sys          x86_64-redhat-linux

module load ruby/2.6.4p104_intel-19.0.4
```

Running Examples

This guide will take you through the basic ways of using Transrate. It's worth reading through once even if you're familiar with running command-line tools, as it provides guidance about proper selection of input data[1] .

Get Some Data

If you don't have your own data, you can use our small example dataset to try out the contig and read-based metrics:

```
$ mkdir example && cd example
$ wget https://bintray.com/artifact/download/blahah/generic/example_data.tar.gz
```

(continues on next page)

[1] TransRate: reference free quality assessment of de-novo transcriptome assemblies (2016). Richard D Smith-Unna, Chris Boursnell, Rob Patro, Julian M Hibberd, Steven Kelly. Genome Research doi: http://dx.doi.org/10.1101/gr.196469.115

(continued from previous page)

```
$ tar -xzvf example_data.tar.gz
$ cd example_data
```

To continue reviewing the examples worked with this file and the functionality of transrate check the following link: http://hibberdlab.com/transrate/getting_started.html

References

Authors

- Santiago Hidalgo Ocampo <shidalgoo1@eafit.edu.co>
- Samuel David Palacios <sdpalaciob@eafit.edu.co>

3.3.73 trimmomatic

Description

Trimmomatic is a fast, multithreaded command line tool that can be used to trim and crop Illumina (FASTQ) data as well as to remove adapters. These adapters can pose a real problem depending on the library preparation and downstream application. There are two major modes of the program: Paired end mode and Single end mode. The paired end mode will maintain correspondence of read pairs and also use the additional information contained in paired reads to better find adapter or PCR primer fragments introduced by the library preparation process. Trimmomatic works with FASTQ files (using phred + 33 or phred + 64 quality scores, depending on the Illumina pipeline used). Files compressed using either „gzip or „bzip2 are supported, and are identified by use of „.gz or „.bz2 file extensions.

0.36

- **Installation date:** 01/03/2017
- **URL:** http://www.usadellab.org/cms/?page=trimmomatic
- **Apolo version:** Apolo II
- **License:** GNU GENERAL PUBLIC LICENSE, Version 3

Tabe of Contents

Installation

These are the steps to install **Trimmomatic:**

1. Download the latest software version (Binaries - zip) (0.36):

```
cd /home/$USER/apps/trimmomatic/src
wget http://www.usadellab.org/cms/uploads/supplementary/Trimmomatic/Trimmomatic-0.36.
↪zip
unzip Trimmomatic-0.36.zip
```

2. For installation, the following steps must be done:

```
cd Trimmomatic-0.36
sudo mkdir -p /share/apps/trimmomatic/0.36/lib
sudo cp -r uk/ org/ net/ trimmomatic-0.36.jar /share/apps/fastqc/0.11.5/lib
```

Module

```
##%Module1.0#####################################################################
##
## module trimmomatic/0.36
##
## /share/apps/modules/trimmomatic/0.36     Written by Mateo Gomez-Zuluaga
##

proc ModulesHelp { } {
    puts stderr "\ttrimmomatic/0.36 - sets the Environment for Trimmomatic in \
    \n\tthe share directory /share/apps/trimmomatic/0.36\n"
}

module-whatis "\n\n\tSets the environment for using Trimmomatic 0.36 \
               \n\tprecompiled\n"

# for Tcl script use only
set       topdir     /share/apps/trimmomatic/0.36
set       version    0.36
set       sys        x86_64-redhat-linux

conflict trimmomatic

module load java/jdk-1.8.0_112

prepend-path CLASSPATH                      $topdir/lib/trimmomatic-0.36.jar
```

Mode of use

Load the necessary environment through the module:

```
module load trimmomatic/0.36
```

SLURM Template

```
#!/bin/sh
#SBATCH --partition=longjobs
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=1
#SBATCH --time=10:00
#SBATCH --job-name=trimmomatic_example
#SBATCH -o result_%N_%j.out
#SBATCH -e result_%N_%j.err
#SBATCH --mail-type=END,FAIL
#SBATCH --mail-user=mgomezz@eafit.edu.co

# Don't share environment variables
export SBATCH_EXPORT=NONE
export OMP_NUM_THREADS=1

module load trimmomatic/0.36

java -jar /share/apps/trimmomatic/0.36/lib/trimmomatic-0.36.jar PE -phred33 input_
↪forward.fq.gz input_reverse.fq.gz output_forward_paired.fq.gz output_forward_
↪unpaired.fq.gz
output_reverse_paired.fq.gz output_reverse_unpaired.fq.gz ILLUMINACLIP:TruSeq3-PE.
↪fa:2:30:10 LEADING:3 TRAILING:3 SLIDINGWINDOW:4:15 MINLEN:36
```

TO-DO

References

- http://www.usadellab.org/cms/?page=trimmomatic

Author

- Mateo Gómez Zuluaga

3.3.74 Trinity

Trinity[1] developed at the Broad Institute and the Hebrew University of Jerusalem, represents a novel method for the efficient and robust *de novo* reconstruction of transcriptomes from RNA-seq data. Trinity combines three independent software modules: Inchworm, Chrysalis, and Butterfly, applied sequentially to process large volumes of RNA-seq reads. Trinity partitions the sequence data into many individual de Bruijn graphs, each representing the transcriptional complexity at a given gene or locus, and then processes each graph independently to extract full-length splicing isoforms and to tease apart transcripts derived from paralogous genes. Briefly, the process works like so:

> Inchworm assembles the RNA-seq data into the unique sequences of transcripts, often generating full-length transcripts for a dominant isoform, but then reports just the unique portions of alternatively spliced transcripts.
>
> Chrysalis clusters the Inchworm contigs into clusters and constructs complete *De Bruijn* graphs for each cluster. Each cluster represents the full transcriptonal complexity for a given gene (or sets of genes that share sequences in common). Chrysalis then partitions the full read set among these disjoint graphs.

[1] Trinityrnaseq. (n.d.). trinityrnaseq/trinityrnaseq. Retrieved October 1, 2019, from https://github.com/trinityrnaseq/trinityrnaseq/wiki.

Butterfly then processes the individual graphs in parallel, tracing the paths that reads and pairs of reads take within the graph, ultimately reporting full-length transcripts for alternatively spliced isoforms, and teasing apart transcripts that corresponds to paralogous genes.

TRINITY 2.8.5

Table of Contents

Basic information

- **Official Website:** https://github.com/trinityrnaseq/trinityrnaseq/wiki
- **Download Website:** https://github.com/trinityrnaseq/trinityrnaseq/releases
- **License:** trinityrnaseq License
- **Installed on:** Apolo II and Cronos

Tested on (Requirements)

- **OS base:** CentOS (x86_64) $\geq$ 6.6 (Rocks 6.2)
- **Dependencies to run trinity:**
  - bowtie2 (tested on version 2.3.2)
  - jellyfish (tested on version 2.3.0) depends on yaggo
  - yaggo (tested on version 1.5.10)
  - salmon (tested on version 0.14.1)
  - samtools (tested on version 1.3.1)
  - Python 2.7 or 3.X with numpy
  - CMAKE (teste on version 3.7.1)

Installation

The following procedure is the easiest way to install Trinity v2.8.5 in a cluster.

1. Download the tar.gz of trinity version 2.8.5 from the release page into a cluster location.

```
$ wget https://github.com/trinityrnaseq/trinityrnaseq/archive/Trinity-v2.8.5.tar.
↪gz
```

2. Now, because trinity depends on other tools, make sure you have them compiled and installed in your cluster, else trinity won't run.

   - https://salmon.readthedocs.io/en/latest/salmon.html
   - http://www.genome.umd.edu/jellyfish.html
   - http://bowtie-bio.sourceforge.net/bowtie2/index.shtml
   - https://github.com/samtools/samtools

   Note: For jellyfish, the documentation for version 2.3.0 says you don't have to compile yaggo. It didn't work for us, we compiled yaggo, and then, it worked. Also, the Intel compiler doesn't work to compile jellyfish, use gcc instead as recommended in the jellyfish documentation.

3. After you have all the dependencies installed, load them so trinity will be able to use them.

```
$ module load bowtie2 samtools jellyfish salmon python/3.6.5_miniconda-4.
↪5.1 intel/19.0.4
$ echo "We load python because Apolo uses different versions of python,␣
↪that's how Apolo works."
```

   Note: In this example, we loaded the Intel compiler icc v19.0.4, because trinity does support it, but we have to do some changes first.

4. Now, we start configuring some components that are inside the directory you extracted from the tar.gz file.

```
$ cd trinityrnaseq-Trinity-v2.8.5/trinity-plugins
$ emacs -nw Makefile
```

   Go to where it says "parafly_target:"

   - Delete the first line that contains `tar -zxvf ${PARAFLY_CODE}.tar.gz && \`
   - Then, search for the `-fopenmp` flags and change them to `-qopenmp`
   - Save the file.

   After this, extract the ParaFly tar file and go inside the directory.

```
$ tar xfz ParaFly-0.1.0.tar.gz
$ cd ParaFly-0.1.0
```

   Now, lets configure one last thing:

```
$ emacs -nw configure
```

   Starting line 3125 inside `configure`, you should see something like this:

```
case $CXX in
    ++*) AM_CXXFLAGS="-pedantic -fopenmp -Wall -Wextra -Wno-long-long -
↪Wno-deprecated $AM_CXXFLAGS"
;;
```

(continues on next page)

(continued from previous page)

```
    sunCC*) AM_CXXFLAGS="-library=stlport4 -xopenmp -xvpara -fast $AM_
↪CXXFLAGS"
;;
    icpc*) AM_CXXFLAGS="-Wall -openmp $AM_CXXFLAGS"
;;
esac
```

Go to `icpc` and change the `-openmp` to `-qopenmp`

Note: You need to be inside the trinityrnaseq-Trinity-v2.8.5/trinity-plugins/ to make this changes.

5. We may compile Trinity V2.8.5 inside the directory you extracted from the tar.gz file.

```
$ cd trinityrnaseq-Trinity-v2.8.5
$ make -j4
$ make -j4 plugins
```

6. Now, Trinity's installer doesn't work well, it copies all the files inside the main trinity directory to /usr/local/bin, so for us to install it correctly, we had to delete some files manually and change the name of the main tirnity directory.

   Based on the Anaconda setup, this is how Apolo has Trinity installed.

```
$ cd trinityrnaseq-Trinity-v2.8.5
$ rm -Rf bioconda_recipe/ trinityrnaseq.wiki/ Docker/
$ cd Butterfly
$ rm -Rf build_jar.xml src/ jar-in-jar-loader.zip
$ cd ..
$ cd Chrysalis
$ rm -Rf aligns/ analysis/ base/ build/ chrysalis.notes CMakeLists.txt␣
↪Makefile system/ util/
$ cd ..
$ cd Inchworm
$ rm -Rf build/ CMakeLists.txt Makefile src/
$ cd ..
$ rm Changelog.txt Makefile notes README.md
$ cd ..
$ mv trinityrnaseq-Trinity-v2.8.5/ 19.0.4
$ sudo mv 19.0.4/ /share/apps/trinity/2.8.5/intel/
```

   You have Trinity v2.8.5 installed inside the /share/apps/trinity/2.8.5/intel/19.0.4 directory

Note: The second to last line is the change the name of the directory, this is easier when creating the module file.

7. After the installation is completed you have to create the corresponding module for Trinity 2.8.5.

```
##%Module1.0#####################################################################
↪#####
##
## module load trinity/2.8.5
##
## /share/apps/modules/trinity/2.8.5
## Written by Tomas David Navarro Munera
```

(continues on next page)

(continued from previous page)

```
##

proc ModulesHelp {} {
     puts stderr "Sets the environment for using Trinity 2.8.5\
                  \nin the shared directory /share/apps/trinity/2.8.5/
↪intel/19.0.4"
}

module-whatis "(Name________) Trinity"
module-whatis "(Version_____) 2.8.5"
module-whatis "(Compilers___) intel-19.0.4
module-whatis "(System______) x86_64-redhat-linux"
module-whatis "(Libraries___) "

# for Tcl script use only
set         topdir        /share/apps/trinity/2.8.5/intel/19.0.4
set         version       2.8.5
set         sys           x86_64-redhat-linux

conflict trinity
module load jellyfish/2.3.0
module load samtools/1.3.1_intel-2017_update-1
module load salmon/0.14.1
module load bowtie2/2.3.2_gcc-4.4.7
module load intel/19.0.4
module load python/3.6.5_miniconda-4.5.1

prepend-path              PATH        $topdir
```

Running Example

In this section, there is an example run that Trinity already has.

1. First, we create a conda environment, in able to run Trinity.

```
$ conda create --name trinity
$ conda activate trinity
$ pip install numpy
$ module load trinity/2.8.5
$ cd /share/apps/trinity/2.8.5/intel/19.0.4/sample_data/test_Trinity_Assembly/
$ ./runMe.sh
```

Note: The python version in this example is the one we loaded at the beginning of the installation.

References

Trinity - Trinity Official website. Retrieved Octubre 4, 2019, from https://github.com/trinityrnaseq/trinityrnaseq/wiki

Installing Trinity - Trinity Official Website. Retrieved Octubre 4, 2019, from https://github.com/trinityrnaseq/trinityrnaseq/wiki/Installing-Trinity

Authors

- Tomas David Navarro Munera <tdnavarrom@eafit.edu.co>

Table of Contents

Trinity 2.4.0

- **Installation date:** 01/03/2017
- **URL:** https://github.com/trinityrnaseq/trinityrnaseq/wiki
- **Apolo version:** Apolo II
- **License:** Copyright (c) 2014, trinityrnaseq

Dependencies

- GNU Gcc >= 4.9.4
- Intel Parallel Studio XE Cluster Edition 2017 Update 1
- bowtie2 >= 2.3.0
- ncbi-blast+ >= 2.6.0
- R and edgeRsource("https://bioconductor.org/biocLite.R")
- **r >= 3.3.**
  - edgeR
  - DESeq2
  - ROTS
  - qvalue
  - goseq

Installation

After solving the aforementioned dependencies, you can proceed with the installation of Trinity.

1. Download the binaries

```
wget https://github.com/trinityrnaseq/trinityrnaseq/archive/Trinity-v2.4.0.tar.gz
tar xf Trinity-v2.4.0.tar.gz
```

2. Compilation (makefile)

```
cd trinityrnaseq-Trinity-v2.4.0/
module load intel/2017_update-1
#change -openmp for -fopenmp in the following scripts
emacs Chrysalis/Makefile_icpc
emacs Inchworm/configure
emacs trinity-plugins/parafly-code/configure
emacs trinity-plugins/parafly-code/configure.ac
make TRINITY_COMPILER=intel 2>&1 | tee trinity-make.log
make TRINITY_COMPILER=intel plugins 2>&1 | tee trinity-make-plugins.log
```

3. Test the installation

```
module load ncbi-blast/2.6.0_x86_64
module load bowtie2/2.3.0_intel-2017_update-1
cd sample_data/test_Trinity_Assembly/
./runMe.sh
```

Module

```
##%Module1.0#####################################################################
##
## module trinity/2.4.0_intel-2017_update-1
##
## /share/apps/modules/trinity/2.4.0_intel-2017_update-1     Written by Mateo Gomez-
↪Zuluaga
##

proc ModulesHelp { } {
    puts stderr "\ttrinity/2.4.0_intel-2017_update-1 - sets the Environment for␣
↪Trinity in \
    \n\tthe share directory /share/apps/trinity/2.4.0/intel/2017_update-1\n"
}

module-whatis "\n\n\tSets the environment for using Trinity 2.4.0 \
               \n\tbuilded with Intel Parallel Studio XE 2017\n"

# for Tcl script use only
set       topdir     /share/apps/trinity/2.4.0/intel/2017_update-1
set       version    2.4.0
set       sys        x86_64-redhat-linux

conflict trinity
module load gcc/4.9.4
module load bowtie2/2.3.0_intel-2017_update-1
module load ncbi-blast/2.6.0_x86_64
module load r/3.3.2_intel_mkl_2017_update-1

prepend-path PATH    $topdir
```

Usage mode

```
module load trinity/2.4.0_intel-2017_update-1
```

References

- https://github.com/trinityrnaseq/trinityrnaseq/wiki/Installing%20Trinity

Author

- Mateo Gómez Zuluaga

3.3.75 USPEX

Description

USPEX (Universal Structure Predictor: Evolutionary Xtallography. . . and in Russian "uspekh" means "success" - owing to the high success rate and many useful results produced by this method) is a method developed by the Oganov laboratory since 2004. The problem of crystal structure prediction is very old and does, in fact, constitute the central problem of theoretical crystal chemistry. In 1988 John Maddox wrote that:

"One of the continuing scandals in the physical sciences is that it remains in general impossible to predict the structure of even the simplest crystalline solids from a knowledge of their chemical composition solids such as crystalline water (ice) are still thought to lie beyond mortals' ken". USPEX method/code solves this problem and is used by over 3000 researchers worldwide. The First Blind Test of Inorganic Crystal Structure Prediction shows that USPEX outperforms other methods in terms of efficiency and reliability. The method continues to be rapidly developed. In addition to crystal structure prediction, USPEX can work in other dimensionalities and predict the structure of nanoparticles, polymers, surfaces, interfaces and 2D-crystals. It can very efficiently handle molecular crystals (including those with flexible and very complex molecules) and can predict stable chemical compositions and corresponding crystal structures, given just the names of the chemical elements. In addition to this fully non-empirical search, USPEX allows one to predict also a large set of robust metastable structures and perform several types of simulations using various degrees of prior knowledge.

USPEX can also be used for finding low-energy metastable phases, as well as stable structures of nanoparticles, surface reconstructions, molecular packings in organic crystals, and for searching for materials with desired physical (mechanical, electronic) properties. The USPEX code is based on an efficient evolutionary algorithm developed by A.R. Oganov's group, but also has options for using alternative methods (random sampling, metadynamics, corrected particle swarm optimization algorithms). USPEX is interfaced with many ab initio codes, such as VASP, SIESTA, GULP, Quantum Espresso, CP2K, CASTEP, LAMMPS, and so on.

Test of USPEX: 40-atom cell of MgSiO3 post-perovskite. Left - structure search using local optimisation of random structures, Right - evolutionary search with USPEX. While random search did not produce the correct structure even after 120000 steps, USPEX found the stable structure in fewer than 1000 steps.

9.4.4

Table of Contents

Basic information

- **Installation Date:** 19/04/2017
- **URL:** http://uspex-team.org/
- **Apolo version:** II
- **License:** open source

Requirements

- Octave 3.4 (recommended), currently use Octave 4.0.3
- python >= 2.7.11

Installation

1. Download the installation medium of the following link https://uspex-team.org/en/uspex/downloads and enter with the credentials provided by Jorge David.

2 .Upload the file **USPEX-9.4.4.tar.gz** to Apollo and perform the following steps:

```
scp USPEX-9.4.4.tar.gz 'USERNAME'@apolo:/home/$USER/apps/uspex/src
cd /home/$USER/apps/uspex/src
tar xf USPEX-9.4.4.tar.gz
cd USPEX-9.4.4
mkdir -p /share/apps/uspex/9.4.4
module load octave/4.0.3_intel-2017_update-1 python/2.7.12_intel-2017_update-1
./install.sh
```

- The installation will automatically detect the octave PATH; The installation path for USPEX must be indicated (/share/apps/uspex/9.4.4)

3. The following file contains the Matlab or Octave PATH that will be used, in this case check that it has the path to the octave binary.

- /share/apps/uspex/9.4.4/CODEPATH

4. To "guarantee" the integration with Octave it is necessary to run the following commands:

```
files=$(grep -lR "echo -e" /share/apps/uspex/9.4.4)
for file in $files; do sed -i -- 's/echo -e/echo/g' $file; done
```

Module

```
##%Module1.0#####################################################################
##
## module load uspex//share/apps/modules/uspex/9.4.4
##
## /share/apps/uspex/9.4.4
## Written by Mateo Gómez-Zuluaga
##

proc ModulesHelp {} {
    global version modroot
    puts stderr "uspex/9.4.4 - sets the Environment for use USPEX in \
        \n\t\tthe shared directory /share/apps/uspex/9.4.4"
}

module-whatis "(Name_________) uspex"
module-whatis "(Version______) 9.4.4"
module-whatis "(Compiler_____) "
module-whatis "(System_______) x86_64-redhat-linux"
module-whatis "(Libraries____) "

# for Tcl script use only
set         topdir        /share/apps/uspex/9.4.4
set         version       9.4.4
set         sys           x86_64-redhat-linux

conflict uspex

module load octave/4.0.3_intel-2017_update-1
module load python/2.7.12_intel-2017_update-1

prepend-path PATH   $topdir

setenv       USPEXPATH  /share/apps/uspex/9.4.4/src
```

Mode of use

module load uspex/9.4.4

TO-DO

References

- User manual USPEX 9.4.4

Author

- Mateo Gómez Zuluaga

3.3.76 VASP

VASP is a complex package for performing ab-initio quantum-mechanical molecular dynamics (MD) simulations using pseudopotentials or the projector-augmented wave method and a plane wave basis set. The approach implemented in VASP is based on the (finite-temperature) local-density approximation with the free energy as variational quantity and an exact evaluation of the instantaneous electronic ground state at each MD time step. VASP uses efficient matrix diagonalisation schemes and an efficient Pulay/Broyden charge density mixing. These techniques avoid all problems possibly occurring in the original Car-Parrinello method, which is based on the simultaneous integration of electronic and ionic equations of motion. The interaction between ions and electrons is described by ultra-soft Vanderbilt pseudopotentials (US-PP) or by the projector-augmented wave (PAW) method. US-PP (and the PAW method) allow for a considerable reduction of the number of plane-waves per atom for transition metals and first row elements. Forces and the full stress tensor can be calculated with VASP and used to relax atoms into their instantaneous ground-state.

Table of Contents

VASP/5.4.4 INTEL

- **Installation date:** 27/07/2017
- **URL:** https://www.vasp.at/
- **Apolo version:** Apolo II
- **License:** VASP is not public-domain or share-ware, and will be distributed only after a license contract has been signed. Please visit VASP homepage to know more details on obtaining the license.

Dependencies

- Intel Parallel Studio XE Cluster Edition >= 2017-U1

Installation

After solving the aforementioned dependencies, you can proceed with the installation.

1. Download the binaries

```
tar -zxvf vasp.5.4.4.tar.gz
cd vasp.5.4.4
```

2. Compilation (makefile)

```
cp arch/makefile.include.linux_intel ./makefile.include
```

It should look like the following:

```
# Precompiler options
CPP_OPTIONS= -DHOST=\"LinuxIFC\"\
             -DMPI -DMPI_BLOCK=64000 \
             -DIFC -DPGF90 -DNGZhalf -DMKL_ILP64 \
         -Duse_collective \
             -DscaLAPACK \
             -DCACHE_SIZE=32000 \
             -Davoidalloc \
             -Duse_bse_te \
             -Dtbdyn \
             -Duse_shmem

CPP         = fpp -f_com=no -free -w0  $*$(FUFFIX) $*$(SUFFIX) $(CPP_OPTIONS)

FC          = mpiifort
FCL         = mpiifort -mkl=cluster -lstdc++

FREE        = -free -names lowerc

ase

FFLAGS      = -FR -names lowercase -assume byterecl -I$(MKLROOT)/include/fftw -w
OFLAG       = -O3 -xCORE-AVX2
OFLAG_IN    = $(OFLAG)
DEBUG       = -O0

MKL_PATH    = $(MKLROOT)/lib/intel64
BLAS        = -mkl=cluster
LAPACK      =
BLACS       =
SCALAPACK   =

OBJECTS     = fftmpiw.o fftmpi_map.o fft3dlib.o fftw3d.o

INCS        =-I$(MKLROOT)/include/fftw

LLIBS       = $(SCALAPACK) $(LAPACK) $(BLAS)


OBJECTS_O1 += fftw3d.o fftmpi.o fftmpiw.o
OBJECTS_O2 += fft3dlib.o

# For what used to be vasp.5.lib
CPP_LIB     = $(CPP)
FC_LIB      = $(FC)
CC_LIB      = icc
CFLAGS_LIB = -O
FFLAGS_LIB = -O1
FREE_LIB    = $(FREE)

OBJECTS_LIB= linpack_double.o getshmem.o

# For the parser library
```

(continues on next page)

(continued from previous page)

```
CXX_PARS    = icpc

LIBS        += parser
LLIBS       += -Lparser -lparser -lstdc++

# Normally no need to change this
SRCDIR      = ../../src
BINDIR      = ../../bin

#================================================
# GPU Stuff

CPP_GPU     = -DCUDA_GPU -DRPROMU_CPROJ_OVERLAP -DUSE_PINNED_MEMORY -DCUFFT_MIN=28 -
↪UscaLAPACK

OBJECTS_GPU = fftmpiw.o fftmpi_map.o fft3dlib.o fftw3d_gpu.o fftmpiw_gpu.o

CC          = icc
CXX         = icpc
CFLAGS      = -fPIC -DADD_ -Wall -openmp -DMAGMA_WITH_MKL -DMAGMA_SETAFFINITY -
↪DGPUSHMEM=300 -DHAVE_CUBLAS

CUDA_ROOT  ?= /usr/local/cuda/
NVCC        := $(CUDA_ROOT)/bin/nvcc -ccbin=icc
CUDA_LIB    := -L$(CUDA_ROOT)/lib64 -lnvToolsExt -lcudart -lcuda -lcufft -lcublas

GENCODE_ARCH     := -gencode=arch=compute_30,code=\"sm_30,compute_30\" \
                    -gencode=arch=compute_35,code=\"sm_35,compute_35\" \
                    -gencode=arch=compute_60,code=\"sm_60,compute_60\"

MPI_INC     = $(I_MPI_ROOT)/include64/
```

3. Compilation

```
module load intel/2017_update-1 mkl/2017_update-1 impi/2017_update-1
make all 2>&1 | tee vasp-make.log
```

Module

```
##%Module1.0#####################################################################
##
## module load vasp/5.4.4_intel_17.0.1
##
## /share/apps/modules/vasp/5.4.4_intel_17.0.1
## Written by Mateo Gómez-Zuluaga
##

proc ModulesHelp {} {
     global version modroot
     puts stderr "Sets the environment for using VASP 5.4.4\
          \nin the shared directory /share/apps/vasp/5.4.4/intel_17.0.1\
          \nbuilded with Intel Parallel Studio XE Cluster Edition 2017 Update 1."
}
```

(continues on next page)

(continued from previous page)

```
module-whatis "(Name________) vasp"
module-whatis "(Version_____) 5.4.4"
module-whatis "(Compilers___) intel_17.0.1"
module-whatis "(System______) x86_64-redhat-linux"
module-whatis "(Libraries___) mkl"

# for Tcl script use only
set         topdir        /share/apps/vasp/5.4.4/intel_17.0.1
set         version       5.4.4
set         sys           x86_64-redhat-linux

conflict vasp

module load intel/2017_update-1
module load impi/2017_update-1
module load mkl/2017_update-1

prepend-path        PATH                    $topdir/bin
```

Usage mode

```
module load trinity/2.4.0_intel-2017_update-1
```

References

- README
- https://software.intel.com/en-us/articles/building-vasp-with-intel-mkl-and-intel-compilers
- http://cms.mpi.univie.ac.at/wiki/index.php/Installing_VASP#For_vasp.5.4.1.24Jun15
- http://cms.mpi.univie.ac.at/wiki/index.php/Installing_VASP

Author

- Mateo Gómez Zuluaga

Table of Contents

VASP/5.4.4 GNU

- **Installation date:** 27/07/2017
- **URL:** https://www.vasp.at/
- **Apolo version:** Apolo II
- **License:** VASP is not public-domain or share-ware, and will be distributed only after a license contract has been signed. Please visit VASP homepage to know more details on obtaining the license.

Dependencies

- FFTW 3.3.7
- OpenBLAS 0.2.20 (with BLAS y LAPACK)
- ScaLapack 2.0.2
- OpenMPI 1.10.7

Installation

After solving the aforementioned dependencies, you can proceed with the installation.

1. Download the binaries

```
module load openmpi/1.10.7_gcc-5.5.0
module load fftw/3.3.7_gcc-5.5.0
module load openblas/0.2.20_gcc-5.5.0
module load scalapack/2.0.2_gcc-5.5.0
tar -zxvf vasp.5.4.4.tar.gz
cd vasp.5.4.4
```

2. Compilation (makefile), modify makefile.include

```
cp arch/makefile.include.linux_gnu ./makefile.include
```

It should look like the following:

```
...
CPP         = gcc -E -P $*$(FUFFIX) >$*$(SUFFIX) $(CPP_OPTIONS)
...
OFLAG       = -O3 -march=native
...
LIBDIR      =
BLAS        = -L/share/apps/openblas/0.2.20/gcc-5.5.0/lib -lopenblas
LAPACK      =
BLACS       =
SCALAPACK   = -L/share/apps/scalapack/2.0.2/gcc-5.5.0/lib -lscalapack
...
FFTW       ?= /share/apps/fftw/3.3.7/gcc-5.5.0
...
MPI_INC     = share/apps/openmpi/1.10.7/gcc-5.5.0/include
```

And

```
# Precompiler options
CPP_OPTIONS= -DHOST=\"LinuxGNU\" \
             -DMPI -DMPI_BLOCK=8000 \
             -Duse_collective \
             -DscaLAPACK \
             -DCACHE_SIZE=4000 \
             -Davoidalloc \
             -Duse_bse_te \
             -Dtbdyn \
             -Duse_shmem

CPP         = gcc -E -P $*$(FUFFIX) >$*$(SUFFIX) $(CPP_OPTIONS)

FC          = mpif90
FCL         = mpif90

FREE        = -ffree-form -ffree-line-length-none

FFLAGS      =
OFLAG       = -O3 -march=native
OFLAG_IN    = $(OFLAG)
DEBUG       = -O0

LIBDIR      =
BLAS        = -L/share/apps/openblas/0.2.20/gcc-5.5.0/lib -lopenblas
LAPACK      =
BLACS       =
SCALAPACK   = -L/share/apps/scalapack/2.0.2/gcc-5.5.0/lib -lscalapack

LLIBS       = $(SCALAPACK) $(LAPACK) $(BLAS)

FFTW        ?= /share/apps/fftw/3.3.7/gcc-5.5.0
LLIBS       += -L$(FFTW)/lib -lfftw3
INCS        = -I$(FFTW)/include

OBJECTS     = fftmpiw.o fftmpi_map.o  fftw3d.o  fft3dlib.o

OBJECTS_O1 += fftw3d.o fftmpi.o fftmpiw.o
OBJECTS_O2 += fft3dlib.o

# For what used to be vasp.5.lib
CPP_LIB     = $(CPP)
FC_LIB      = $(FC)
CC_LIB      = gcc
CFLAGS_LIB  = -O
FFLAGS_LIB  = -O1
FREE_LIB    = $(FREE)

OBJECTS_LIB= linpack_double.o getshmem.o

# For the parser library
CXX_PARS    = g++

LIBS        += parser
LLIBS       += -Lparser -lparser -lstdc++

# Normally no need to change this
```

(continues on next page)

(continued from previous page)

```
SRCDIR     = ../../src
BINDIR     = ../../bin

#=============================================
# GPU Stuff

CPP_GPU    = -DCUDA_GPU -DRPROMU_CPROJ_OVERLAP -DCUFFT_MIN=28 -UscaLAPACK # -DUSE_
↪PINNED_MEMORY

OBJECTS_GPU= fftmpiw.o fftmpi_map.o fft3dlib.o fftw3d_gpu.o fftmpiw_gpu.o

CC         = gcc
CXX        = g++
CFLAGS     = -fPIC -DADD_ -openmp -DMAGMA_WITH_MKL -DMAGMA_SETAFFINITY -DGPUSHMEM=300␣
↪-DHAVE_CUBLAS

CUDA_ROOT  ?= /usr/local/cuda
NVCC       := $(CUDA_ROOT)/bin/nvcc
CUDA_LIB   := -L$(CUDA_ROOT)/lib64 -lnvToolsExt -lcudart -lcuda -lcufft -lcublas

GENCODE_ARCH    := -gencode=arch=compute_30,code=\"sm_30,compute_30\" \
                   -gencode=arch=compute_35,code=\"sm_35,compute_35\" \
                   -gencode=arch=compute_60,code=\"sm_60,compute_60\"

MPI_INC    = /share/apps/openmpi/1.10.7/gcc-5.5.0/include
```

3. Compilation

```
make all 2>&1 | tee vasp-make.log
```

Module

```
#%Module1.0#####################################################################
##
## module load vasp/5.4.4_gcc-5.5.0
##
## /share/apps/modules/vasp/5.4.4_gcc-5.5.0
## Written by Andrés Felipe Zapata Palacio
##

proc ModulesHelp {} {
     global version modroot
     puts stderr "Sets the environment for using vasp 5.4.4\
                  \nin the shared directory \
                  \n/share/apps/vasp/5.4.4/gcc-5.5.0/\
                  \nbuilded with gcc-5.5.0"
}

module-whatis "(Name________) vasp"
module-whatis "(Version_____) 5.4.4"
module-whatis "(Compilers___) gcc-5.5.0"
module-whatis "(System______) x86_64-redhat-linux"
module-whatis "(Libraries___) fftw-3.3.7, openblas-0.2.20, openmpi-1.10.7, scaLapack-
↪2.0.2"
```

(continues on next page)

(continued from previous page)

```
# for Tcl script use only
set         topdir        /share/apps/vasp/5.4.4/gcc-5.5.0/
set         version       5.4.4
set         sys           x86_64-redhat-linux

conflict vasp
module load openmpi/1.10.7_gcc-5.5.0
module load fftw/3.3.7_gcc-5.5.0
module load openblas/0.2.20_gcc-5.5.0
module load scalapack/2.0.2_gcc-5.5.0

prepend-path    PATH                    $topdir/bin
```

Usage mode

```
module load vasp/5.4.4_gcc-5.5.0
```

References

- http://www.ivofilot.nl/posts/view/28/How+to+build+VASP+5.3.5+using+the+GNU+compiler+on+Linux+Ubuntu+14.04+LTS

Author

- Andrés Felipe Zapata Palacio

Table of Contents

VASP/5.4.4 - GNU - VASPsol

- **Installation date:** 14/02/2018
- **URL:** https://www.vasp.at/ and http://vaspsol.mse.ufl.edu/
- **Apolo version:** Apolo II

- **License:** VASP is not public-domain or share-ware, and will be distributed only after a license contract has been signed. Please visit VASP homepage to know more details on obtaining the license.

Dependencies

- FFTW 3.3.7
- OpenBLAS 0.2.20 (with BLAS y LAPACK)
- ScaLapack 2.0.2
- OpenMPI 1.10.7

Installation

After solving the aforementioned dependencies, you can proceed with the installation.

1. Download the binaries

```
git clone https://github.com/henniggroup/VASPsol.git
cd VASPsol
module load openmpi/1.10.7_gcc-5.5.0
module load fftw/3.3.7_gcc-5.5.0
module load openblas/0.2.20_gcc-5.5.0
module load scalapack/2.0.2_gcc-5.5.0
tar -zxvf vasp.5.4.4.tar.gz
cd vasp.5.4.4
```

2. Compilation (makefile), modify makefile.include

```
cp src/solvation.F /home/mgomezzul/apps/vaspsol/vasp.5.4.4/src/
```

It should look like the following:

```
CPP_OPTIONS= -DHOST=\"LinuxGNU\" \
             -DMPI -DMPI_BLOCK=8000 \
             -Duse_collective \
             -DscaLAPACK \
             -DCACHE_SIZE=4000 \
             -Davoidalloc \
             -Duse_bse_te \
             -Dtbdyn \
             -Duse_shmem \
             -Dsol_compat
```

And edit

```
...
CPP         = gcc -E -P $*$(FUFFIX) >$*$(SUFFIX) $(CPP_OPTIONS)
...
OFLAG       = -O3 -march=native
...
LIBDIR      =
BLAS        = -L/share/apps/openblas/0.2.20/gcc-5.5.0/lib -lopenblas
LAPACK      =
BLACS       =
```

(continues on next page)

(continued from previous page)

```
SCALAPACK  = -L/share/apps/scalapack/2.0.2/gcc-5.5.0/lib -lscalapack
...
FFTW      ?= /share/apps/fftw/3.3.7/gcc-5.5.0
...
MPI_INC    = share/apps/openmpi/1.10.7/gcc-5.5.0/include
```

And

```
# Precompiler options
CPP_OPTIONS= -DHOST=\"LinuxGNU\" \
             -DMPI -DMPI_BLOCK=8000 \
             -Duse_collective \
             -DscaLAPACK \
             -DCACHE_SIZE=4000 \
             -Davoidalloc \
             -Duse_bse_te \
             -Dtbdyn \
             -Duse_shmem

CPP        = gcc -E -P $*$(FUFFIX) >$*$(SUFFIX) $(CPP_OPTIONS)

FC         = mpif90
FCL        = mpif90

FREE       = -ffree-form -ffree-line-length-none

FFLAGS     =
OFLAG      = -O3 -march=native
OFLAG_IN   = $(OFLAG)
DEBUG      = -O0

LIBDIR     =
BLAS       = -L/share/apps/openblas/0.2.20/gcc-5.5.0/lib -lopenblas
LAPACK     =
BLACS      =
SCALAPACK  = -L/share/apps/scalapack/2.0.2/gcc-5.5.0/lib -lscalapack

LLIBS      = $(SCALAPACK) $(LAPACK) $(BLAS)

FFTW       ?= /share/apps/fftw/3.3.7/gcc-5.5.0
LLIBS      += -L$(FFTW)/lib -lfftw3
INCS       = -I$(FFTW)/include

OBJECTS    = fftmpiw.o fftmpi_map.o  fftw3d.o  fft3dlib.o

OBJECTS_O1 += fftw3d.o fftmpi.o fftmpiw.o
OBJECTS_O2 += fft3dlib.o

# For what used to be vasp.5.lib
CPP_LIB    = $(CPP)
FC_LIB     = $(FC)
CC_LIB     = gcc
CFLAGS_LIB = -O
FFLAGS_LIB = -O1
FREE_LIB   = $(FREE)

OBJECTS_LIB= linpack_double.o getshmem.o
```

(continues on next page)

(continued from previous page)

```
# For the parser library
CXX_PARS    = g++

LIBS        += parser
LLIBS       += -Lparser -lparser -lstdc++

# Normally no need to change this
SRCDIR      = ../../src
BINDIR      = ../../bin

#================================================
# GPU Stuff

CPP_GPU     = -DCUDA_GPU -DRPROMU_CPROJ_OVERLAP -DCUFFT_MIN=28 -UscaLAPACK # -DUSE_
↪PINNED_MEMORY

OBJECTS_GPU= fftmpiw.o fftmpi_map.o fft3dlib.o fftw3d_gpu.o fftmpiw_gpu.o

CC          = gcc
CXX         = g++
CFLAGS      = -fPIC -DADD_ -openmp -DMAGMA_WITH_MKL -DMAGMA_SETAFFINITY -DGPUSHMEM=300␣
↪-DHAVE_CUBLAS

CUDA_ROOT  ?= /usr/local/cuda
NVCC        := $(CUDA_ROOT)/bin/nvcc
CUDA_LIB    := -L$(CUDA_ROOT)/lib64 -lnvToolsExt -lcudart -lcuda -lcufft -lcublas

GENCODE_ARCH    := -gencode=arch=compute_30,code=\"sm_30,compute_30\" \
                   -gencode=arch=compute_35,code=\"sm_35,compute_35\" \
                   -gencode=arch=compute_60,code=\"sm_60,compute_60\"

MPI_INC     = /share/apps/openmpi/1.10.7/gcc-5.5.0/include
```

3. Compilation

```
make all 2>&1 | tee vasp-make.log
```

Module

```
##%Module1.0#####################################################################
##
## module load vasp/5.4.4_gcc-5.5.0
##
## /share/apps/modules/vasp/5.4.4_gcc-5.5.0
## Written by Andrés Felipe Zapata Palacio
##

proc ModulesHelp {} {
     global version modroot
     puts stderr "Sets the environment for using vasp 5.4.4\
                  \nin the shared directory \
                  \n/share/apps/vasp/5.4.4/gcc-5.5.0/\
                  \nbuilded with gcc-5.5.0"
```

(continues on next page)

(continued from previous page)

```
}

module-whatis "(Name_________) vasp"
module-whatis "(Version______) 5.4.4"
module-whatis "(Compilers____) gcc-5.5.0"
module-whatis "(System_______) x86_64-redhat-linux"
module-whatis "(Libraries____) fftw-3.3.7, openblas-0.2.20, openmpi-1.10.7, scaLapack-
↪2.0.2"

# for Tcl script use only
set         topdir        /share/apps/vasp/5.4.4/gcc-5.5.0/
set         version       5.4.4
set         sys           x86_64-redhat-linux

conflict vasp
module load openmpi/1.10.7_gcc-5.5.0
module load fftw/3.3.7_gcc-5.5.0
module load openblas/0.2.20_gcc-5.5.0
module load scalapack/2.0.2_gcc-5.5.0

prepend-path    PATH                    $topdir/bin
```

Usage mode

```
module load vasp/5.4.4_gcc-5.5.0
```

References

- http://www.ivofilot.nl/posts/view/28/How+to+build+VASP+5.3.5+using+the+GNU+compiler+on+Linux+Ubuntu+14.04+LTS
- https://github.com/henniggroup/VASPsol

Author

- Andrés Felipe Zapata Palacio

3.3.77 VMD

VMD VMD is designed for modeling, visualization, and analysis of biological systems such as proteins, nucleic acids, lipid bilayer assemblies, etc. It may be used to view more general molecules, as VMD can read standard Protein Data Bank (PDB) files and display the contained structure. VMD provides a wide variety of methods for rendering and coloring a molecule: simple points and lines, CPK spheres and cylinders, licorice bonds, backbone tubes and ribbons, cartoon drawings, and others. VMD can be used to animate and analyze the trajectory of a molecular dynamics (MD) simulation. In particular, VMD can act as a graphical front end for an external MD program by displaying and animating a molecule undergoing simulation on a remote computer.[1]

[1] Retrieved 08:25, December 06, 2019, from https://www.ks.uiuc.edu/Research/vmd/allversions/what_is_vmd.html

VMD 1.9.3

Table of Contents

Basic information

- **Official Website:** https://www.ks.uiuc.edu/Research/vmd/
- **Installed on:** *Apolo II*
- **License:** UIUC Open Source License - http://www.ks.uiuc.edu/Research/vmd/current/LICENSE.html

Dependencies

- Cuda 8.0
- GCC 5.4.0
- Python 2.7.15
- TCL 8.5

Installation

1. Get source code from https://www.ks.uiuc.edu/Development/Download/download.cgi?PackageName=VMD.

   Note: Yo hould register before download it.

2. Create a folder for VMD and decompress tar file. It will take out two folders (vmd-1.9.3 and plugins)

```
$ tar -xvf vmd-1.9.3.src.tar.gz
```

3. Go to vmd-1.9.3 folder and create a new directory called plugins and save the path to it because it will be used in the following step.

```
$ cd vmd-1.9.3
$ mkdir plugins
```

4. Before starting the compilation of vmd, we should compile plugins in the following way in the folder called plugins that was extracted in the first step.

```
$ cd plugins
$ module load netcdf/4.5.0_intel-17.0.1
$ module load gcc/7.4.0
$ export PLUGINDIR=<path_to_new_folder_plugins>
$ make LINUXAMD64 TCLINC=/usr/include TCLLIB=/usr/lib64/libtcl.so
$ make distrib
```

5. Return to vmd-1.9.3 folder and edit configure.options file according to the tools and dependencies to use during the compilation.

```
$ vim configure.options
$ LINUXAMD64 CUDA TCL PYTHON PTHREADS NUMPY ICC
```

6. Edit configure file and change the path of **$install_bin_dir** and **$install_library_dir** with the location bin and lib directories of vmd.

```
$ vim configure
$ $install_bin_dir="<path_to_bin_vmd_dir>";
$ $install_library_dir="<path_to_lib_vmd_dir>";
```

7. Run configure script.

```
$ ./configure
```

8. Go to src folder and edit the Makefile.

```
$ cd src
$ vim Makefile
```

Note: This file has a lot of paths that are not compatible because they are burned into the file.

- Put the correct path for the following variables: INCDIRS and LIBDIRS.

```
INCDIRS = -I/share/apps/python/2.7_miniconda-4.5.4/include/python2.7 -I/share/
↪apps/python/2.7_miniconda-4.5.4/lib/python2.7/site-packages/numpy/core/include -
↪I/usr/include -I../plugins/include -I../plugins/LINUXAMD64/molfile -I.
LIBDIRS      = -L/share/apps/cuda/8.0/lib64 -L/share/apps/python/2.7_miniconda-4.5.
↪4/lib/python2.7/config -L/usr/lib64 -L../plugins/LINUXAMD64/molfile
```

- Change NVCC binary with the correct path.

```
NVCC = /share/apps/cuda/8.0/bin/nvcc
```

- Put the correct architecture of Apolo in NVCCFLAGS, it is important to notice that the maximum verion of cuda that VMD supports is 8.0 so it is not compatible with Tesla V100 GPUs in Apolo. K80 GPUS is architecture compute_37 and sm_37 for arch and code variables.

```
NVCCFLAGS   = --ptxas-options=-v -gencode arch=compute_37,code=sm_37 --ftz=true  -
↪-machine 64 -O3  -DARCH_LINUXAMD64 $(DEFINES) $(INCDIRS)
```

- Also, change the GPU arthitecture in **.cu.ptx** section.

```
$(NVCC) $(DEFINES) --use_fast_math -I-L/share/apps/cuda/8.0/include -gencode␣
↪arch=compute_37,code=sm_37 -ptx $< -o ../LINUXAMD64/$@
```

- Change python version in LIBS, by default it will have python2.5.

```
LIBS = -Wl,-rpath -Wl,$$ORIGIN/ -lcudart_static -lpython2.7 -lpthread -lpthread  -
↪ltcl8.5  -lmolfile_plugin -ll -lm -ldl -lutil -lrt $(VMDEXTRALIBS)
```

9. Load modules needed for compilation and check dependencies.

```
$ module load python/2.7.15_miniconda-4.5.4
$ module load gcc/5.4.0
$ module load cuda/8.0
$ make depend
```

10. Finish with compilation and installation. Notice that in the process you should create folders for bin and lib specified in step 6.

```
$ mkdir -p /share/apps/vmd/1.9.3/bin
$ mkdir -p /share/apps/vmd/1.9.3/lib
$ make
$ sudo make install
```

11. Create the corresponding module of VMD 1.9.3.

```
$ mkdir /share/apps/modules/vmd/
$ vim /share/apps/modules/vmd/1.9.3

##%Module1.0#####################################################################
↪#####
##
## module load vmd/1.9.3
##
## /share/apps/modules/vmd/1.9.3
## Written by Manuela Carrasco Pinzon
##

proc ModulesHelp {} {
     global version modroot
     puts stderr "Sets the environment for using vmd 1.9.3\
                  \nin the shared directory /share/apps/vmd/1.9.3/"
}

module-whatis "(Name_________) vmd"
module-whatis "(Version______) 1.9.3"
module-whatis "(Compilers____) intel-17.0.1"
module-whatis "(System_______) x86_64-redhat-linux"
module-whatis "(Libraries____) "

# for Tcl script use only
set         topdir        /share/apps/vmd/1.9.3
set         version       1.9.3
set         sys           x86_64-redhat-linux

module load gcc/5.4.0
module load cuda/8.0
module load python/2.7.15_miniconda-4.5.4

prepend-path    PATH                    $topdir/bin

prepend-path    MANPATH                 $topdir/lib/doc
```

Authors

- Juan Diego Ocampo García <jocamp18@eafit.edu.co>

3.3.78 vsearch

The aim of this project is to create an alternative to the USEARCH tool developed by Robert C. Edgar (2010). The new tool should:

- have open source code with an appropriate open source license
- be free of charge
- have a 64-bit design that handles very large databases and much more than 4GB of memory
- be as accurate or more accurate than usearch
- be as fast or faster than usearch

We have implemented a tool called VSEARCH which supports de novo and reference based chimera detection, clustering, full-length and prefix dereplication, rereplication, reverse complementation, masking, all-vs-all pairwise global alignment, exact and global alignment searching, shuffling, subsampling and sorting. It also supports FASTQ file analysis, filtering, conversion and merging of paired-end reads.

VSEARCH stands for vectorized search, as the tool takes advantage of parallelism in the form of SIMD vectorization as well as multiple threads to perform accurate alignments at high speed. VSEARCH uses an optimal global aligner (full dynamic programming Needleman-Wunsch), in contrast to USEARCH which by default uses a heuristic seed and extend aligner. This usually results in more accurate alignments and overall improved sensitivity (recall) with VSEARCH, especially for alignments with gaps.

VSEARCH binaries are provided for x86-64 systems running GNU/Linux, macOS (version 10.7 or higher) and Windows (64-bit, version 7 or higher), as well as ppc64le systems running GNU/Linux.

VSEARCH can directly read input query and database files that are compressed using gzip and bzip2 (.gz and .bz2) if the zlib and bzip2 libraries are available.

Most of the nucleotide based commands and options in USEARCH version 7 are supported, as well as some in version 8. The same option names as in USEARCH version 7 has been used in order to make VSEARCH an almost drop-in replacement. VSEARCH does not support amino acid sequences or local alignments. These features may be added in the future.

VSEARCH 2.4.0

Table of Contents

Basic information

- **Official Website:** https://github.com/torognes/vsearch
- **License:** GNU GLP3
- **Installed on:** Apolo II and Cronos
- **Installation date:** 20/02/2017

Tested on (Requirements)

- **OS base:** CentOS (x86_64) $\geq$ 6.6 (Rocks 6.2)
- **Dependencies:**
  - g++ >= 4.9.4

Installation

1. Download the desired version of the software (Source code - tar.gz)

```
cd /home/mgomezzul/apps/vsearch/gcc
wget https://github.com/torognes/vsearch/archive/v2.4.0.tar.gz
tar -zxvf v2.4.0.tar.gz
```

2. Do the following steps fpr the compilation:

```
module load gcc/4.9.4
cd vsearch-2.4.0
./autogen.sh
./configure --prefix=/share/apps/vsearch/2.4.0/gcc/4.9.4 --build=x86_64-
↪redhat-linux
make
```

3. After compiling vshare, continue with the following steps:

```
sudo mkdir -p /share/apps/vsearch/2.4.0/gcc/4.9.4
sudo chown -R mgomezzul.apolo /share/apps/vsearch/2.4.0/gcc/4.9.4
make install
sudo chown -R root.root /share/apps/vsearch/2.4.0/gcc/4.9.4
```

Module

```
##%Module1.0#####################################################################
↪###
##
## module vsearch/2.4.0_gcc-4.9.4
```

(continues on next page)

(continued from previous page)

```
##
## /share/apps/modules/vsearch/2.4.0_gcc-4.9.4     Written by Mateo Gomez-
↪Zuluaga
##

proc ModulesHelp { } {
    puts stderr "\tvsearch/2.4.0_gcc-4.9.4 - sets the Environment for␣
↪VSEARCH 2.4.0 in \
    \n\tthe share directory /share/apps/vsearch/2.4.0/gcc/4.9.4\n"
}

module-whatis "\n\n\tSets the environment for using VSEARCH 2.4.0 \
            \n\tbuilded with GNU GCC 4.9.4\n"

# for Tcl script use only
set       topdir     /share/apps/vsearch/2.4.0/gcc/4.9.4
set       version    2.4.0
set       sys        x86_64-redhat-linux

module load gcc/4.9.4

prepend-path PATH    $topdir/bin
prepend-path MANPATH $topdir/share/man
```

Use

Slurm template

```
#!/bin/bash
 #SBATCH --partition=longjobs
 #SBATCH --nodes=1
 #SBATCH --ntasks-per-node=32
 #SBATCH --time=1:00:00
 #SBATCH --job-name=vsearch
 #SBATCH -o result_%N_%j.out
 #SBATCH -e result_%N_%j.err

 export SBATCH_EXPORT=NONE
 export OMP_NUM_THREADS=???

 module load vsearch/2.4.0

 vsearch --usearch_global queries.fsa --db database.fsa --id 0.9 --alnout␣
↪alnout.txt
```

Resources

- https://github.com/torognes/vsearch

Author

- Mateo Gómez Zuluaga

3.3.79 Wps

The WRF[1] Preprocessing System (WPS) is a set of three programs whose collective role is to prepare input to the real.exe program for real-data simulations. Each of the programs performs one stage of the preparation: geogrid defines model domains and interpolates static geographical data to the grids; ungrib extracts meteorological fields from GRIB-formatted files; and metgrid horizontally interpolates the meteorological fields extracted by ungrib to the model grids defined by geogrid. The work of vertically interpolating meteorological fields to WRF eta levels is now performed within the real.exe program, a task that was previously performed by the vinterp program in the WRF SI.

WPS 3.7.1

Table of Contents

Basic information

- **Implementation Date:** 19/20/2018
- **URL:** https://www.mmm.ucar.edu/weather-research-and-forecasting-model
- **License:** Public domain license, more info in application README
- **Installed on:** Apolo II and Cronos

Tested on (Requirements)

- **OS base:** CentOS (x86_64) $\geq$ 6.6 (Rocks 6.2)
- **Dependencies to run WPS:**
  - WRF 3.7.1
  - jasper 1.900.1

[1] Description taken from http://www2.mmm.ucar.edu/wrf/users/docs/user_guide/users_guide_chap3.html#_Function_of_Each

Installation

The following procedure is the easiest way to install Wps v3.7.1 in a cluster.

1. A WPS module must be created to add the dependencies to the environment:

```
##%Module1.0#####################################################################
↪#####
##
## module load wps/3.7.1_intel-2017_update-1
##
## /share/apps/modules/wps/3.7.1_intel-2017_update-1
## Written by Alejandro Salgado-Gómez
##

proc ModulesHelp {} {
    global version modroot
    puts stderr "Sets the environment for using wps-3.7.1\
        \nin the shared directory /share/apps/wps/3.7.1/intel-2017_update-
↪1/\
        \nbuilded with intel-2017_update-1, wrf-3.7.1, netcdf-fortran-4.4.
↪3,\
        \nnetcdf-4.4.0, hdf5-1.8.16, jasper-1.900.1\n"
}

module-whatis "(Name________) wps"
module-whatis "(Version_____) 3.7.1"
module-whatis "(Compilers___) intel-2017_update-1"
module-whatis "(System______) x86_64-redhat-linux"
module-whatis "(Libraries___) "

# for Tcl script use only
set         topdir        /share/apps/wps/3.7.1/intel-2017_update-1
set         version       3.7.1
set         sys           x86_64-redhat-linux

module load intel/2017_update-1
module load jasper/1.900.1_intel-2017_update-1
```

2. Descargar la versión deseada del software (Source code - tar.gz)[1]

```
cd /home/asalgad2/wps
wget http://www2.mmm.ucar.edu/wrf/src/WPSV3.7.1.TAR.gz
tar -xvf WPSV3.7.1.TAR.gz
```

3. After decompressing WPS, we continue with the following steps for its configuration and compilation:

```
cd WPS
```

4. The necessary environment modules are loaded:

```
module load wps/3.7.1_<compilador>
```

5. We launch the configuration

```
./configure
```

[1] **Download link:** http://www2.mmm.ucar.edu/wrf/users/download/get_sources.html

- The necessary environment modules are loaded:

1. Edit the configuration file (configure.wps) to enable the use of large files:

- Change the WRF_DIR variable to the location

```
WRF_DIR = /share/apps/wrf/3.7.1/intel-2017_update-1/
```

- Depending on the compiler version, the parameter -f90 = ifort must be removed from the following line

```
DM_FC = mpif90 -f90=ifort
```

1. Now if we can start with the Wps compilation:

```
./compile | tee wps-compilation.log
sudo mkdir -p /share/apps/wps/3.7.1/<compilador usado>
sudo cp -r * /share/apps/wps/3.7.1/<compilador usado>
```

Mode of Use

- **NOTE:** Before starting it is necessary to load the WRF module if it exists, otherwise load its dependencies.

```
module load wrf/3.7.1_gcc-5.4.0
```

Geogrid

Objective: Generate the files 'geo_em.dxx.nc' as output, where xx represents the domain number.[2]

1. Position yourself in the directory where you want to generate the output files, in our case it will be in the same directory of the geogrid.exe binary

```
cd WPS/gcc-5.4.0/
```

2. Edit the **geog_data_path** field in the **namelist.wps** file located in the WPSV3 directory to specify the location of the input files referring to the Terrain, usually located in a folder called **wrfhelp/WPS_GEOG**

```
geog_data_path = '/path/to/data/wrfhelp/WPS_GEOG'
```

3. Run Geogrid

```
./geogrid.exe
```

Ungrib

Objective: From GRIB files, generate files with an intermediate format that will then be processed by **Metgrid**[3]

1. Execute the **link_grib.csh** script specifying as the only parameter the location of the input GRIB files

```
./link_grib.csh /path/to/grib/files/
```

2. Create in the directory where the executable **ungrib.exe** is located a file called Vtable based on the desired file, either by copying it or by making a symbolic link.

[2] **Reference page:** http://www2.mmm.ucar.edu/wrf/OnLineTutorial/Basics/GEOGRID/index.html
[3] **Procedure based on official documentation:** http://www2.mmm.ucar.edu/wrf/OnLineTutorial/Basics/UNGRIB/

```
ln -sf ./ungrib/Variable_Tables/VtableDeseada Vtable
```

3. Run ungrib.exe

```
./ungrib.exe
```

Metgrid

Objective: Generate the files **met_em.d01.YYYY-MM-DD_HH:00:00.nc** and **met_em.dxx.YYYY-MM-DD_HH:00:00.nc**, which are necessary to use WRF.

1. Once the **Geogrid** and **Ungrib** files are generated in the root directory of the WPS binaries, run **Metgrid.**

```
./metgrid
```

References

Authors

- Alejandro Salgado-Gómez
- Andrés Felipe Zapata Palacio

3.3.80 WRF

The Weather Research and Forecasting (WRF)[1] Model is a next-generation mesoscale numerical weather prediction system designed for both atmospheric research and operational forecasting applications. It features two dynamical cores, a data assimilation system, and a software architecture supporting parallel computation and system extensibility. The model serves a wide range of meteorological applications across scales from tens of meters to thousands of kilometers.

WRF 3.7

Basic Information

- **Deploy date:** April 2015
- **Official Website:** https://www.mmm.ucar.edu/weather-research-and-forecasting-model
- **License:**
- **Installed on:** *Apolo II*

Installation

This entry covers the entire process performed for the installation and configuration of WRF 3.7 on a cluster.

[1] Mesoscale & Microscale Meteorology Laboratory. (n.d.). WEATHER RESEARCH AND FORECASTING MODEL. In Mesoscale & Microscale Meteorology Laboratory, from https://www.mmm.ucar.edu/weather-research-and-forecasting-model

Contents

Tested on (Requirements)

- **OS base:** CentOS (x86_64) $\geq$ 6.6
- **Compiler:** GCC $\geq$ 5.4.0.
- **Requirements:**
  - Fortran NetCDF $\geq$ 4.4.3 with GCC
  - MPICH $\geq$ 3.2 with GCC
  - zlib $\geq$ 1.2.11 with GCC
  - JasPer $\geq$ 1.900.1 with GCC

Build process

1. Get the source code

```
wget http://www2.mmm.ucar.edu/wrf/src/WRFV3.7.TAR.gz
tar xvf WRFV3.7.TAR.gz
cd WRFV3
```

2. Load the necessary modules

```
module load gcc/5.4.0
module load mpich2/3.2_gcc-5.4.0
module load jasper/1.900.1_gcc-5.4.0
module load autoconf/2.69
module load hdf5/1.8.16_gcc-5.4.0
module load netcdf-fortran/4.4.3_gcc-5.4.0
module load zlib/1.2.11_gcc-5.4.0
```

3. Execute the configuration script.

```
./configure
checking for perl5... no
checking for perl... found /usr/bin/perl (perl)
Will use NETCDF in dir: /usr/bin/netcdf
HDF5 not set in environment. Will configure WRF for use without.
PHDF5 not set in environment. Will configure WRF for use without.
Will use 'time' to report timing information
$JASPERLIB or $JASPERINC not found in environment, configuring to build without␣
↪grib2 I/O...
------------------------------------------------------------------------
Please select from among the following Linux x86_64 options:
```

(continues on next page)

(continued from previous page)

```
1. (serial)   2. (smpar)   3. (dmpar)   4. (dm+sm)   PGI (pgf90/gcc)
5. (serial)   6. (smpar)   7. (dmpar)   8. (dm+sm)   PGI (pgf90/pgcc): SGI MPT
9. (serial)  10. (smpar)  11. (dmpar)  12. (dm+sm)   PGI (pgf90/gcc): PGI␣
↪accelerator
13. (serial)  14. (smpar)  15. (dmpar)  16. (dm+sm)   INTEL (ifort/icc)
17. (dm+sm)   INTEL (ifort/icc): Xeon Phi (MIC architecture)
18. (serial)  19. (smpar)  20. (dmpar)  21. (dm+sm)   INTEL (ifort/icc): Xeon␣
↪(SNB with AVX mods)
22. (serial)  23. (smpar)  24. (dmpar)  25. (dm+sm)   INTEL (ifort/icc): SGI MPT
26. (serial)  27. (smpar)  28. (dmpar)  29. (dm+sm)   INTEL (ifort/icc): IBM POE
30. (serial)               31. (dmpar)                 PATHSCALE (pathf90/pathcc)
32. (serial)  33. (smpar)  34. (dmpar)  35. (dm+sm)   GNU (gfortran/gcc)
36. (serial)  37. (smpar)  38. (dmpar)  39. (dm+sm)   IBM (xlf90_r/cc_r)
40. (serial)  41. (smpar)  42. (dmpar)  43. (dm+sm)   PGI (ftn/gcc): Cray XC CLE
44. (serial)  45. (smpar)  46. (dmpar)  47. (dm+sm)   CRAY CCE (ftn/cc): Cray XE␣
↪and XC
48. (serial)  49. (smpar)  50. (dmpar)  51. (dm+sm)   INTEL (ftn/icc): Cray XC
52. (serial)  53. (smpar)  54. (dmpar)  55. (dm+sm)   PGI (pgf90/pgcc)
56. (serial)  57. (smpar)  58. (dmpar)  59. (dm+sm)   PGI (pgf90/gcc): -f90=pgf90
60. (serial)  61. (smpar)  62. (dmpar)  63. (dm+sm)   PGI (pgf90/pgcc): -f90=pgf90
64. (serial)  65. (smpar)  66. (dmpar)  67. (dm+sm)   INTEL (ifort/icc): HSW/BDW
68. (serial)  69. (smpar)  70. (dmpar)  71. (dm+sm)   INTEL (ifort/icc): KNL MIC

Enter selection [1-71] : 35
------------------------------------------------------------------------
Compile for nesting? (1=basic, 2=preset moves, 3=vortex following) [default 1]: 1

Configuration successful!
------------------------------------------------------------------------
testing for MPI_Comm_f2c and MPI_Comm_c2f
MPI_Comm_f2c and MPI_Comm_c2f are supported
testing for fseeko and fseeko64
fseeko64 is supported
------------------------------------------------------------------------
```

The configuration file is configuration.wrf.

4. Compile WRF, with the case you need.

```
./compile <case> | tee wrf-compilation.log
```

In `main/` you should see the following executables:

- If you compile a real case:

```
wrf.exe
real.exe
ndown.exe
tc.exe
```

- If you compile an idealized case

```
wrf.exe
ideal.exe
```

Compile WSP

The WRF Preprocessing System (WPS)[1] is a set of three programs whose collective role is to prepare input to the real.exe program for real-data simulations.

1. Download the latest version of WSP

```
wget https://github.com/wrf-model/WPS/archive/v3.7.tar.gz
tar vf 3.7.tar.gz
cd WPS-3.7
```

2. Load the correspondent modules and execute the configuration script.

```
module load jasper
./configure
```

3. Edit the configuration file `configure.wps`

```
WRF_DIR = path/to/wps

// Depends on your compiler version you should remove -f90=ifort from the␣
↪following line
DM_FC = mpif90 -f90=ifort
```

4. Compile it.

```
./compile | tee wps-compilation.log
```

Modulefile

WRF must be installed locally in the user's home because of that, there is no module file.

Usage

This section describes the method to submit jobs with the resource manager SLURM.

#. Run WRF from a SLURM bash script, for this example we will use a test case in `wrf-3.7/tests/em_real`

```
sbatch example.sh
```

The following code is an example for running WRF using SLURM:

Listing 62: `example.sh`

```
#!/bin/bash
#SBATCH --job-name=wps-wrf                          # Job name
#SBATCH --mail-type=ALL                             # Mail notification
#SBATCH --mail-user=<user>@<domain>                 # User Email
#SBATCH --error=%x-%j.err                           # Stderr (%j expands to jobId)
#SBATCH --output=%x-%j.out                          # Stdout (%j expands to jobId)
#SBATCH --ntasks=2                                  # Number of tasks (processes)
#SBATCH --nodes=1                                   # Number of nodes
```

(continues on next page)

[1] Mesoscale & Microscale Meteorology Laboratory. (n.d.). Chapter 3: WRF Preprocessing System. [online] Available at: http://www2.mmm.ucar.edu/wrf/users/docs/user_guide/users_guide_chap3.html [Accessed 28 Aug. 2019].

(continued from previous page)

```
#SBATCH --time=3:00:00                          # Walltime
#SBATCH --partition=longjobs                    # Partition

##### MODULES #####

module load gcc/5.4.0
module load mpich2/3.2_gcc-5.4.0
module load jasper/1.900.1_gcc-5.4.0
module load autoconf/2.69
module load hdf5/1.8.16_gcc-5.4.0
module load netcdf/4.4.0_gcc-5.4.0
module load netcdf-fortran/4.4.3_gcc-5.4.0
module load zlib/1.2.11_gcc-5.4.0

#### ENVIRONMENT VARIABLES ####

export OMP_NUM_THREADS=1

#### VARIABLES ####

WPS_DIR=/path/to/WPS/
WRF_DIR=/path/to/WRFV3/
WRF_HELP=/path/to/wrfhelp/

##### JOB COMMANDS ####

## WPS ##

## Delete previous wrf files ##
cd ${WPS_DIR}
rm -rf GRIBFILE.A*
rm -rf met_em.d0*
rm -rf GFS*
rm -rf geo_em.d0*

## Start ##
./geogrid.exe >& output_geogrid.log
./link_grib.csh ${WRF_HELP}/GFS030416/gfs*
ln -sf ${WPS_DIR}/ungrib/Variable_Tables/Vtable.GFS Vtable
./ungrib.exe >& output_ungrib.log
./metgrid.exe >& output_metgrid.exe

## WRF ##
cd ${WRF_DIR}/run
rm -rf rsl.*
rm -rf met_em.d0*
rm -rf wrffdda_d0*
rm -rf wrfinput_d0*
ln -sf ${WPS_DIR}/met_em* .
./real.exe

## Run MPI with SLURM ##
srun --mpi=pmi2 ./wrf.exe
```

Authors

- Manuela Carrasco Pinzón <mcarras1@eafit.edu.co>

WRF 4.1.1

Basic Information

- **Deploy date:** April 2015
- **Official Website:** https://www.mmm.ucar.edu/weather-research-and-forecasting-model
- **License:**
- **Installed on:** *Apolo II*

Installation

This entry covers the entire process performed for the installation and configuration of WRF 4.1.1 on a cluster.

WRF 4.1.1 Dependencies

Basic Information

- **Deploy date:** June 2020
- **Installed on:** *Apolo II*
- **Compiler:** Intel $\geq$ 19.0.4

Installation

This entry covers the entire process performed for the installation and configuration of all WRF 4.1.1 dependencies on a cluster with an Intel compiler. **FOLLOW THE ORDER**

1. Export the following environmental variables.

```
export CFLAGS='-O3 -xHost -ip -no-prec-div -static-intel'
export CXXFLAGS='-O3 -xHost -ip -no-prec-div -static-intel'
export FFLAGS='-O3 -xHost -ip -no-prec-div -static-intel'
export CPP='icc -E'
export CXXCPP='icpc -E'
export FC='ifort -E'
export F77='ifort -E'
```

2. Follow this guide to install zlib correctly *Zlib 1.2.11 - Intel*.
3. Follow this guide to install szip correctly *Szip 2.1.1 - Intel*.
4. Follow this guide to install netcdf-c correctly *NetCDF 4.7.4*.
5. Follow this guide to install mpich correctly *MPICH 3.3.2*.
6. Follow this guide to install netcdf-fortran correctly *NetCDF-Fortran 4.5.3*.
7. Follow this guide to install Jasper correctly *Jasper 1.900.1*.

8. After you have installed all the dependencies above correctly, you may proceed with the WRF Installation Guide.

Authors

- Tomas Navarro <tdnavarrom@eafit.edu.co>
- Satiago Alzate <salzatec1@eafit.edu.co>

Contents

WRF 4.1.1 Installation

Tested on (Requirements)

- **OS base:** CentOS (x86_64) $\geq$ 6.6
- **Compiler:** Intel $\geq$ 19.0.4
- **Requirements:**
  - zlib $\geq$ 1.2.11 with Intel 19.0.4
  - szip $\geq$ 2.1.1 with Intel 19.0.4
  - hdf5 $\geq$ 1.12 with Intel 19.0.4
  - NetCDF-C $\geq$ 4.7.4 with Intel 19.0.4
  - NetCDF-Fortran $\geq$ 4.5.3 with Intel 19.0.4
  - MPICH $\geq$ 3.3.2 with Intel 19.0.4
  - JasPer $\geq$ 1.900.1 with Intel 19.0.4

Make sure you have all the dependencies installed correctly. If you don't have them installed or think you have them installed correctly, please go to the following guide because your installation will most likely fail. *WRF 4.1.1 Dependencies*

Build process

1. Get the source code

```
wget https://github.com/wrf-model/WRF/archive/v4.1.1.tar.gz
tar xvf v4.1.1.tar.gz
cd WRF-4.1.1
```

Distributed Memory Installation

This is the installation for the distributed memory option that WRF has, please follow it exactly as it is.

1. Load the necessary modules

```
module load intel/19.0.4
module load szip/2.1.1_intel_19.0.4
module load zlib/1.2.11_intel_19.0.4
module load hdf5/1.12_intel-19.0.4
module load netcdf/4.7.4_intel-19.0.4
module load netcdf-fortran/4.5.3_intel-19.0.4
module load mpich2/3.3.2_intel-19.0.4
module load jasper/1.900.1_intel-19.0.4
```

2. Execute the configuration script, you will be asked two questions, choose `16` for the fist one (Enables distributed-memory processing with the Intel compiler), and `1` for the second one.

```
./configure
```

3. Compile WRF, with the case you need, we recommend the case to be `em_real`.

```
./compile <case> | tee wrf-compilation.log
```

In `main/` you should see the following executables:

- If you compile a real case:

```
wrf.exe
real.exe
ndown.exe
tc.exe
```

- If you compile an idealized case

```
wrf.exe
ideal.exe
```

Modulefile

```
##%Module1.0#####################################################################
↪##
##
## module load wrf/4.1.1_intel-19.0.4
##
## /share/apps/modules/wrf/4.1.1/intel-19.0.4
## Written by Tomas Navarro & Santiago Alzate
##

proc ModulesHelp {} {
     global version modroot
     puts stderr "Sets the environment for using wrf-4.1.1\
                  \nin the shared directory /share/apps/wrf/4.1.1/intel-19.0.
↪4/\
                  \nbuilded with intel-2017_update-1, netcdf-fortran-4.4.4,\
          \nnetcdf-4.5.0, hdf5-1.8.19, jasper-1.900.1"
```

(continues on next page)

(continued from previous page)

```
}

module-whatis "(Name_________) wrf"
module-whatis "(Version______) 4.1.1"
module-whatis "(Compilers____) intel-2019_update-4"
module-whatis "(System_______) x86_64-redhat-linux"
module-whatis "(Libraries____) "

# for Tcl script use only
set         topdir        /share/apps/wrf/4.1.1/intel-19.0.4
set         version       4.1.1
set         sys           x86_64-redhat-linux

module load intel/19.0.4
module load hdf5/1.12_intel-19.0.4
module load netcdf/4.7.4_intel-19.0.4
module load netcdf-fortran/4.5.3_intel-19.0.4
module load mpich2/3.3.2_intel-19.0.4
module load jasper/1.900.1_intel-19.0.4
module load wps/4.1_intel-19.0.4

setenv       WRFIO_NCD_LARGE_FILE_SUPPORT   1

setenv       WRF_DIR                        $topdir

prepend-path PATH               $topdir/main
```

Compile WPS 4.1 Distributed

The WRF Preprocessing System (WPS)[1] is a set of three programs whose collective role is to prepare input to the real.exe program for real-data simulations.

1. Download the latest version of WSP

```
wget https://github.com/wrf-model/WPS/archive/v4.1.tar.gz
tar xvf 4.1.tar.gz
cd WPS-4.1
```

2. Load the correspondent modules and execute the configuration script, use the option `19`.

```
module load wrf/4.1.1_intel-19.0.4
./configure
```

3. Edit the configuration file `configure.wps`

   In the section `WRF_LIB =` add after the following parameter `-lnetcdf` these parameters `-liomp5 -lpthread`

4. Compile it.

```
./compile | tee wps-compilation.log
```

[1] Mesoscale & Microscale Meteorology Laboratory. (n.d.). Chapter 3: WRF Preprocessing System. [online] Available at: http://www2.mmm.ucar.edu/wrf/users/docs/user_guide/users_guide_chap3.html [Accessed 28 Aug. 2019].

Modulefile

```
#%Module1.0#####################################################################
↪##
##
## module load wps/4.1_intel-19.0.4
##
## /share/apps/modules/wps/4.1/intel-19.0.4
## Written by Tomas Navarro & Santiago Alzate
##

proc ModulesHelp {} {
     global version modroot
     puts stderr "Sets the environment for using wps-4.1\
                  \nin the shared directory /share/apps/wps/4.1/intel-19.0.4/
↪\
                  \nbuilded with intel-2019_update-4, wrf-4.1.1, netcdf-
↪fortran-4.5.3,\
          \nnetcdf-4.7.4, hdf5-1.12, jasper-1.900.1\n"
}

module-whatis "(Name________) wps"
module-whatis "(Version_____) 4.1"
module-whatis "(Compilers___) intel-2019_update-4"
module-whatis "(System______) x86_64-redhat-linux"
module-whatis "(Libraries___) "

# for Tcl script use only
set         topdir        /share/apps/wps/4.1/intel-19.0.4
set         version       4.1
set         sys           x86_64-redhat-linux

module load intel/19.0.4
module load jasper/1.900.1_intel-19.0.4

prepend-path PATH               $topdir
```

Shared Memory Installation

This is the installation for the Shared-Memory/Serial option that WRF has, please follow it exactly as it is.

1. Load the necessary modules

```
module load intel/19.0.4
module load szip/2.1.1_intel_19.0.4
module load zlib/1.2.11_intel_19.0.4
module load hdf5/1.12_intel-19.0.4
module load netcdf/4.7.4_intel-19.0.4
module load netcdf-fortran/4.5.3_intel-19.0.4
module load jasper/1.900.1_intel-19.0.4
```

2. Execute the configuration script, you will be asked two questions, choose `14` for the fist one (Enables shared-memory processing with the Intel compiler), and `1` for the second one.

```
./configure
```

3. Compile WRF, with the case you need, we recommend the case to be `em_real`.

```
./compile <case> | tee wrf-compilation.log
```

In `main/` you should see the following executables:

- If you compile a real case:

```
wrf.exe
real.exe
ndown.exe
tc.exe
```

- If you compile an idealized case

```
wrf.exe
ideal.exe
```

Modulefile

```
##%Module1.0#####################################################################
↪##
##
## module load wrf/4.1.1_sm_intel-19.0.4
##
## /share/apps/modules/wrf/4.1.1/sm-intel-19.0.4
## Written by Tomas Navarro & Santiago Alzate
##

proc ModulesHelp {} {
     global version modroot
     puts stderr "Sets the environment for using wrf-4.1.1\
                  \nin the shared directory /share/apps/wrf/4.1.1/sm-intel-
↪19.0.4/\
                  \nbuilded with intel-2017_update-1, netcdf-fortran-4.4.4,\
          \nnetcdf-4.5.0, hdf5-1.8.19, jasper-1.900.1"
}

module-whatis "(Name________) wrf"
module-whatis "(Version_____) 4.1.1 Shared-Memory"
module-whatis "(Compilers___) intel-2019_update-4"
module-whatis "(System______) x86_64-redhat-linux"
module-whatis "(Libraries___) "

# for Tcl script use only
set         topdir        /share/apps/wrf/4.1.1/sm-intel-19.0.4
set         version       4.1.1
set         sys           x86_64-redhat-linux

module load intel/19.0.4
module load hdf5/1.12_intel-19.0.4
module load netcdf/4.7.4_intel-19.0.4
module load netcdf-fortran/4.5.3_intel-19.0.4
module load jasper/1.900.1_intel-19.0.4
module load wps/4.1_sm_intel-19.0.4
```

(continues on next page)

(continued from previous page)

```
setenv       WRFIO_NCD_LARGE_FILE_SUPPORT    1

setenv       WRF_DIR                         $topdir

prepend-path PATH                $topdir/main
```

Compile WPS 4.1 Serial

The WRF Preprocessing System (WPS)[1] is a set of three programs whose collective role is to prepare input to the real.exe program for real-data simulations.

1. Download the latest version of WSP

```
wget https://github.com/wrf-model/WPS/archive/v4.1.tar.gz
tar xvf 4.1.tar.gz
cd WPS-4.1
```

2. Load the correspondent modules and execute the configuration script, use the option `17`.

```
module load wrf/4.1.1_intel-19.0.4
./configure
```

3. Edit the configuration file `configure.wps`

   In the section `WRF_LIB =` add after the following parameter `-lnetcdf` these parameters `-liomp5 -lpthread`

4. Compile it.

```
./compile | tee wps-compilation.log
```

Modulefile

```
##%Module1.0#####################################################################
↪##
##
## module load wps/4.1_sm_intel-19.0.4
##
## /share/apps/modules/wps/4.1/sm-intel-19.0.4
## Written by Tomas Navarro & Santiago Alzate
##

proc ModulesHelp {} {
     global version modroot
     puts stderr "Sets the environment for using wps-4.1\
                  \nin the shared directory /share/apps/wps/4.1/sm-intel-19.
↪0.4/\
                  \nbuilded with intel-2019_update-4, wrf-4.1.1, netcdf-
↪fortran-4.5.3,\
          \nnetcdf-4.7.4, hdf5-1.12, jasper-1.900.1\n"
}

module-whatis "(Name_________) wps"
```

(continues on next page)

(continued from previous page)

```
module-whatis "(Version_____) 4.1 Serial"
module-whatis "(Compilers___) intel-2019_update-4"
module-whatis "(System______) x86_64-redhat-linux"
module-whatis "(Libraries___) "

# for Tcl script use only
set         topdir        /share/apps/wps/4.1/sm-intel-19.0.4
set         version       4.1
set         sys           x86_64-redhat-linux

module load intel/19.0.4
module load jasper/1.900.1_intel-19.0.4

prepend-path PATH               $topdir
```

References

Usage

TO DO

Authors

- Tomas Navarro <tdnavarrom@eafit.edu.co>
- Satiago Alzate <salzatec1@eafit.edu.co>

WRF 4.2

Basic Information

- **Deploy date:** May 2022
- **Official Website:** https://www.mmm.ucar.edu/weather-research-and-forecasting-model
- **License:**
- **Installed on:** *Apolo II*

Installation

This entry covers the entire process performed for the installation and configuration of WRF 4.2 on a cluster.

Contents

WRF 4.2 Installation

Tested on (Requirements)

- **OS base:** Rocky Linux (x86_64) $\geq$ 8.5
- **Compiler:** GCC 9.3.0
- **Requirements:**
  - gcc 9.3.0
  - curl 7.77.0 with gcc 9.3.0
  - MPICH 3.4.2 with gcc 9.3.0
  - zlib 1.2.12 with gcc 9.3.0
  - hdf5 1.12.0 with gcc 9.3.0
  - NetCDF-C 4.7.4 with gcc 9.3.0
  - NetCDF-Fortran 4.5.3 with gcc 9.3.0
  - JasPer 1.900.29 with gcc 9.3.0
  - libpng 1.6.37 with gcc 9.3.0
  - jpeg 9e with gcc 9.3.0

Installation

This entry covers the entire process performed for the installation and configuration of all WRF 4.2 dependencies on a cluster with an GCC compiler. **FOLLOW THE ORDER**

Distributed Memory Installation

This is the installation for the distributed memory option that WRF has, please follow it exactly as it is.

```
module load gcc/9.3.0
module load curl/7.77.0_gcc-9.3.0
module load mpich/3.4.2_gcc-9.3.0
```

Create directory for libraries

To install the libraries you must do it all in one place.

```
mkdir wrf_install_gcc
mkdir wrf_lib_gcc
```

Download libraries

```
cd wrf_install_gcc
wget https://www.zlib.net/fossils/zlib-1.2.11.tar.gz
tar -zxvf zlib-1.2.11.tar.gz
wget https://onboardcloud.dl.sourceforge.net/project/libpng/libpng16/1.6.37/
↪libpng-1.6.37.tar.gz
tar -zxvf libpng-1.6.37.tar.gz
wget https://hdf-wordpress-1.s3.amazonaws.com/wp-content/uploads/manual/HDF5/
↪HDF5_1_12_0/source/hdf5-1.12.0.tar.gz
tar -zxvf hdf5-1.12.0.tar.gz
wget https://github.com/Unidata/netcdf-c/archive/refs/tags/v4.7.4.tar.gz
tar -zxvf v4.7.4.tar.gz
https://github.com/Unidata/netcdf-fortran/archive/refs/tags/v4.5.3.tar.gz
tar -zxvf v4.5.3.tar.gz
wget https://www.ece.uvic.ca/~frodo/jasper/software/jasper-1.900.29.tar.gz
tar -zxvf jasper-1.900.29.tar.gz
wget https://www.ijg.org/files/jpegsrc.v9e.tar.gz
tar -zxvf jpegsrc.v9e.tar.gz
```

Build zlib

```
cd zlib-1.2.11
./configure --prefix=/home/blopezp/wrf_lib_gcc
make
make install
```

Exports libraries

```
export LD_LIBRARY_PATH=/home/blopezp/wrf_lib_gcc/lib:$LD_LIBRARY_PATH
export LDFLAGS=-L/home/blopezp/wrf_lib_gcc/lib
export CPPFLAGS=-I/home/blopezp/wrf_lib_gcc/include
export LD_RUN_PATH=/home/blopezp/wrf_lib_gcc/lib:$LD_RUN_PATH
export PATH=/home/blopezp/wrf_lib_gcc/bin:$PATH
```

Build libpng

```
cd libpng-1.6.37
./configure --prefix=/home/blopezp/wrf_lib_gcc
make
make install
```

Build Jpeg

```
./configure --prefix=/home/wrf/wrf_libs_intel/
make
make install
```

Build HDF5

```
cd hdf5-1.12.0
./configure --prefix=/home/blopezp/wrf_lib_gcc --with-zlib=/home/blopezp/wrf_
↪lib_gcc/ --enable-fortran
make
make install
```

Build NetCDF-C

```
cd netcdf-c-4.7.4
export HDF5=/home/blopezp/wrf_lib_gcc
./configure --prefix=/home/blopezp/wrf_lib_gcc
make
make install
```

Build NetCDF-Fortran

```
cd netcdf-fortran-4.5.3
./configure --prefix=/home/blopezp/wrf_lib_gcc
make
make install
```

Build JasPer

```
cd jasper-1.900.29
./configure --prefix=/home/wrf/wrf_libs_intel/
make
make install
```

Warning: If there is a compilation error then following fix maybe implemented (Thanks to Lena Marie Müller):

```
sed -i 's/char *optstr/const char *optstr/g' src/libjasper/jpg/jpg_dummy.c
```

Distributed Memory Installation

This is the installation for the distributed memory option that WRF has, please follow it exactly as it is.

1. Download the source code.
2. Export the necessary modules

```
module load gcc/9.3.0 curl/7.77.0_gcc-9.3.0 mpich/3.4.2_gcc-9.3.0
export LD_LIBRARY_PATH=/home/blopezp/wrf_lib_gcc/lib:$LD_LIBRARY_PATH
export LDFLAGS=-L/home/blopezp/wrf_lib_gcc/lib
export CPPFLAGS=-I/home/blopezp/wrf_lib_gcc/include
export LD_RUN_PATH=/home/blopezp/wrf_lib_gcc/lib:$LD_RUN_PATH
export NETCDF=/home/blopezp/wrf_lib_gcc
export HDF5=/home/blopezp/wrf_lib_gcc
export JASPERLIB=/home/blopezp/wrf_lib_gcc/lib
export JASPERINC=/home/blopezp/wrf_lib_gcc/include
export PATH=/home/blopezp/wrf_lib_gcc/bin:$PATH
```

3. Execute the configuration script, you will be asked two questions, choose `34` for the fist one (Enables distributed-memory processing with the GCC compiler), and `1` for the second one.

```
./configure
```

4. Remove the "time" command from the following line in the configure.wrf file.

```
FC = time $(DM_FC)
```

5. Compile WRF, with the case you need, we recommend the case to be `em_real`.

```
./compile <case> | tee wrf-compilation.log
```

In `main/` you should see the following executables:

- If you compile a real case:

```
wrf.exe
real.exe
ndown.exe
tc.exe
```

- If you compile an idealized case

```
wrf.exe
ideal.exe
```

Compile WPS 4.2 Serial

The WRF Preprocessing System (WPS)[1] is a set of three programs whose collective role is to prepare input to the real.exe program for real-data simulations.

[1] Mesoscale & Microscale Meteorology Laboratory. (n.d.). Chapter 3: WRF Preprocessing System. [online] Available at: http://www2.mmm.ucar.edu/wrf/users/docs/user_guide/users_guide_chap3.html [Accessed 28 Aug. 2019].

1. Download the latest version of WSP

```
wget https://github.com/wrf-model/WPS/archive/refs/tags/v4.2.tar.gz
tar -zxvf 4.2.tar.gz
cd WPS-4.2
```

2. Load the correspondent modules and execute the configuration script, use the option `1`.

```
export WRF_DIR=/home/blopezp/wrf_install_gcc/WRF-4.2
./configure
```

3. Edit the configuration file `configure.wps`

   In the section `WRF_LIB =` add after the following parameter `-lnetcdf` these parameters `-liomp5 -lpthread`

4. Compile it.

```
./compile | tee wps-compilation.log
```

References

Usage

TO DO

Authors

- Bryan López Parra <blopezp@eafit.edu.co>

3.4 Management Software

3.4.1 Lmod

This entry contains all relevant information about Lmod and its compilation in Apolo.

Lmod

Lmod[1] is a Lua based module system that easily handles the MODULEPATH Hierarchical problem. Environment Modules provide a convenient way to dynamically change the users' environment through modulefiles. This includes easily adding or removing directories to the PATH environment variable. Modulefiles for Library packages provide environment variables that specify where the library and header files can be found.

Lmod 8.2.7

Table of Contents

[1] Robert McLay - Lmod: A New Environment Module System. Retrieved May 15, 2020, from https://lmod.readthedocs.io/en/latest/

Basic information

- **Official Website:** https://lmod.readthedocs.io/en/latest/
- **Package's Information:** https://centos.pkgs.org/8/epel-x86_64/Lmod-8.2.7-1.el8.x86_64.rpm.html
- **License:** MIT and LGPLv2
- **Installed on:** Apolo II

Tested on (Requirements)

- **OS base:** CentOS (x86_64) $\geq$ 8
- **Dependencies to run Lmod:**
  - lua-filesystem
  - lua-posix
  - lua-term
  - coreutils

Installation

The following procedure is the easiest way to install and configure Lmod V8.2.7 in a cluster.

1. Install epel-release package and enable all the necessary repositories (epel and PowerTools for Centos 8) and install Lmod.

```
$ dnf install epel-release
$ dnf upgrade -y
$ dnf --enablerepo=PowerTools,epel install Lmod
```

2. Now, we will add our custom modulefiles path in the Lmod configuration files.

   With the `00-modulepath.sh`.

```
$ cd /etc/profile.d/
$ nano 00-modulepath.sh
```

   Go to where it says "export MODULEPATH="

   - Before the `||` `:` add the following `:/share/apps/modules/`
   - Save the file.

   Your file should look like this:

```
[ -z "$MODULEPATH" ] &&
  [ "$(readlink /etc/alternatives/modules.sh)" = "/usr/share/lmod/lmod/
↪init/profile" -o -f /etc/profile.d/z00_lmod.sh ] &&
  export MODULEPATH=/etc/modulefiles:/usr/share/modulefiles:/share/apps/
↪modules/ || :
```

Now, with the `00-modulepath.csh`.

```
$ cd /etc/profile.d/
$ nano 00-modulepath.csh
```

Go to where it says "setenv MODULEPATH"

- Add the following at the end of the line `:/share/apps/modules/`
- Save the file.

Your file should look like this:

```
if (! $?MODULEPATH && ( `readlink /etc/alternatives/modules.csh` == /usr/
↪share/lmod/lmod/init/cshrc || -f /etc/profile.d/z00_lmod.csh ) ) then
  setenv MODULEPATH /etc/modulefiles:/usr/share/modulefiles:/share/apps/
↪modules/
endif
```

Note: `/share/apps/modules/` is our custom location for modules, you need to add in the configuration files your own path.

---

3. Now, lets configure one last thing, these are the files needed to load pre-defined modules when an user logs into your cluster:

```
$ cd /etc/profile.d/
$ nano z01_StdEnv.sh
```

Inside the newly created `z01_StdEnv.sh` file, add the following:

```
if [ -z "$__Init_Default_Modules" ]; then
   export __Init_Default_Modules=1;

   ## ability to predefine elsewhere the default list
   LMOD_SYSTEM_DEFAULT_MODULES=${LMOD_SYSTEM_DEFAULT_MODULES:-"StdEnv"}
   export LMOD_SYSTEM_DEFAULT_MODULES
   module --initial_load --no_redirect restore
else
   module refresh
fi
```

```
$ cd /etc/profile.d/
$ nano z01_StdEnv.csh
```

Inside the newly created `z01_StdEnv.csh` file, add the following:

```
if ( ! $?__Init_Default_Modules )  then
  setenv __Init_Default_Modules 1
  if ( ! $?LMOD_SYSTEM_DEFAULT_MODULES ) then
    setenv LMOD_SYSTEM_DEFAULT_MODULES "StdEnv"
```

(continues on next page)

(continued from previous page)

```
  endif
  module --initial_load restore
else
  module refresh
endif
```

4. We create the StdEnv.lua file, which has the modules that we want to load from the beginning. For this example, we will use SLURM and Munge.

```
$ cd /share/apps/modules
$ nano StdEnv.lua
```

Inside the newly created `StdEnv.lua` file, add the following:

```
load("slurm","munge")
```

Note: Make sure you already have the modules created, Lmod supports both lua modulefiles and tcl modulefiles.

Lmod Module hierarchy

In this section, we will explain how your modules should be saved in order for Lmod to load them correctly.

1. First, we go to our custom modulefiles path

```
$ cd /share/apps/modules
$ ls
slurm/ munge/ gcc/ StdEnv.lua
```

As you can see, all apps have their own directory.

```
$ cd slurm/
$ ls
20.02.0
```

This is a tcl modulefile, the name is the version of the program.

```
$ cd /share/apps/modules/gcc
$ ls
5.4.0.lua
```

This is a lua modulefile, as you see, Lmod supports both pretty well.

2. Conclusion, each app needs its own directory (the directory should have the app's name). And inside the directory, the modulefiles' name should be the app's version.

Note: If you make a lua modulefile, you need to add the file extension `.lua` to the modulefile, if it's an tcl modulefile, no file extension is needed.

References

Robert McLay - Lmod: A New Environment Module System. Retrieved May 15, 2020, from https://lmod.readthedocs.io/en/latest/

Robert McLay - How to use a Software Module hierarchy. Retrieved May 15, 2020, from https://lmod.readthedocs.io/en/latest/080_hierarchy.html

Robert McLay - Converting from TCL/C Environment Modules to Lmod. Retrieved May 15, 2020, from https://lmod.readthedocs.io/en/latest/073_tmod_to_lmod.html

Robert McLay - Lua Modulefile Functions. Retrieved May 15, 2020, from https://lmod.readthedocs.io/en/latest/050_lua_modulefiles.html

Robert McLay - An Introduction to Writing Modulefiles. Retrieved May 15, 2020, from https://lmod.readthedocs.io/en/latest/015_writing_modules.html

Robert McLay - How Lmod Picks which Modulefiles to Load. Retrieved May 15, 2020, from https://lmod.readthedocs.io/en/latest/060_locating.html

Robert McLay - Providing A Standard Set Of Modules for all Users. Retrieved May 15, 2020, from https://lmod.readthedocs.io/en/latest/070_standard_modules.html

Packages Search for Linux and Unix - Lmod-8.2.7-1.el8.x86_64.rpm. Retrieved May 15, 2020, from https://centos.pkgs.org/8/epel-x86_64/Lmod-8.2.7-1.el8.x86_64.rpm.html

Authors

- Tomas David Navarro Munera <tdnavarrom@eafit.edu.co>
- Santiago Alzate Cardona <salzatec1@eafit.edu.co>

3.4.2 SLURM

This entry contains all relevant information about Apolo's current resource manager SLURM, and the available tools related to this software.

SLURM

This is a **user guide**, it does not matter if you have or not some experience using Slurm, here you will find some useful examples and commands. We try to cover our most common use cases.

Note: This guide assumes a basic comprehension about Slurm. If you don't have any idea of what Slurm is or what things it does in the HPC world, we strongly encourage reading some of the articles describe in *more information* section.

Submitting jobs

Contents

What is `sbatch`?

Slurm has a lot of options to manage all the resources of a cluster to achieve any possible combination of needs like: Number of CPUs, Number of Nodes, Memory, Time, GPUs, Licenses, etc.

The command `sbatch` is used to submit a `batch` script, making your job running in the cluster. Like this:

```
$ sbatch <batch_script>
```

A Slurm `batch` is a shell script (usually written in `bash`) where you specify all these options to Slurm, including the creation of the environment to make your job run correctly, and the set of commands to run that job.

Thus, we say that a `batch` script has **three** parts:

1. **Sbatch parameters**:

   The idea is to include all the information you think Slurm should know about your job (name, notification mail, partition, std_out, std_err, etc) and request all your computational needs, which consist at least in a number of CPUs, the computing expected duration and the amount of RAM to use.

   All these parameters must start with the comment `#SBATCH`, one per line, and need to be included at the beginning of the file, just after the shebang (`e.g. #!/bin/bash`) which should be the first line.

   The following table[3] shows important and common options, for further information see `man sbatch`.

[3] SchedMD LLC (2018). Slurm, resource management [sbatch]. Copy of manual text available at https://slurm.schedmd.com/sbatch.html. Retrieved 17:20 January 30, 2019

Table 1: Sbatch option's

Option	Description	Possible value	Mandatory
`-J, --job-name`	Job's name	Letters and numbers	no
`-t, --time`	Maximum **Walltime** of the job	Numbers with the format DD-HH:MM:SS	**yes**
`--mem`	Requested memory per **node**	size with units: 64G, 600M	no
`-n, --ntasks`	Number of tasks of the job	Number	no (default 1)
`--ntasks-per-node`	Number of tasks assigned to a node	Number	no (default 1)
`-N, --nodes`	Number of nodes requested	Number	no (default 1)
`-c, --cpus-per-task`	Number of threads per task	Number	no (default 1)
`-p, --partition`	Partition/queue where the job will be submited	longjobs, bigmem, accel and debug	no (default longjobs)
`--output`	File where the standard output will be written	Letters and numbers	no
`--error`	File where the standard error will be written	Letters and numbers	no
`--mail-type`	Notify user by email when certain event types occur to the job	NONE, ALL, BEGIN, FAIL, REQUEUE, TIME_LIMIT, TIME_LIMIT_%	no
`--mail-user`	Email to receive notification of state shanges	Valid email	no
`--exclusive`	The job allocation can not share nodes with other running jobs	Does not have values	no
`--test-only`	Validate the batch script and return an estimate of when a job would be scheduled to run	Does not have values	no
`--constraint`	Some nodes have *features* associated with them. Use this option to specify which features the nodes associated with your job must have	The name of the feature to use	no

Note: Each option must be included using `#SBATCH <option>=<value>`

Warning: Some values of the options/parameters may be specific for our clusters.

Note: About the `--mail-type` option, the value `TIME_LIMIT_%` means the reached time percent, thus, `TIME_LIMIT_90` notify reached the 90% of walltime, `TIME_LIMIT_50` at the 50%, etc.

2. **Environment creation**

Next, you should create the necessary environment to make your job run correctly. This often means include the same set of steps that you do to run your application locally on your sbatch script, things like export environment variables, create or delete files and directory structures, etc. Remember a Slurm script is a shell script.

In case you want to submit a job that uses an *application* that is installed in our *clusters* you have to `load` its module.

An application Module. is used to create the specific environment needed by your application.

The following table[1] show useful commands about modules.

Table 2: Module useful commands

Command	Functionality
`module avail`	check what software packages are available
`module whatis <module-name>`	Find out more about a software package
`module help <module-name>`	A module file may include more detailed help for the software package
`module show <module-name>`	see exactly what effect loading the module will have with
`module list`	check which modules are currently loaded in your environment
`module load <module-name>`	load a module
`module unload <module-name>`	unload a module
`module purge`	remove all loaded modules from your environment

Warning: Slurm **always** propagate the environment of the current user to the job. This could impact the behavior of the job. If you want a clean environment, add `#SBATCH --export=NONE` to your sbatch script. This option is particularly important for jobs that are submitted on one cluster and execute on a different cluster (e.g. with different paths).

3. **Job(s) steps**

   Finally, you put the command(s) that executes your application, including all the parameters. You will often see the command `srun` calling the executable instead of executing the application binary. For more information see *MPI jobs* section.

There are other options beyond using `sbatch` to submit jobs to Slurm, like `salloc` or simply using `srun`. We recommend using `sbatch`, but depending on the specific need of your application those options could be better. To know more about see: *FAQ* and *Testing my job*

Serial jobs

Serial jobs only use a process with **one** execution thread, this means one core of a CPU, given our configuration without HTT (Hyper-Threading Technology).

This kind of job does not take advantage of our computational resources but is the basic step to create more complex jobs.

In terms of Slurm, this job uses one `task (process)` and one `cpu-per-task (thread)` in one `node`. In fact, we don't need to specify any resource, the default value for those options in Slurm is `1`.

Here is a good article about the differences between `Processes` and `Threads`.

In the template below we specify `ntasks=1` to make it explicit.

[1] NYU HPC. (n.d). Slurm + tutorial - Software and Environment Modules. Retrieved 17:47, January 21, 2019 from https://wikis.nyu.edu/display/NYUHPC/Slurm+Tutorial

Listing 63: `serial-template.sh`

```
#!/bin/bash

#SBATCH --job-name=serial_test       # Job name
#SBATCH --mail-type=FAIL,END         # Mail notification
#SBATCH --mail-user=<user>@<domain>  # User Email
#SBATCH --output=slurm-serial.%j.out # Stdout (%j expands to jobId)
#SBATCH --error=slurm-serial.%j.err  # Stderr (%j expands to jobId)
#SBATCH --ntasks=1                   # Number of tasks (processes)
#SBATCH --time=01:00                 # Walltime
#SBATCH --partition=longjobs         # Partition


##### ENVIRONMENT CREATION #####



##### JOB COMMANDS ####
hostname
date
sleep 50
```

Shared Memory jobs (OpenMP)

This set up is made to create parallelism using threads on a single machine. OpenMP makes communication between threads (`-c` in Slurm) but they must be on the same machine, it does not make any kind of communication between process/threads of different physical machines.

In the below example we launched the classical "Hello world" OpenMP example[5]. It was compiled in *Cronos* using `intel compiler 18.0.1` as follow:

```
$ module load intel/18.0.1
$ icc -fopenmp omp_hello.c -o hello_omp_intel_cronos
```

We used **16** `threads`, the maximum number allowed in the Cronos' longjobs partition. In terms of Slurm, we specify **16** `cpus-per-task` and one `ntasks`.

Listing 64: `openmp-template.sh`

```
#!/bin/bash

#SBATCH --job-name=openmp_test       # Job name
#SBATCH --mail-type=FAIL,END         # Mail notification
#SBATCH --mail-user=<user>@<domain> # User Email
#SBATCH --output=slurm-omp.%j.out    # Stdout (%j expands to jobId)
#SBATCH --error=slurm-omp.%j.err     # Stderr (%j expands to jobId)
#SBATCH --time=01:00                 # Walltime
#SBATCH --partition=longjobs         # Partition
#SBATCH --ntasks=1                   # Number of tasks (processes)
#SBATCH --cpus-per-task=16           # Number of threads per task (Cronos-longjobs)
```

(continues on next page)

[5] Barney Blaise (2005) OpenMP Example - Hello World - C/C++ Version. Example was taken from https://computing.llnl.gov/tutorials/openMP/samples/C/omp_hello.c Retrieved 09:32 February 12, 2019

(continued from previous page)

```
##### ENVIRONMENT CREATION #####
module load intel/18.0.1


##### JOB COMMANDS ####
export OMP_NUM_THREADS=$SLURM_CPUS_PER_TASK

./hello_omp_intel_cronos
```

Output

```
Hello World from thread = 8
Hello World from thread = 0
Number of threads = 16
Hello World from thread = 4
Hello World from thread = 15
Hello World from thread = 5
Hello World from thread = 3
Hello World from thread = 2
Hello World from thread = 10
Hello World from thread = 9
Hello World from thread = 1
Hello World from thread = 11
Hello World from thread = 6
Hello World from thread = 12
Hello World from thread = 14
Hello World from thread = 7
Hello World from thread = 13
```

Warning: Remember the maximum number of total threads that can be running at the same time in a compute node.

- **Apolo:**
  - **Longjobs queue:** 32
  - **Accel queue:** 32
  - **Bigmem queue:** 24
  - **Debug queue:** 2
- **Cronos:**
  - **Longjobs queue:** 16

Otherwise, your job will overpass the maximum multiprocessing grade and this is going to cause a drastic decrease in the performance of your application. To know more about see: *FAQ*

As extra information, our setup does not use HTT (Hyper-Threading Technology).

Note: We highly recommend using the Slurm variable `$SLURM_CPUS_PER_TASK` to specify the number of threads that OpenMP is going to work with. Most of the applications use the variable `OMP_NUM_THREADS` to defined it.

MPI jobs

MPI jobs are able to launch multiple processes on multiple nodes. There is a lot of possible workflows using MPI, here we are going to explain a basic one. Based on this example and modifying its parameters, you can find the configuration for your specific need.

The example was compiled in *Cronos* using `impi` as follow:

```
$ module load impi
$ impicc hello_world_mpi.c -o mpi_hello_world_apolo
```

We submited the classic "Hello world" MPI example[6] using 5 processes (`--ntasks=5`), each one on a different machine (`--ntasks-per-node=1`). Just to be clear, we used 5 machines and 1 CPU per each, leaving the other CPUs (15, in this specific case) free to be allocated by Slurm to other jobs.

Listing 65: `mpi-template.sh`

```
#!/bin/bash

#SBATCH --job-name=mpi_test          # Job name
#SBATCH --mail-type=FAIL,END         # Mail notification
#SBATCH --mail-user=<user>@<domain> # User Email
#SBATCH --output=slurm-mpi.%j.out    # Stdout (%j expands to jobId)
#SBATCH --error=slurm-mpi.%j.err     # Stderr (%j expands to jobId)
#SBATCH --time=01:00                 # Walltime
#SBATCH --partition=longjobs         # Partition
#SBATCH --ntasks=5                   # Number of tasks (processes)
#SBATCH --ntasks-per-node=1          # Number of task per node (machine)


##### ENVIRONMENT CREATION #####
module load impi


##### JOB COMMANDS ####
srun --mpi=pmi2 ./mpi_hello_world_apolo
```

Note: The use of `srun` is mandatory here. It creates the necessary environment to launch the MPI processes. There you can also specify other parameters. See `man srun` to more information.

Also, the use of `--mpi=pmi2` is mandatory, it tells MPI to use the pmi2 Slurm's plugin. This could change when you are using a different implementation of MPI (e.g MVAPICH, OpenMPI) but we strongly encourage our users to specify it.

Output

```
HELLO_MPI - Master process:
  C/MPI version
  An MPI example program.

  Process 3 says 'Hello, world!'
  The number of processes is 5.
```

(continues on next page)

[6] Burkardt John (2008) Using MPI: Portable Parallel Programming with the Message-Passing Interface. Example was taken from https://people.sc.fsu.edu/~jburkardt/c_src/heat_mpi/heat_mpi.c Retrived 09:38 February 12, 2019

(continued from previous page)

```
  Process 0 says 'Hello, world!'
  Elapsed wall clock time = 0.000019 seconds.
  Process 1 says 'Hello, world!'
  Process 4 says 'Hello, world!'
  Process 2 says 'Hello, world!'

HELLO_MPI - Master process:
  Normal end of execution: 'Goodbye, world!'

30 January 2019 09:29:56 AM
```

Warning: As you can see in that example, we do not specify `-N` or `--nodes` to submit the job in 5 different machines. You can let Slurm decides how many machines your job needs.

Try to think in terms of "tasks" rather than "nodes".

This table shows some other useful cases[2]:

Table 3: MPI jobs table

You want	You ask
N CPUs	`--ntasks=N`
N CPUs spread across distinct nodes	`--ntasks=N --nodes=N`
N CPUs spread across distinct nodes and nobody else around	`--ntasks=N --nodes=N --exclusive`
N CPUs spread across N/2 nodes	`--ntasks=N --ntasks-per-node=2`
N CPUs on the same node	`--ntasks=N --ntasks-per-node=N`

Array jobs

Also called Embarrassingly-Parallel, this set up is commonly used by users that do not have a native parallel application, so they run **multiple parallel instances** of their application changing its `input`. Each instance is independent and does not have any kind of communication with others.

To do this, we specify an array using the `sbatch` parameter `--array`, multiple values may be specified using a comma-separated list and/or a range of values with a "-" separator (e.g `--array=1,3,5-10` or `--array=1,2,3`). This will be the values that the variable `SLURM_ARRAY_TASK_ID` is going to take in each `array-job`.

This `input` usually refers to these cases:

1. **File input**

   You have **multiple files/directories** to **process**.

   In the below example/template we made a "parallel copy" of the files contained in `test` directory using the `cp` command.

```
./test/
├── file1.txt
├── file2.txt
├── file3.txt
```

(continues on next page)

[2] UCLouvai - University of Leuven (n.d). Slurm Workload Manager - Slide 57. Retrieved 11:33 January 25, 2019 from http://www.cism.ucl.ac.be/Services/Formations/slurm/2016/slurm.pdf

(continued from previous page)

```
├── file4.txt
└── file5.txt
```

We used one process (called `task` in Slurm) per each `array-job`. The array goes from 0 to 4, so there were 5 processes copying the 5 files contained in the `test` directory.

Listing 66: `array-file-input-template.sh`

```
#!/bin/bash

#SBATCH --job-name=array_file_test        # Job name
#SBATCH --mail-type=FAIL,END              # Mail notification
#SBATCH --mail-user=<user>@<domain>       # User Email
#SBATCH --output=slurm-arrayJob%A_%a.out # Stdout (%a expands to stepid, %A to␣
↪jobid )
#SBATCH --error=slurm-array%J.err         # Stderr (%J expands to GlobalJobid)
#SBATCH --ntasks=1                        # Number of tasks (processes) for each␣
↪array-job
#SBATCH --time=01:00                      # Walltime for each array-job
#SBATCH --partition=debug                 # Partition

#SBATCH --array=0-4     # Array index


##### ENVIRONMENT CREATION #####


##### JOB COMMANDS ####

# Array of files
files=(./test/*)

# Work based on the SLURM_ARRAY_TASK_ID
srun cp ${files[$SLURM_ARRAY_TASK_ID]} copy_$SLURM_ARRAY_TASK_ID
```

Thus, the generated file `copy_0` is the copy of the file `test/file1.txt` and the file `copy_1` is the copy of the file `test2.txt` and so on. Each one was done by a different Slurm process in parallel.

Warning: Except to `--array`, **ALL** other `#SBATCH` options specified in the submitting Slurm script are used to configure **EACH** `job-array`, including **ntasks**, **ntasks-per-node**, **time**, **mem**, etc.

2. **Parameters input**

   You have **multiple parameters** to **process**.

   Similarly to the last example, we created an array with some values that we wanted to use as parameters of the application. We used one process (`task`) per `array-job`. We had 4 parameters (`0.05 100 999 1295.5`) to process and 4 `array-jobs`.

   Force Slurm to run array-jobs in different nodes

   To give another feature to this example, we used `1` node for each `array-job`, so, even knowing that one node can run up to 16 processes (in the case of *Cronos*) and the 4 `array-jobs` could be assigned to `1` node, we forced Slurm to use `4` nodes.

   To get this we use the parameter `--exclusive`, thus, for each `job-array` Slurm will care about not to have other Slurm-job in the same node, even other of your `job-array`.

Note: Just to be clear, the use of `--exclusive` as a **SBATCH** parameter tells Slurm that the job allocation cannot share nodes with other running jobs[4] . However, it has a slightly **different** meaning when you use it as a parameter of a job-step (each separate srun execution inside a **SBATCH** script, e.g `srun --exclusive $COMMAND`). For further information see `man srun`.

Listing 67: `array-params-template.sh`

```
#!/bin/bash

#SBATCH --job-name=array_params_test     # Job name
#SBATCH --mail-type=FAIL,END             # Mail notification
#SBATCH --mail-user=<user>@<domain>      # User Email
#SBATCH --output=slurm-arrayJob%A_%a.out # Stdout (%a expands to stepid, %A␣
↪to jobid )
#SBATCH --error=slurm-array%J.err        # Stderr (%J expands to GlobalJobid)
#SBATCH --ntasks=1                       # Number of tasks (processes) for␣
↪each array-job
#SBATCH --time=01:00                     # Walltime for each array-job
#SBATCH --partition=debug                # Partition

#SBATCH --array=0-3    # Array index
#SBATCH --exclusive    # Force slurm to use 4 different nodes

##### ENVIRONMENT CREATION #####


##### JOB COMMANDS ####

# Array of params
params=(0.05 100 999 1295.5)

# Work based on the SLURM_ARRAY_TASK_ID
srun echo ${params[$SLURM_ARRAY_TASK_ID]}
```

Remember that the **main idea** behind using Array jobs in Slurm is based on the use of the variable `SLURM_ARRAY_TASK_ID`.

Note: The parameter `ntasks` specify the number of processes that **EACH** `array-job` is going to use. So if you want to use more, you just can specify it. This idea also applies to all other `sbatch` parameters.

Note: You can also limit the number of simultaneously running tasks from the job array using a `%` separator. For example `--array=0-15%4` will limit the number of simultaneously running tasks from this job array to 4.

Slurm's environment variables

In the above examples, we often used the output of the environment variables provided by Slurm. Here you have a table[3] with the most common variables.

[4] SchedMD LLC (2018). Slurm, resource management [srun]. Copy of manual text available at https://slurm.schedmd.com/srun.html. Retrieved 12:20 January 31, 2019

Table 4: Output environment variables

Variable	Functionality
SLURM_JOB_ID	job Id
SLURM_ARRAY_TASK_ID	Index of the slurm array
SLURM_CPUS_PER_TASK	Same as --cpus-per-task
SLURM_NTASKS	Same as -n, --ntasks
SLURM_JOB_NUM_NODES	Number of nodes allocated to job
SLURM_SUBMIT_DIR	The directory from which sbatch was invoked

Slurm's file-patterns

sbatch allows filename patterns, this could be useful to name std_err and std_out files. Here you have a table[3] with some of them.

Table 5: Slurm's file-patterns

File-patern	Expands to
%A	Job array's master job allocation number
%a	Job array ID (index) number
%j	jobid of the running job
%x	Job name
%N	short hostname. This will create a separate IO file per node

Note: If you need to separate the output of a job per each node requested, %N is specially useful, for example in array-jobs.

For instance, if you use #SBATCH --output=job-%A.%a in an array-job the output files will be something like job-1234.1, job-1234.2 , job-1234.3; where: 1234 refers to the job array's master job allocation number and 1 , 2 and 3 refers to the id of each job-array.

Constraining Features on a job

In Apolo II, one can specify what type of CPU instruction set to use. One can choose between AVX2 and AVX512. These *features* can be specify using the SBATCH option --constraint=<list> where <list> is the **features** to constrain. For example, --constraint="AVX2" will allocate only nodes that have AVX2 in their instruction set. --constraint="AVX2|AVX512" will allocate only nodes that have either AVX512 or AVX2.

One can also have a job requiring some nodes to have AVX2 and some others using AVX512. For this one would use operators **'&'** and **'*'**. The ampersand works as a **'and'** operator, and the **'*'** is used to specify the number of nodes that must comply a single feature. For example, --constraint="[AVX2*2&AVX512*3]" is asking for two nodes with AVX2 and three with AVX512. The squared brackets are mandatory.

References

Testing my job

Here you will find some interesting ways to test your jobs before submitting it to compute a real problem with a big Walltime. It is always a good idea to use some of these tips in order to prevent the failure of your jobs.

Contents

Debug partition

The debug partition is a useful queue created to test your Slurm job script, it does not have any performance capabilities but its nodes have the same environment as the longjobs partition.

To use this partition you only need to specify it in your batch script like this:

```
#!/bin/bash
#SBATCH --partition=debug
# Other sbatch parameters
```

Note: Quick aspects about the debug partition:

- **Apolo:**
  - Number of Nodes: 2
  - Number of CPUs per node: 2
  - Memory per node: 2GB
- **Cronos:** Not deployed yet.

For more information, see *getting cluster information* section

Warning: Debug partition has the same environment of longjobs, so if you want to test a job that will be executed in a different queue (e.g Accel or Bigmem) it does not guarantee a successful execution.

`salloc` command

This command is used to allocate resources (e.g Nodes, tasks, mem, etc.) and, if available, interactively run jobs steps under that allocation using a sub-shell created by Slurm[1] .

This sub-shell will let you write those commands that you use to execute your application on the allocated **compute node(s)** of your job, as they were executed on a sbatch script. Thus, every command will give you immediate feedback of its `std_out`, `std_err` and `EXIT_CODE` in your terminal. We recommend following these parts: **Environment creation** and **Job(s) steps**. See *Submitting jobs* to know more information.

This example shows the submission of HPL, a well-known implementation of the High Performance Computing Linpack Benchmark, it uses `MPI`.

[1] SchedMD LLC (2018). Slurm, resource management [salloc]. Copy of manual text available at https://slurm.schedmd.com/salloc.html. Retrieved 18:11 February 11, 2019

Following the same parts showed in the *Submitting jobs* we will have:

- **Environment creation**: We need to load two modules: `impi` and `mkl`
- **Job step**: HPL uses one job-step, the execution of `xhpl`

In this case, we executed HPL using `16` processes (`--ntaks=16` in Slurm). You need to specify the **Walltime**, it refers to the time that the sub-shell will be alive.

First, we ran `salloc` specifying the number of processes and the period of time, allocating the resources and, if available, it will create a sub-shell.

```
*[cronos: hpl]$ salloc -n16 -t1:00:00
salloc: Granted job allocation 72364
```

Next, we ran the commands to create the environment

```
*[cronos: hpl]$ module load impi/18.0.2
*[cronos: hpl]$ module load mkl/18.0.2
```

Then we executed the job-step. You can notice the *std_out* was immediately written on the terminal

```
*[cronos: hpl]$ srun --mpi=pmi2 ./xhpl
================================================================================
HPLinpack 2.3  --  High-Performance Linpack benchmark  --   December 2, 2018
Written by A. Petitet and R. Clint Whaley,  Innovative Computing Laboratory, UTK
Modified by Piotr Luszczek, Innovative Computing Laboratory, UTK
Modified by Julien Langou, University of Colorado Denver
================================================================================

An explanation of the input/output parameters follows:
T/V    : Wall time / encoded variant.
N      : The order of the coefficient matrix A.
NB     : The partitioning blocking factor.
P      : The number of process rows.
Q      : The number of process columns.
Time   : Time in seconds to solve the linear system.
```

Warning: If you are using `salloc`, the use of `srun` in every **job-step** is mandatory. It does not matter the type of job you are computing (even a Serial or OpenMP job). This is because, compared with the jobs submitted using sbatch, salloc's jobs does not have a `BatchHost` by default, so all the commands are executed in the **master node**, instead of one of the allocated compute nodes. You need to use `srun` to **explicitly** run that command in the allocated node(s)

This example illustrates the problem of **not** using `srun` in a job-step

```
*[cronos: ~]$ salloc -n2 -t1:00
salloc: Granted job allocation 72363
*[cronos: ~]$ hostname
cronos.eafit.edu.co
```

and using `srun`, then:

```
*[cronos: ~]$ srun hostname
compute-1-0.local
compute-1-0.local
```

`--test-only` parameter

This parameter will validate the batch script and return an estimate of when the job would be scheduled to run[2], given the current state of the queue and other arguments specified (e.g `Nodes`, `tasks`, `cores`) on the job requirements. **THE JOB is NOT actually submitted**.

As an example, we use this sbatch script, as you can notice, we add the line `#SBATCH --test-only`

```
#!/bin/bash

#SBATCH --job-name=test_only          # Job name
#SBATCH --ntasks=80                   # Number of tasks (processes)
#SBATCH --ntasks-per-node=16          # Maximum possible value in Cronos
#SBATCH --time=01:00                  # Walltime
#SBATCH --partition=longjobs          # Partition

#SBATCH --test-only                   # Test the job


##### ENVIRONMENT CREATION #####



##### JOB COMMANDS ####
srun not_a_executable
```

output:

```
*[cronos: serial]$ sbatch test-only-slurm.sh
sbatch: Job 72371 to start at 2019-02-14T03:57:26 using 80 processors
 on nodes compute-1-[1-5] in partition longjobs
```

References

Getting information about jobs

Contents

[2] SchedMD LLC (2018). Slurm, resource management [sbatch]. Copy of manual text available at https://slurm.schedmd.com/sbatch.html. Retrieved 17:20 January 30, 2019

Getting cluster(s) state

In Slurm, nodes have different states[2], this tells if a job can or not be allocated.

Table 6: Slurm node's states

State	Description
DOWN	The node is unavailable for use
ALLOCATED	The node has been allocated to one or more jobs.
IDLE	The node is not allocated to any jobs and is available for use
MIXED	The node has some of its CPUs **ALLOCATED** while others are **IDLE**
DRAINED	The node is unavailable for use per **system administrator** request
MAINT	The node is under maintenance by **system administrator**

The simplest way to get information about the state of our clusters is using the commands: `sinfo` and `squeue`. Here we list some useful examples[1][2][3] .

- View information about nodes and partitions and a longer version (`-N`)

```
$ sinfo
$ sinfo -N
```

- Show nodes that are in a specific state.

```
$ sinfo -t idle
$ sinfo -t mix
$ sinfo -t alloc
```

- Report node state reason (if exists)

```
$ sinfo -R
```

- Show queued jobs and long version

```
$ squeue
$ squeue -l
```

Note: `squeue` also includes running jobs.

- Show queued jobs by a specific user. Most of the cases you will need to get information about your jobs, using the variable `$USER` could be useful.

[2] SchedMD LLC (2018). Slurm, resource management [sinfo]. Copy of manual text available at https://slurm.schedmd.com/sinfo.html. Retrieved 14:24 January 31, 2019

[1] University of Luxembourg (UL) HPC Team (2018). UL HPC Tutorial: Advanced scheduling with SLURM. Retrieved 16:45 January 28, 2019 from https://ulhpc-tutorials.readthedocs.io/en/latest/scheduling/advanced/

[3] SchedMD LLC (2018). Slurm, resource management [squeue]. Copy of manual text available at https://slurm.schedmd.com/squeue.html. Retrieved 12:30 February 1, 2019

```
$ squeue -u $USER
$ squeue -u pepito77
```

- Show queued jobs of a specific partition/queue.

```
$ squeue -p debug
$ squeue -p bigmem
$ squeue -p accel
```

- Show queued jobs that are in a specific state. To know more about the job's state see: *What's going on with my job? Getting information about submitted jobs*

```
$ squeue -t PD
$ squeue -t R
$ squeue -t F
$ squeue -t PR
```

- Show detailed information about the node(s)

```
$ scontrol show node compute-1-25
$ scontrol show node compute-0-5
$ scontrol show node debug-0-0
```

Note: If you need further information, you can always check the command's manual `man squeue`, `man sinfo`, etc.

What's going on with my job? Getting information about submitted jobs

Once your job is queued in a specific partition you may want to know its state. There some of the Slurm's job states[3].

Table 7: Job's states

State	Description
CANCELLED (CA)	Job was explicitly cancelled by the user or system administrator
COMPLETED (CD)	Job has terminated all processes on all nodes with an exit code of zero
PENDING (PD)	Job is awaiting resource allocation, there is some different reasons
RUNNING (R)	Job currently has an allocation
STOPPED (ST)	Job has an allocation, but execution has been stopped with SIGSTOP signal. CPUS have been retained by this job
SUSPENDED (S)	Job has an allocation, but execution has been suspended and CPUs have been released for other jobs

You can check the expected start time of a job(s) base on the actual queue state:

```
$ squeue --start --job 1234
$ squeue --start -u $USER
```

You can also check the reason why your job is waiting, usually is displayed by default in the command `squeue`. You can also change the output format, thus, display the field `reason (%R)` more clearly.

```
$ squeue -u $USER --format="%j %name %U %R"
$ squeue --jobid 1234 --format="%j %name %U %R"
```

Note: Not only pending jobs set the reason field, also failed jobs set it, showing its failure message.

Note: You can also use `sprio` in order to know the priority of your job(s). For further information see `man sprio`

In the following table[3] we describe the most common reasons:

Table 8: Job's reasons

Reason	Description
QOSMaxCpuPerUserLimit	User's allocated jobs are already using the maximum number of CPUs allowed per user. Once the number of allocated CPUs decrease, the job(s) will start
Priority	One or more higher priority jobs exist for this queue, usually jobs are allocated with a First In First Out set up, for further information see `man sprio`
Resources	The job is waiting for resources (CPUs, Memory, nodes, etc). to become available
TimeLimit	The job exhausted its time limit
BadConstraints	The job's constraints can not be satisfied

Warning: Related with `QOSMaxCpuPerUserLimit` Slurm's reason, the maximum number of allocated resources at the same time (in specific Memory and CPUs) per user differ between clusters:

- **Apolo: CPUs:** 96 **Memory:** 192G
- **Cronos: CPUs:** 96 **Memory:** 384G

It is important to note that those are policies defined by Apolo - Centro de Computación Científica.

Another useful command to show information about recent jobs is:

```
$ scontrol show job 1234
```

There is an example of its output from *Apolo II*.

```
JobId=1234 JobName=CuteJob
   UserId=user1(11) GroupId=user1(34) MCS_label=N/A
   Priority=2000 Nice=0 Account=ddp QOS=normal
   JobState=RUNNING Reason=None Dependency=(null)
   Requeue=1 Restarts=0 BatchFlag=1 Reboot=0 ExitCode=0:0
   RunTime=2-22:33:43 TimeLimit=4-03:00:00 TimeMin=N/A
   SubmitTime=2019-01-29T03:46:05 EligibleTime=2019-01-29T03:46:05
   AccrueTime=2019-01-29T03:46:05
   StartTime=2019-01-29T15:47:12 EndTime=2019-02-02T18:47:12 Deadline=N/A
   PreemptTime=None SuspendTime=None SecsPreSuspend=0
   LastSchedEval=2019-01-29T15:47:12
   Partition=accel AllocNode:Sid=apolo:2222
   ReqNodeList=(null) ExcNodeList=(null)
   NodeList=compute-0-5
   BatchHost=compute-0-5
   NumNodes=1 NumCPUs=32 NumTasks=32 CPUs/Task=1 ReqB:S:C:T=0:0:*:*
   TRES=cpu=32,mem=60000M,node=1,billing=32
   Socks/Node=* NtasksPerN:B:S:C=0:0:*:* CoreSpec=*
   MinCPUsNode=1 MinMemoryCPU=1875M MinTmpDiskNode=0
   Features=(null) DelayBoot=00:00:00
```

(continues on next page)

(continued from previous page)

```
   OverSubscribe=OK Contiguous=0 Licenses=(null) Network=(null)
   Command=/home/user1/cutejob/slurm.sh
   WorkDir=/home/user1/cutejob
   StdErr=/home/user1/cutejob/cutejob.1234.err
   StdIn=/dev/null
   StdOut=/home/user1/cutejob/cutejob.1234.out
   Power=
```

Note: We also recommend to log in (using `ssh`) into the respective compute node(s) of your job and run `htop` in order to see if your process(es) are actually running in the way you would expect and check if the compute's `CPU Load` is the optimum. To know more about see: *FAQ*

Canceling a job

Once your job is submitted, you can do some operations in order to change its state. Here we list some useful examples[14] .

- Cancel job 1234

```
$ scancel 1234
```

- Cancel only array ID 9 of job array 1234

```
$ scancel 1234_9
```

- Cancel all my jobs (without taking care of its state)

```
$ scancel -u $USER
```

- Cancel my waiting (`pending` state) jobs.

```
$ scancel -u $USER -t pending
```

- Cancel the jobs queue on a given partition (queue)

```
$ scancel -p longjobs
```

- Cancel one or more jobs by name

```
$ scancel --name MyJobName
```

- Pause the job 1234

```
$ scontrol hold 1234
```

- Resume the job 1234

```
$ scontrol resume 1234
```

- Cancel and restart the job 1234

[4] SchedMD LLC (2018). Slurm, resource management [scancel]. Copy of manual text available at https://slurm.schedmd.com/sinfo.html. Retrieved 15:47 January 31, 2019

```
$ scontrol requeue 1234
```

What happened with my job? Getting information about finished jobs

Here we are going to explain how to get information about completed jobs (that are no longer in the queue). Those commands use the Slurm database to get the information.

Note: By default, these commands only search jobs associated with the cluster you are log in, however, for example, if you want to search a job that was executed on *Cronos* while you are in a session in *Apolo II*, you can do it using the argument `-M slurm-cronos`. Other possible options are `-M slurm-apolo` and `-M all`

- `sacct`: is used to get general accounting data for all jobs and job steps in the Slurm[5].
  - In case you remember the `jobid` you can use

    ```
    $ sacct -j1234
    ```

  - Get information about today's jobs submitted by a user (or users)

    ```
    $ sacct -S$(date +'%m/%d/%y') -u $USER
    ```

  - Get information about jobs submitted by a user (or users) 1 week ago

    ```
    $ sacct -S$(date +'%m/%d/%y' --date="1 week ago") -u $USER
    ```

  - Get information about the job(s) by its name(s)

    ```
    $ sacct -S$(date +'%m/%d/%y') --name job_name
    ```

Note: `-S` argument is to select eligible jobs in any state after the specified time. It is mandatory to search jobs in case that a `jobid` was not specified. It supports multiple date formats, see `man sacct` to know more about.

References

Fairshare

Contents

[5] SchedMD LLC (2018). Slurm, resource management [sacct]. Copy of manual text available at https://slurm.schedmd.com/sacct.html. Retrieved 8:44 February 4, 2019

Tested

- **Installation date:** 30/04/2021
- **Apolo version:** Apolo Test Environment

Fair-share Factor

The fair-share is a penalty policy not implemented in slurm by default, its component to a job's priority influences the order in which a user's queued jobs are scheduled to run based on the portion of the computing resources they have been allocated and the resources their jobs have already consumed. The fair-share factor does not involve a fixed allotment, whereby a user's access to a machine is cut off once that allotment is reached.

Instead, the fair-share factor serves to prioritize queued jobs such that those jobs charging accounts that are under-serviced are scheduled first, while jobs charging accounts that are over-serviced are scheduled when the machine would otherwise go idle.

Slurm's fair-share factor is a floating point number between 0.0 and 1.0 that reflects the shares of a computing resource that a user has been allocated and the amount of computing resources the user's jobs have consumed. The higher the value, the higher is the placement in the queue of jobs waiting to be scheduled.

Note: Computing the fair-share factor requires the installation and operation of the Slurm Accounting Database to provide the assigned shares and the consumed, computing resources described below.

Regenerate the initram on CentOS 8

In CenOS 8 you need to use the `dracut` tool to manage the initram. In order to regenerate and install the `initram`:

```
dracut --force # This will force the generation and installation when an initram␣
↪already exists
```

Warning: Apolo is currently supported by CentOS 6.6, which makes it impossible to use initram.

Plugins

Multifactor Priority Plugin

The Multi-factor Job Priority plugin provides a very versatile facility for ordering the queue of jobs waiting to be scheduled.

Note: The multi-factor priority plugin is already installed in the Apolo cluster.

Slurm.conf

The slurm.conf file describes general Slurm configuration information. Here we may put the plugins we want to use to manage our cluster resources. In our case we are going to use the `PriorityWeightFairshare=2000` in the section `PriorityType=priority/multifactor`. Equal to 2000 since in the previous configuration of other factors such as `PriorityWeightPartition` is equal to 2000, in general all factors should be equal to the same amount.

Note: Fairshare is only supported on CentOS 7 onwards.

Warning: slurmdbd must be restarted when making changes to slurm.conf.

¿How do I configure Slurm Fair Shares?

Slurm Fair Sharing can be configured using the sacctmgr tool. The following example illustrates how 50% Fair Sharing between two Users, User1 and User2, can be configured.

1. **The Fairshare of the parent account must be modified.**

```
sacctmgr modify where account=science set fairshare=50
```

2. **The Fairshare of the account must be modified.**

```
sacctmgr modify where account=chemistry set fairshare=30

sacctmgr modify where account=physics set fairshare=20
```

3. **Enroll two users into the physics accounts with 0.5 of the resources assigned to each user:**

```
sacctmgr modify where user name=User1 cluster=apolito account=physics␣
↪fairshare=10

sacctmgr modify where user name=User2 cluster=apolito account=physics␣
↪fairshare=10
```

4. **The fair share configurations can be reviewed for a particular cluster as follows:**

```
sacctmgr list associations cluster=apolito format=Account,Cluster,User,
↪Fairshare
```

Note: In this case the fairshare has been configured with small values, since it has been tested in Apolo test environment, which has few resources and is used for testing before moving to production, also has the CentOS 8 operating system.

More information

- Quick Start User Guide.
- Slurm + tutorial - Software and Environment Module
- SLURM: Simple Linux Utility for Resource Management.

Authors

Bryan López Parra <blopezp@eafit.edu.co>

FAQ

Contents

What should be the CPU Load on my node?

The CPU Load is the measure of the amount of computational work that a compute node has been performing. It is always a good idea to keep monitoring this measure while you are computing in order to know how well your job is running and if it agrees with the given parameters.

To do this, we are going to log in to the compute node (or nodes) where your job is running on, then we are going to use htop. For example, `compute-0-0`:

```
$ ssh compute-0-0
$ htop
```

Warning: You must have a running job on the compute node that you will log in to, otherwise, you will not be able to do it. In case you have an active session on a compute node and all of your jobs have finished their execution, Slurm is going to kill all the user's remaining processes on that node, including your current session.

On the htop. screen you will find all the running processes of that node.

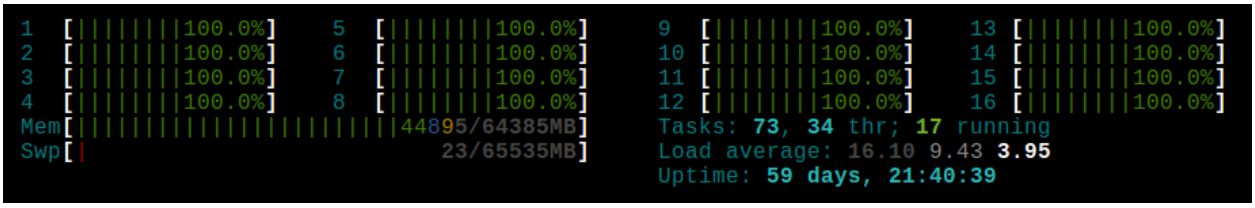
The first thing you should check is if there's `n` number of running process plus one (the htop process).

Now, check the CPU load, htop display 3 different values: the Average of 1 minute, 5 minutes and 15 minutes. As a general rule, you should expect that the values will be the number of running cores that you asked for in your sbatch script, in most of the cases is the same number of tasks `n`.

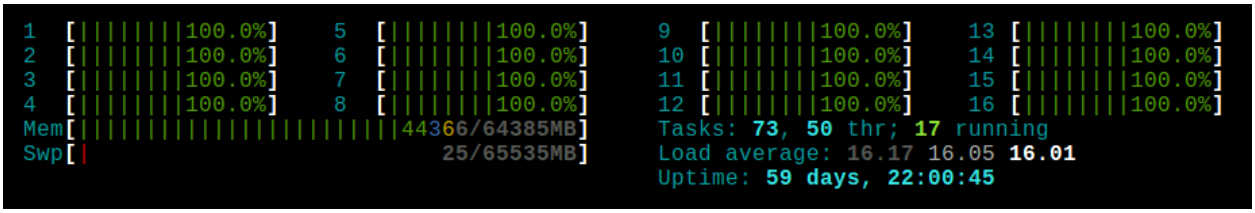
In this example, we ran a job in *Cronos* with 16 processes. Here, are the sbatch parameters.

```
#SBATCH --ntasks=16
#SBATCH --time=30:00
```

And here, is the screen of htop corresponding to the compute node that was running that job after 3 minutes.



and here is the screen of htop on the same node after 20 minutes.



As you can see, there is **17** running process (16 tasks + htop) and the load average is close to `16.00`. All the cores (`16`) are in use.

Do not mind if you find that your `CPUs Load` is less than expected, it does not necessarily mean that your job is not running in an optimal way. It could be due that your application can not reach that CPU load, for example, it frequently does `I/O` processing.

Warning: However, if you have a considerably high `CPUs Load` (e.g `40.00`, `50.00`, `300.00`), no matter the case, this is **not a good thing** and it will affect substantially the performance of your job. You are **OVER LOADING** the node.

In which cases should I use `srun`?

There are three different ways in which jobs can be run by users in Slurm, using: `srun`, `salloc` and `sbatch`.

`srun` is used to create an interactive job, this means the job is going to be linked with the current `tty` and `session` of the user. If you disconnect it, you will lose control over the job, or it might be killed. In case the resources that you specified are not available, you will need to wait, keeping the session alive. The `std_out` and `std_err` will be written in your terminal.

`salloc` is used to create jobs in allocated mode, the user is able to allocate resources and, if available, interactively run jobs steps under that allocation using a sub-shell created by Slurm. For further information see `man salloc`.

`sbatch` is used to submit a job for later execution using a script. We have documentation about in the *submit* section.

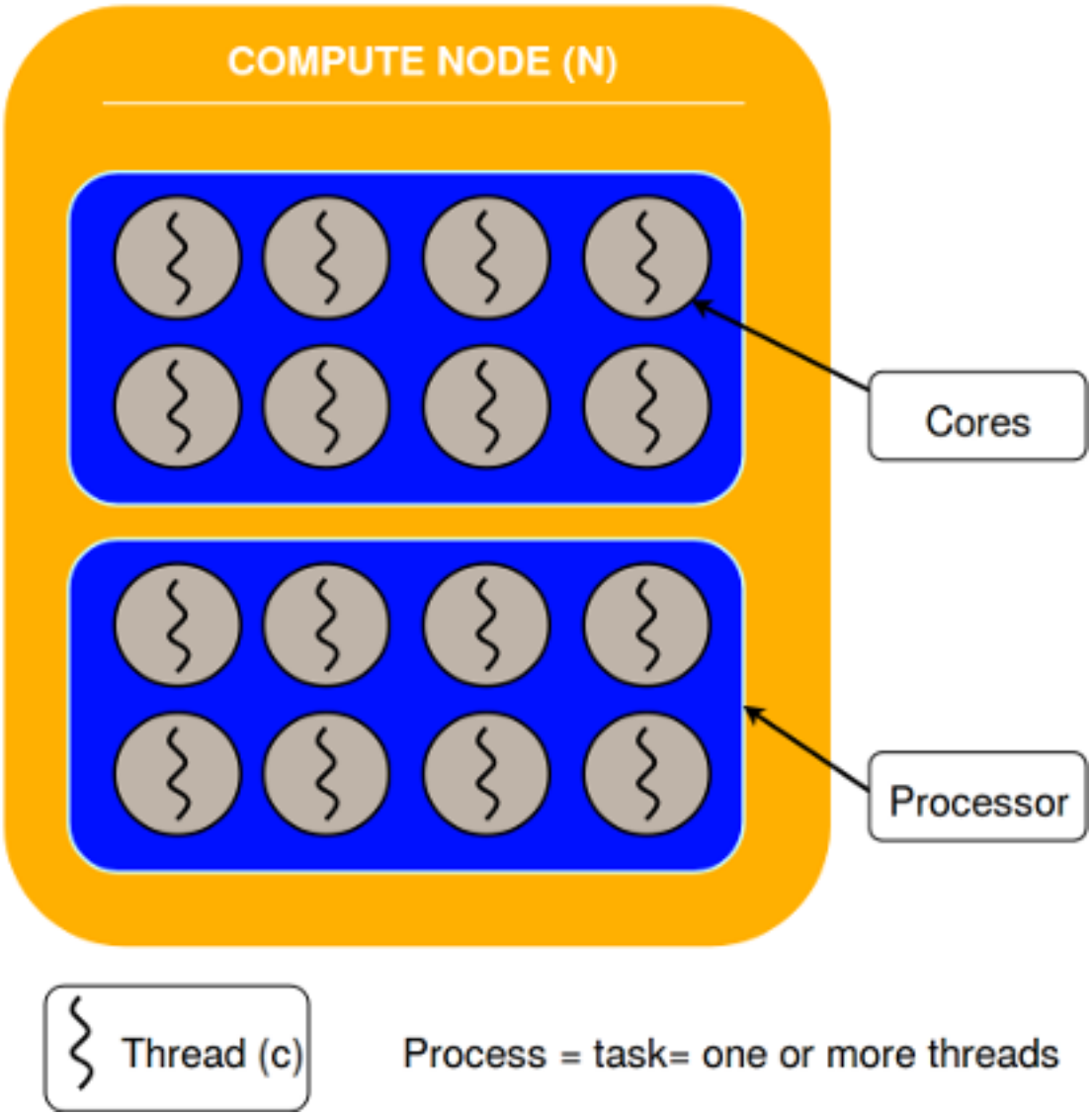
When you use `srun` inside a `sbatch` script, `srun` takes all the parameters given to that script, once the needed resources of your job are available and your job is ready to start running, `srun` will communicate to the Slurm compute node daemons to launch the `ntasks` of your job as it was specified.

In conclusion, as a **short answer**, you **need** to use `srun` inside your sbatch scripts when your job uses more than **one** `ntask` in order to allow Slurm to distribute these tasks in the cluster. Otherwise, the number of `ntasks` won't take effect and the job will be serial. Thus, you should use `srun` inside your MPI and Array jobs, in the *submit* section we have some examples of these cases.

What is the difference between `N`, `n` and `c`?

`N` refers to the number of **Nodes**. Nodes can compute one or more **tasks** (`n`), ideally a **node** can compute up to its number of cores at the same time, also depending on the specific application. Each task has one or more **threads** (`c`). We recommend to specify `n` tasks rather than `N` nodes, you should let Slurm decides how many machines your job needs.

Here is a good explanation about the difference between Process and Threads.



How to know the name(s) of the allocated node(s) inside my job ?

The are cases where is useful to know which node(s) has been allocated to run my job. In order to make your script dynamic, you can ask Slurm that information using:

```
# SBATCH PAMATERTS

scontrol show hostname > hostnames.out
```

(continues on next page)

(continued from previous page)

```
# job steps
```

This example store the name of the allocated machines in the file `hostname.out`. You can use it inside a `sbatch` script or as a command in a `salloc` sub-shell.

More information

- Quick Start User Guide.
- Slurm + tutorial - Software and Environment Module.
- Morris A. Jette , Andy B. Yoo , Mark Grondona (2002) SLURM: Simple Linux Utility for Resource Management.

Authors

Juan David Arcila-Moreno <jarcil13@eafit.edu.co>

3.4.3 Tools

In this subsection we present the current tools available for its use with SLURM.

DMTCP

DMTCP (Distributed MultiThreaded Checkpointing) transparently checkpoints a single-host or distributed computation in user-space with no modifications to user code or to the O/S. It works on most Linux applications, including Python, Matlab, R, etc.[1]

Warning: The current Apolo implementation only offers checkpointing support to serial and parallel programs. It isn't compatible with distributed programs like those based on MPI.

DMTCP-2.5.2

Basic Information

- **Deploy date:** 3 August 2018.
- **Official Website:** http://dmtcp.sourceforge.net/
- **License:** Lesser GNU Public License (LGPL)
- **Installed on:** *Cronos*
- **Supported versions:** Serial, parallel jobs

[1] DMTCP: Distributed MultiThreaded CheckPointing. (n.d.). Retrieved from http://dmtcp.sourceforge.net/

Installation

This entry covers the entire process performed for the installation and configuration of DMTCP on a cluster with the conditions described above.

Contents

Tested on (Requirements)

- **OS base:** CentOS (x86_64) $\geq$ 6.6
- **Scheduler:** SLURM $\geq$ 16.05.6
- **Compiler:** GNU GCC $\geq$ 4.4.7-11

Build process

This entry describes the installation process of DMTCP.

1. Get DMTCP latest version from its sourceforge page (page)
2. Send the installation package to the master node on your cluster.

```
scp dmtcp-2.5.2.tar.gz <usename>t@<FQDN>:$installer_path$
```

3. Unzip and access the installer files.

```
ssh -X root@<FQDN>
cd $installer_path$
tar xf dmtcp-2.5.2.tar.gz
```

4. Create the installation directory and change its owner to the user that it is doing this process.

```
mkdir -p /share/apps/dmtcp/2.5.2
chown <username>.<user group> /share/apps/dmtcp/2.5.2
```

5. Go to DMTCP path and build it with the acording flags presented here.

```
./configure \
--prefix=/share/apps/dmtcp/2.5.2 \
--build=x86_64-redhat-linux \
--enable-infiniband-support \
--enable-pthread-mutex-wrappers
make && make install
```

6. After this process, repeat the configure process to install the (**32 Bits**) compatibility following these commands.

```
./configure \
--prefix=/share/apps/dmtcp/2.5.2 \
--build=x86_64-redhat-linux \
--enable-infiniband-support \
--enable-pthread-mutex-wrappers \
--enable-m32
make clean
make && make install
```

7. Change the owner of the installation directory to finish this process.

```
chown root.root /share/apps/dmtcp/2.5.2
```

Module file

Listing 68: `Module file`

```
##%Module1.0#####################################################################
##
## module load dmtcp/2.5.2
##
## /share/apps/modules/dmtcp/2.5.2
## Written by Sebastian Patiño Barrientos
##

proc ModulesHelp {} {
     global version modroot
     puts stderr "Sets the environment for using dmtcp 2.5.2\
                  \nin the shared directory \
                  \n/share/apps/dmtcp/2.5.2/\
                  \nbuilded with gcc-4.4.7"
}

module-whatis "(Name________) dmtcp"
module-whatis "(Version_____) 2.5.2"
module-whatis "(Compilers___) gcc-4.4.7"
module-whatis "(System______) x86_64-redhat-linux"
module-whatis "(Libraries___) "

# for Tcl script use only
set         topdir        /share/apps/dmtcp/2.5.2/
set         version       2.5.2
set         sys           x86_64-redhat-linux

conflict dmtcp


prepend-path        PATH                        $topdir/bin

prepend-path        LD_LIBRARY_PATH               $topdir/lib
prepend-path        LIBRARY_PATH                $topdir/lib
prepend-path        LD_RUN_PATH               $topdir/lib

prepend-path        C_INCLUDE_PATH                $topdir/include
prepend-path        CXX_INCLUDE_PATH        $topdir/include
```

(continues on next page)

(continued from previous page)

```
prepend-path        CPLUS_INCLUDE_PATH        $topdir/include

prepend-path        MANPATH                         $topdir/share/man
```

Usage

This subsection explains a method for submiting jobs to the cluster and restarting them using DMTCP's checkpointing services.

For both types of jobs, in the SLURM launch script, load the necessary environment including DMTCP's module. After that, source the coordinator bash script in order to use the start_coordinator function. Remember to assing a checkpointing interval **in seconds** with the -i flag.

The last step in both cases is launching the program in the next way.

```
dmtcp_launch --rm <Your program binary> <args>...
```

For serial software

Listing 69: `Launch script example`

```
#!/bin/bash
# Put your SLURM options here
#SBATCH --time=00:30:00           # put proper time of reservation here
#SBATCH --nodes=1                 # number of nodes
#SBATCH --ntasks-per-node=1       # processes per node
#SBATCH --job-name=serial_example     # change to your job name
#SBATCH --output=serial_example.%j.out # Stdout (%j expands to jobId)
#SBATCH --error=serial_example.%j.err # Stderr (%j expands to jobId)

module load dmtcp

source coordinator.sh

###############################################################################
# 1. Start DMTCP coordinator
###############################################################################

start_coordinator -i 35


###############################################################################
# 2. Launch application
###############################################################################

dmtcp_launch --rm  ./serial_example
```

Listing 70: `Restart script example`

```
#!/bin/bash
# Put your SLURM options here
#SBATCH --time=00:02:00           # put proper time of reservation here
#SBATCH --nodes=1                 # number of nodes
```

(continues on next page)

(continued from previous page)

```
#SBATCH --ntasks-per-node=1       # processes per node
#SBATCH --job-name=serial_example     # change to your job name
#SBATCH --output=serial_example.%j.out # Stdout (%j expands to jobId)
#SBATCH --error=serial_example.%j.err # Stderr (%j expands to jobId)

module load dmtcp

source coordinator.sh

###############################################################################
# 1. Start DMTCP coordinator
###############################################################################

start_coordinator

###############################################################################
# 2. Restart application
###############################################################################

/bin/bash ./dmtcp_restart_script.sh -h $DMTCP_COORD_HOST -p $DMTCP_COORD_PORT
```

For parallel software

In this example we run an OpenMP application. Notice that in the restart script we don't assign again the OMP_NUM_THREADS variable again.

Listing 71: `Launch script example`

```
#!/bin/bash
# Put your SLURM options here
#SBATCH --time=00:02:00           # put proper time of reservation here
#SBATCH --nodes=1                 # number of nodes
#SBATCH --ntasks-per-node=8       # processes per node
#SBATCH --job-name=parallel_example     # change to your job name
#SBATCH --output=parallel_example.%j.out # Stdout (%j expands to jobId)
#SBATCH --error=parallel_example.%j.err # Stderr (%j expands to jobId)

module load dmtcp

source coordinator.sh

export OMP_NUM_THREADS=8

###############################################################################
# 1. Start DMTCP coordinator
###############################################################################

start_coordinator -i 35


###############################################################################
# 2. Launch application
###############################################################################

dmtcp_launch --rm  ./parallel_example
```

Listing 72: Restart script example

```
#!/bin/bash
# Put your SLURM options here
#SBATCH --time=00:02:00           # put proper time of reservation here
#SBATCH --nodes=1                 # number of nodes
#SBATCH --ntasks-per-node=8       # processes per node
#SBATCH --job-name=parallel_example     # change to your job name
#SBATCH --output=parallel_example.%j.out # Stdout (%j expands to jobId)
#SBATCH --error=parallel_example.%j.err # Stderr (%j expands to jobId)

module load dmtcp

source coordinator.sh

################################################################################
# 1. Start DMTCP coordinator
################################################################################

start_coordinator

################################################################################
# 2. Restart application
################################################################################

/bin/bash ./dmtcp_restart_script.sh -h $DMTCP_COORD_HOST -p $DMTCP_COORD_PORT
```

Sending commands to the coordinator

If you want to send commands to the coordinator of a set of processes, the *start_coordinator* function you used in the script generates in your launch directory a *dmtcp_command.<job_id>* file. Using this, you can communicate with your applications currently running. You can use this to generate a manual checkpoint or to change the checkpointing interval.

Examples

For launching a manual checkpoint use this command

```
$JOBDIR/dmtcp_command.$JOBID -c
```

For changing the checkpointing interval use this command

```
$JOBDIR/dmtcp_command.$JOBID -i <time_in_seconds>
```

Authors

- Sebastian Patiño Barrientos <spatino6@eafit.edu.co>

3.5 Provisioning

This entry contains relevant information on how to use ansible as a provisioning tool to maintain system configurations and automate several processes within a cluster.

3.5.1 Ansible

Basic information

- **Official Website:** https://www.ansible.com/
- **License:** GNU GPL v3
- **Installed on:** *Apolo II*, *Cronos*

Preliminaries

Information contained within this section elucidate the basic usage of ansible's common features (i.e. those which, at the very least, allow it to perform simple tasks).

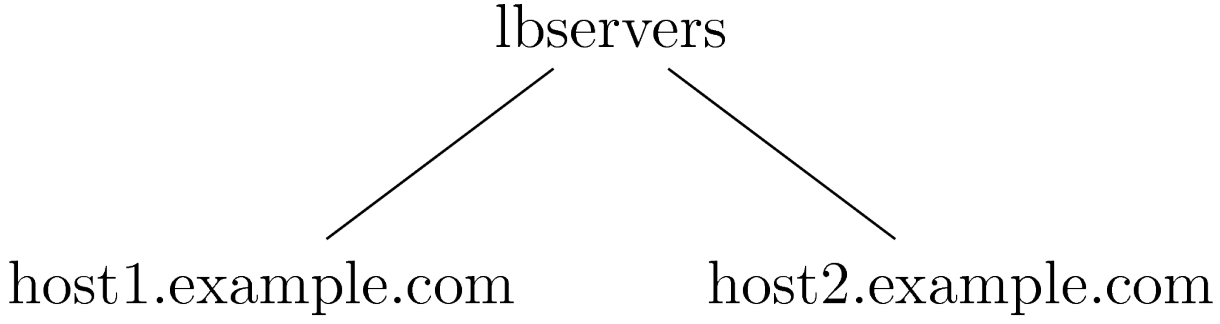
Inventory

Ansible performs actions, also known as tasks, over a group of computers; in order to do so it reads a plain text file called "inventory file" containing a list of hostnames, or IP addresses, often grouped based on one or multiple shared features.

The inventory file is located by default under `/etc/ansible/hosts` and would typically follow the conventions shown below:

1. Group names are delimited by `[` and `]`. e.g. group lbservers would be written as `[lbservers]`.
2. Hosts below a group definition are to be taken as members of it. e.g.

```
; lbservers -> Group
; [host1,host2].example.com -> Members
[lbservers]
host1.example.com
host2.example.com
```



3. Using the suffix `:children` within a group definition indicates the presence of nested groups (i.e. subgroups). e.g.

```
; lbservers -> Group
; lb[south,north] -> Subgroups
[lbservers:children]
lbsouth
lbnorth
```

Note: Subgroups are only declared as part of a parent-child relation (i.e. nesting depth is 1), thus implying that relations where nesting depth is greater than 1 require multiple declarations.

```
; lbservers -> Grandparent
; lb[south,north] -> Children
[lbservers:children]
lbsouth
lbnorth

; lbs[1,2].example.com -> Grandchildren
[lbsouth]
lbs1.example.com
lbs2.example.com
```

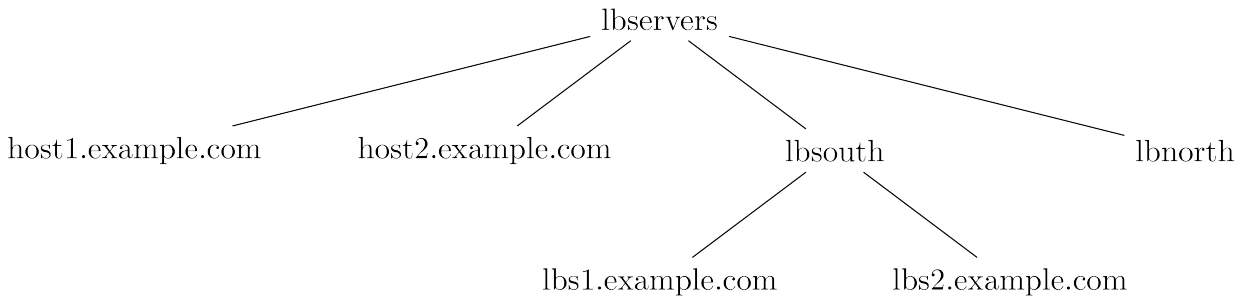
4. The suffix `:vars` within a group definition is used to declare and assign variables to a particular set of hosts or subgroups. e.g.

Note: These variables are relative to group members and can be overwritten by subgroups and other ansible components (e.g. playbooks, tasks). See Ansible's Variable Precedence article for more information.

```
; lbsouth and lbnorth will inherit all
; variables declared within lbservers.
[lbservers:children]
lbsouth
lbnorth

[lbservers:vars]
requests_timeout=5
max_hosts_to_serve=10

; "requests_timeout" will be overwritten
; for lbsouth members only.
[lbsouth:vars]
requests_timeout=3

; Members of this group will not recognize
; variables declared for lbservers, as they
; do not belong to it.
[backupservers]
bk1.example.com
bk2.example.com
```

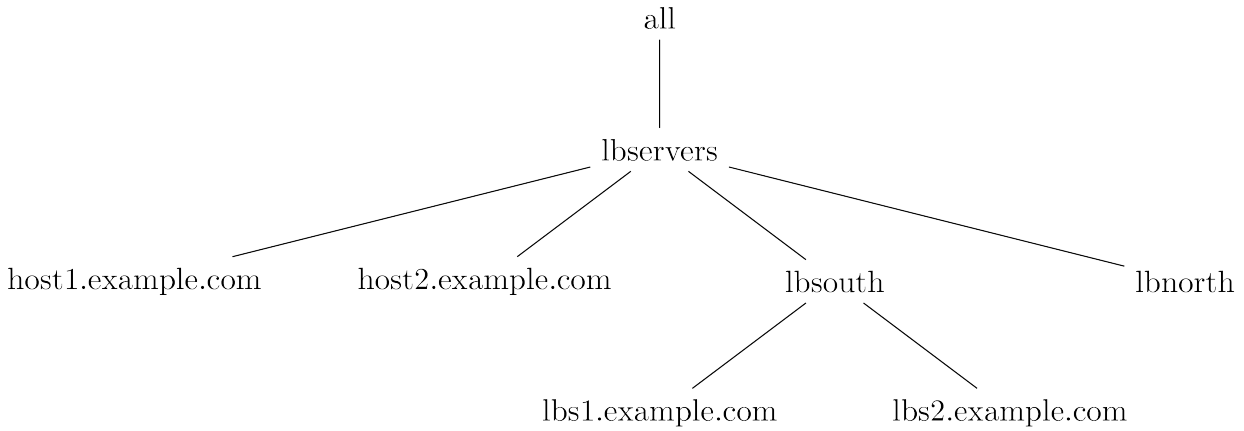
It is impotant to highlight that there are two default groups: `all` and `ungrouped`, which, unlike any other group, can be omitted within the inventory file, as their definitions are both implicit. Please be aware that:

1. Hierarchically, all groups and hosts are members of `all`.
2. Hosts with no group other than all belong to `ungrouped`. Therefore, hosts will be members of at least two



groups.

Hence, it is true for the examples above:



Group variables

Note: This feature will not be detailed, as there is plenty of information about it in Ansible's document: Working with Inventory

Keeping too much data within the inventory file can make it become complex, difficult to read and maintain. Ansible allows to easily bypass this issue by introducing a mechanism to split groups and hosts data:

1. Create a folder called `group_vars` at the same level as the inventory file. That is, if the inventory file is located under `$ANSIBLE_HOME` then the folder must be there as well. e.g.

```
mkdir -p $ANSIBLE_HOME/group_vars
ls $ANSIBLE_HOME/
inventory    group_vars/
```

2. Create files under `group_vars` matching your group names and store the corresponding variables into each one. Take the example from the *Inventory* section; There are variables declared for two groups, hence there would be two files under `group_vars` as shown below:

```
# $ANSIBLE_HOME/group_vars/lbservers
---
requests_timeout: 5
max_hosts_to_serve: 10
```

```
# $ANSIBLE_HOME/group_vars/lbsouth
---
requests_timeout: 3
```

Moreover, variables within a group can be further organized by decoupling the files inside `group_vars`. Ansible will read all files under directories named after groups or hosts. For instance, variables from the lbservers group can reside in multiple files under `$ANSIBLE_HOME/group_vars/lbservers/`. e.g.

```
# $ANSIBLE_HOME/group_vars/lbservers/requests
---
requests_timeout: 5
```

```
# $ANSIBLE_HOME/group_vars/lbservers/config
---
max_hosts_to_serve: 10
```

Modules

A module can be interpreted as a function ansible calls from a task. Basically, a module is the function's entire body (i.e. declaration), waiting to be called from a task or an ansible ad-hoc command.

Playbooks

A playbook is a text file, written in YAMl format, containing information on which tasks to apply on which hosts. This information is contained within a definition block called "Play". Take the following playbook for example:

```
---
- hosts: lbsouth
  vars:
    nginx_conf_dir: /etc/nginx/

- hosts: lbnorth
  vars:
    nginx_conf_dir: /opt/nginx/

- hosts: lbservers
  vars:
    nginx_log_dir: /var/log/ansible
  tasks:
  - name: Install/update nginx
    yum:
      name: nginx
      state: latest
  - name: Place nginx config file
    template:
      src: templates/nginx.conf.j2
      dest: "{{ nginx_conf_dir }}/nginx.conf"
    notify:
      - restart nginx
  - name: Ensure nginx is running
    systemd:
      name: nginx
      state: started
```

(continues on next page)

(continued from previous page)

```
    enabled: true
  handlers:
    - name: restart nginx
      systemd:
        name: nginx
        state: restarted
```

Plays are separated by a non-printable '\n', thus there are three plays. Each one uses the keyword "hosts" to describe a group, defined in the inventory file, on which to apply some tasks and/or set variables, keywords "tasks" and "vars" respectively.

An easy way to comprehend what a playbook is, and why it is useful, is thinking on what would one need to do in scripting languages, like bash, to accomplish what a playbook is meant to. Take the task "Place nginx config file". It calls Ansible's `template` module, which creates a file based on a Jinja2 template. Hence, one could either use templates alongside bash, which becomes complex and difficult to maintain really fast, use an external software to parse them, like ruby `erb` or python + Jinja2, or manage static files. Thereupon, additional concerns arise: how to deliver files to lbservers' hosts?, how to manage variables within them?, etc. Basically, these questions represent steps to achieve something specific (for the task under consideration, place a file called `nginx.conf`, whose content may vary, on all hosts within lbservers) that can be interpreted as to lead a system to a desired state. e.g.

- Original state: lbservers' hosts not having `nginx.conf`
- Desired state: lbservers' hosts having `nginx.conf`

A playbook can be, therefore, defined as the abstraction of a system's final state, comprised of intermediate states represented by tasks. Sort of an assembly line analogy:

Task 1 would represent an ansible run being triggered, tasks 2 to 5 the system's pass through each intermediate state (i.e. bun toasted, bun assembled with condiments, patty wrapped, Order placed on heated landing pad) and task 6 the desired state (i.e. customer satisfied).

Roles

A role is a hierarchical directory structure intended to decouple playbooks by breaking them into multiple files, which is particularly useful to create reusable components and write simpler playbooks. A role's layout would typically look as below:

Note: There are more directories than those listed below. See Ansible's official documentation for more information.

```
<playbook 1>
<playbook 2>
.
.
.
<playbook n>
inventory
roles/
  common/
    tasks/
    handlers/
    files/
    templates/
    vars/
```

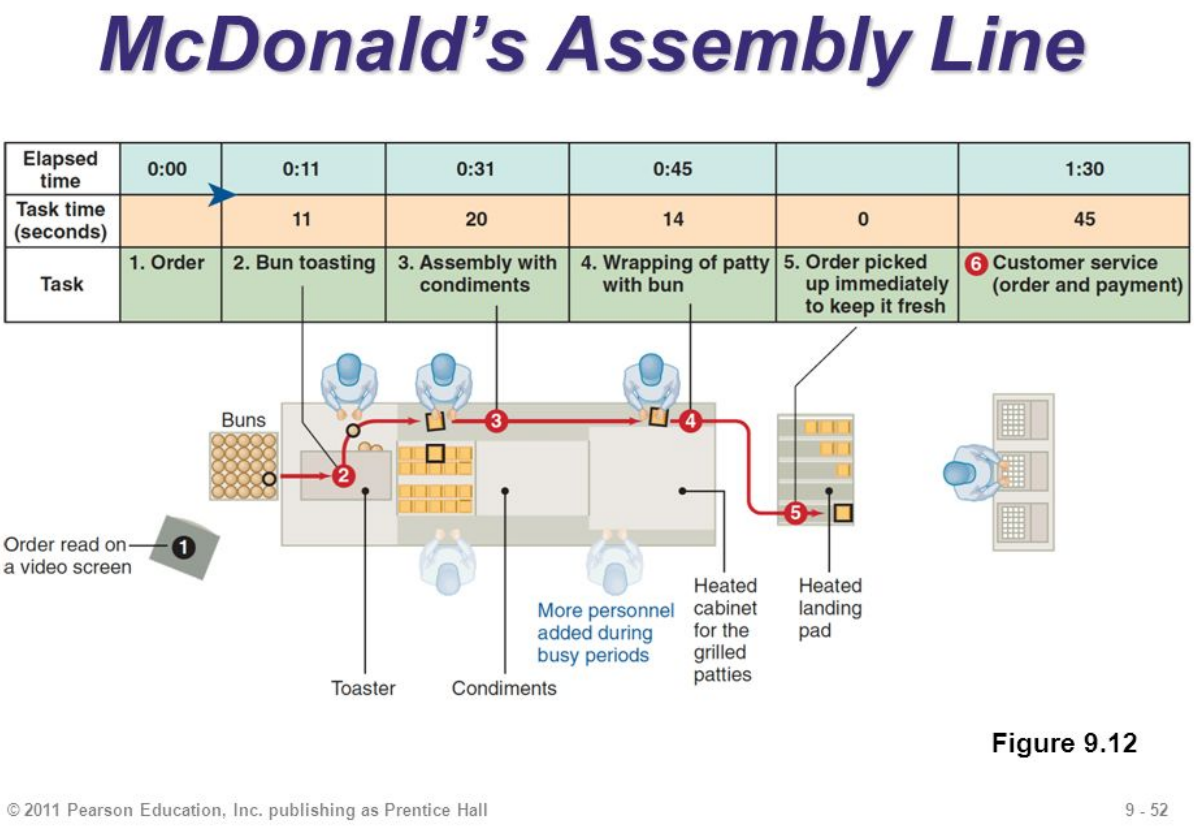


Fig. 1: McDonald's assembly line. Retrieved august 28, 2018 from https://slideplayer.com/slide/9882222/

Let us elucidate on how playbooks can be decoupled by using the notion of a role. Take the example on the *Playbooks* section.

1. **Identify a common feature within your tasks**. For example, all tasks on the third play are related to nginx.
2. Use that common feature as a base to name your role and create a directory under `$ANSIBLE_HOME/roles`.

> **Note:** `$ANSIBLE_HOME` is used as a way to represent ansible's folder location within the filesystem (e.g. /etc/ansible), which may vary depending on the setup.

```
mkdir -p  $ANSIBLE_HOME/roles/nginx
```

3. **Decouple tasks by placing them in taskfiles**. As the name implies, a taskfile is a file containing task declarations; this files are often stored under `$ANSIBLE_HOME/roles/<role>/tasks` and their name is irrelevant exept for `main.yml`, which must always be present. Although tasks can be all defined inside `main.yml`, it is recommended to declare them in different taskfiles when their number is large enough to make a coupled taskfile difficult to read, and then call each one from `main.yml`.

```
# $ANSIBLE_HOME/roles/nginx/tasks/packages.yml
---
- name: Install/update nginx
  yum:
    name: nginx
    state: latest
```

```
# $ANSIBLE_HOME/roles/nginx/tasks/config.yml
---
- name: Place nginx config file
 template:
   src: templates/nginx.conf.j2
   dest: "{{ nginx_conf_dir }}/nginx.conf"
 notify:
   - restart nginx

- name: Ensure nginx is running
  systemd:
    name: nginx
    state: started
    enabled: true
```

```
# $ANSIBLE_HOME/roles/nginx/tasks/main.yml
---
- name: "Including taskfile {{ taskfile }}"
  include_tasks: "{{ taskfile }}"
  with_items:
    - 'packages.yml'
    - 'config.yml'
  loop_control:
    loop_var: taskfile
```

4. **Decouple variables**. Declare them as *Group variables*, in the role's local context or within a task. For instance, if one desires the variable `nginx_log_dir` to be set for all hosts applying the nginx role:

> **Note:** Using `$ANSIBLE_HOME/roles/<role>/vars` to store variables visible to all tasks within a role is a common practice. However, "vars" can be named differently or even placed under some other location.

One would typically store variables inside `$ANSIBLE_HOME/roles/<role>/vars/main.yml` as for ansible to auto-load them, but there is also the alternative to do it manually (shown in this example).

```
mkdir -p $ANSIBLE_HOME/roles/nginx/vars
```

```
# $ANSIBLE_HOME/roles/nginx/vars/config.yml
---
nginx_log_dir: /var/log/ansible
```

```
# $ANSIBLE_HOME/roles/nginx/tasks/main.yml
---
# Unlike group_vars, ansible does not read files
# inside the vars folder automatically, except "main.yml".
# Therefore, in this case, it must explicitly be told to do so.
# Remark: vars' location may vary.
- name: 'Include variables'
  include_vars:
    dir: '../vars'
    extensions:
      - yml

- name: "Including taskfile {{ taskfile }}"
  include_tasks: "{{ taskfile }}"
  with_items:
    - 'packages.yml'
    - 'config.yml'
  loop_control:
    loop_var: taskfile
```

As for the variables under `lbsouth` and `lbnorth`:

```
# $ANSIBLE_HOME/group_vars/lbnorth
---
nginx_conf_dir: /opt/nginx/conf
```

```
# $ANSIBLE_HOME/group_vars/lbsouth
---
requests_timeout: 3
nginx_conf_dir: /etc/nginx/conf
```

5. **Decouple handlers**. Handlers are stored the same way taskfiles are, but in a different location. They are placed inside the "handler" directory, which is at the same level as the "tasks" directory.

```
mkdir -p $ANSIBLE_HOME/roles/nginx/handlers
```

```
# $ANSIBLE_HOME/roles/nginx/handlers/main.yml
---
- name: restart nginx
  systemd:
    name: nginx
    state: restarted
```

6. **Decouple templates**. Stored under `$ANSIBLE_HOME/roles/<role>/templates`, it is highly recommended to create a directory structure resembling that of the location where templates will be rendered. e.g. `nginx.conf` will be rendered in `/etc/nginx/conf` for `lbsouth` and `/opt/nginx/conf`, for

lbnorth, hence the template would reside in either $ANSIBLE_HOME/roles/nginx/templates/etc/nginx/conf or $ANSIBLE_HOME/roles/nginx/templates/opt/nginx/conf. Note modifying the layout also implies adjusting all tasks using nginx.conf.j2.

```
# $ANSIBLE_HOME/roles/nginx/tasks/config.yml
---
- name: Place nginx config file
 template:
   src: templates/etc/nginx/conf/nginx.conf.j2
   dest: "{{ nginx_conf_dir }}/nginx.conf"
 notify:
   - restart nginx

- name: Ensure nginx is running
  systemd:
    name: nginx
    state: started
    enabled: true
```

7. **Call the role** from the playbook (Note how it became simpler).

```
---
- hosts: lbservers
  roles:
    - nginx
```

Finally, consider the designated behavior for each role 'x' component (Taken from[5]):

- If roles/x/tasks/main.yml exists, tasks listed therein will be added to the play.
- If roles/x/handlers/main.yml exists, handlers listed therein will be added to the play.
- If roles/x/vars/main.yml exists, variables listed therein will be added to the play.
- If roles/x/defaults/main.yml exists, variables listed therein will be added to the play.
- If roles/x/meta/main.yml exists, any role dependencies listed therein will be added to the list of roles (ansible 1.3 and later).
- Any copy, script, template or include tasks (in the role) can reference files in roles/x/{files,templates,tasks}/ (dir depends on task) without having to path them relatively or absolutely.

Vault (Encryption)

Note: Some features will not be detailed. Basic usage can be found in Ansible's document: Ansible Vault

"New in Ansible 1.5, "Vault" is a feature of ansible that allows keeping sensitive data such as passwords or keys in encrypted files, rather than as plaintext in your playbooks or roles. These vault files can then be distributed or placed in source control."[1]

[5] Using Roles, September 06 - 2018. Retrieved September 06 - 2018, from https://docs.ansible.com/ansible/2.5/user_guide/playbooks_reuse_roles.html#using-roles

[1] Ansible Vault, August 17 - 2018. Retrieved August 30 - 2018, from https://docs.ansible.com/ansible/latest/user_guide/vault.html?highlight=vault

Create encrypted files

The command below will create a temporary file and subsequentially open it for you to write. Once the file is saved, and the text editor closed, *ansible-vault* will automatically generate an encrypted version of it and erase the original.

```
ansible-vault --vault-id <env>@<vault-password script> create <file>
```

Alternatively, if you intend to be prompted for the password, then:

```
# You could also use:
# ansible-vault create <file>
# However, environments would not be taked into consideration.
ansible-vault --vault-id <env>@prompt create <file>
```

Encrypt files

```
ansible-vault --vault-id <env>@<vault-password script> encrypt <file-1>␣
↪[file-2 file-3 ... file-n]
```

Encrypt variables as a string

```
ansible-vault encrypt_string --vault-id <env>@<vault-password script> --
↪stdin-name '<varname>'
```

Edit encrypted files

Encrypted files can be edited without being decrypted a priori. See the command below:

```
ansible-vault --vault-id <env>@<vault-password script> edit <file>
```

View encrypted file

```
ansible-vault --vault-id <env>@<vault-password script> view <file>
```

View encrypted string

```
ansible localhost -m debug -a var='<variable_to_decrypt>' \
-e "@<file_containing_variable>" \
--vault-id <env>@<vault-password script>
```

Decrypt files

```
ansible-vault --vault-id <env>@<vault-password script> decrypt <file-1>␣
↪[file-2 file-3 ... file-n]
```

Change encryption key

```
ansible-vault rekey <file-1> [file-2 file-3 ... file-n]
```

vault-password script

Vault's password can be retrieved from a script, as described in[2], passed to the option `--vault-id`, or `--vault-password-file` from the `ansible-vault` and `ansible-playbook` executables.

Note: The script can be written in python, bash or any other scripting language.

Scripts invoked by `--vault-password-file` take no arguments, return the password on stdout and do not have any knowledge about `--vault-id` or multiple password files whatsoever. Using `--vault-id` to call upon scripts, on the other hand, enables a 'protocol' under which a vault id can be looked up and its associated password returned thereafter.

Furthermore, `--vault-id` allows for a vault id to be passed a as an argument thus giving developers the ability to program more sophisticated vault-password scripts.

Warning:

- A vault id will only be passed to the script if the latter is named after the convention `<some name>-client.<extension>` (e.g. `keyring-client.sh`). See[3] and[4] for more information.
- Make sure the script is executable. Otherwise, ansible will not be able to use it.

For instance,

```
ansible-playbook --vault-id some_id@/path/to/keyring-client.sh some_playbook.yml
```

will result in `keyring-client.sh` to be invoked as:

```
/path/to/keyring-client.sh --vault-id some_id
```

Let us delve into a more detailed example:

Assumptions

1. Ansible is being run from three clusters.Cluster orchestrators (masters) are named after the convention `cluster<cluster number>.<domain>` and compute nodes `compute<cluster number>-<number>`. e.g. Cluster 1 is comprised of `cluster1.local` and `compute-1-0.local`, `compute-1-1.local`.
2. Clusters 1 and 2 belong to the production environment. Cluster 3 belongs to the development environment.
3. Servers from a particular cluster cannot access servers from other cluster.
4. The script `/usr/sbin/keyring-client.sh` has the content shown below:

[2] Ansible Vault, Providing Vault Passwords, August 17 - 2018. Retrieved August 30 - 2018, from https://docs.ansible.com/ansible/latest/user_guide/vault.html?highlight=vault#providing-vault-passwords.

[3] Issue: Allow the vault_id to be passed to vault password scripts #31001, September 27 - 2018. Retrieved Retrieved August 30 - 2018, from https://github.com/ansible/ansible/issues/31001

[4] Vault secrets client inc new 'keyring' client #27669, October 13 - 2018. Retrieved August 30 - 2018, from https://github.com/ansible/ansible/pull/27669

```
#!/bin/bash

case $1 in
  "--vault-id")
  declare -r env="$2"
  ;;
*)
  exit 1
  ;;
esac

declare -r cluster=`hostname | awk -F'.' '{print $1}'`
declare -r cmd="ssh remote \
cat /etc/secrets/$env/$cluster"

declare -r vault_passwd="$($cmd)"

echo "$vault_passwd"
```

5. The vault id reprents an environment: dev (development), prod (production).
6. A server called `remote` (see line 13 from script) holds multiple passwords, one per cluster, stored under `/etc/secrets/<environment>/<cluster>`:
   - `/etc/secrets/prod/cluster1`
   - `/etc/secrets/prod/cluster2`
   - `/etc/secrets/dev/cluster3`

Sample use case

1. Create a git repository to hold ansible's information.

```
mkdir -p ~/ansible
cd ~/ansible
git init
```

2. Create an inventory file.

```
; ~/ansible/inventory

[cluster1]
cluster1.local
compute-1-0.local
compute-1-1.local

[cluster2]
cluster2.local
compute-2-0.local
compute-2-1.local

[cluster3]
cluster3.local
compute-3-0.local
compute-3-1.local

[clusters]
cluster1
```

(continues on next page)

(continued from previous page)

```
cluster2
cluster3
```

3. Create a playbook to change the root password. Since repeating code is an awful practice, we decided to create a reusable task and manage the user password through a variable.

```
# ~/ansible/playbook.yml
---
- hosts: clusters
  tasks:
  - name: Set root password
    user:
      name: root
      password: "{{ root_password_hash }}"
```

4. Retrive each root password hash.

```
# Password - cluster1: 123
openssl passwd -1 -salt
Password:
Verifying - Password:
$1$PpScqWH9$/Rpsq9/mJVxnaCEmrSAv31
# Password - cluster2: 456
openssl passwd -1 -salt
Password:
Verifying - Password:
$1$RB/C07h4$t1lWpEQO/APEBwYPyhjai1
# Password - cluster3: 789
openssl passwd -1 -salt
Password:
Verifying - Password:
$1$mRBrUoTy$xAoiS8xIeT6pm8HZZvKmL1
```

5. Encrypt the hashes using the vault-password script. Note the process is exactly the same for all servers (login, run ansible-vault, paste hash, press Ctrl-d, retrieve hash), therefore showing it for one will be enough of a clarification.

Caution:

- DO NOT underestimate string trimming. That is, Vault does not trim any \n. Hence, pasting the hash, pressing [Return] and then [Ctrl]-[d] would include an EOL.
- Remember to give Vault's –vault-id option the apropriate environment for each server.

```
ssh cluster1.local
ansible-vault encrypt_string \
--vault-id prod@/usr/sbin/keyring-client.sh \
--stdin-name 'root_password_hash'
Reading plaintext input from stdin. (ctrl-d to end input)
$1$PpScqWH9$/Rpsq9/mJVxnaCEmrSAv31root_password_hash: !vault |
    $ANSIBLE_VAULT;1.2;AES256;prod
  ␣
↪3437666664633561656164396561376361316362326266326231396161326231656562323736343 4
  ␣
↪6138363635336330616364633539653466323264653133330a3264653461363836353439613464 34
```

(continues on next page)

(continued from previous page)

```
↪6637666535653461636633346534616663336437343862313362336330326234346466326662333
↪7
↪6136363864643936620a373734656435376331393265653138613835336237636437656666663361
↪
↪6663613061323238376665613430656635356233316632316466373162323835343063383034383
    3
    6131643734643639383332613635323264363065316464366232
Encryption successful
exit
```

6. Create the group variable `root_password_hash` and assign it the appropriate hash.

```
mkdir -p ~/ansible/group_vars
```

```
# ~/ansible/group_vars/cluster1
---
root_password_hash: !vault |
    $ANSIBLE_VAULT;1.2;AES256;prod
    ↪
    ↪3437666664633561656164396561376361316362326266326231396161326231656562323736343
    ↪
    ↪6138363635336330616364633539653466323264653133330a326465346136383635343961346434
    ↪
    ↪6637666535653461636633346534616663336437343862313362336330326234346466326662333
    ↪
    ↪6136363864643936620a373734656435376331393265653138613835336237636437656666663361
    ↪
    ↪6663613061323238376665613430656635356233316632316466373162323835343063383034383
    3
    6131643734643639383332613635323264363065316464366232
```

```
# ~/ansible/group_vars/cluster2
---
root_password_hash: !vault |
    $ANSIBLE_VAULT;1.2;AES256;prod
    <encrypted hash>
```

```
# ~/ansible/group_vars/cluster3
---
root_password_hash: !vault |
    $ANSIBLE_VAULT;1.2;AES256;dev
    <encrypted hash>
```

Note how each vault id corresponds to the cluster's environment, which, in this case, determines the script's behavior (see figure *Sample vault script workflow*).

7. Connect the repository to Github, Gitlab or any other remote platform. Then commit and push the changes.

```
cd ~/ansible
git remote add origin git@github.com:username/ansible
git add --all
git commit -m "<some message>"
git push -u origin master
```

8. Download the repository from each cluster orchestrator and run ansible.

Warning: Since clusters cannot see each other, ansible will only apply changes to the servers belonging to the same cluster an orchestrator is member of despite the existance of multiple cluster declarations within the inventory file. This approach, however, is not recommended for a production environment.

```
ssh cluster1.local
cd /etc
git clone git@github.com:username/ansible
ansible-playbook --vault-id prod@/usr/sbin/keyring-client.sh \
-i /etc/ansible/inventory \
/etc/ansible/site.yml
exit

ssh cluster2.local
cd /etc
git clone git@github.com:username/ansible
ansible-playbook --vault-id prod@/usr/sbin/keyring-client.sh \
-i /etc/ansible/inventory \
/etc/ansible/site.yml
exit

ssh cluster3.local
cd /etc
git clone git@github.com:username/ansible
ansible-playbook --vault-id dev@/usr/sbin/keyring-client.sh \
-i /etc/ansible/inventory \
/etc/ansible/site.yml
exit
```

In order to decrypt the variable `root_password_hash` ansible executes `/usr/sbin/keyring-client.sh`, which:

1. Acesses `remote` using ssh
2. Retrieves the appropriate password, contingent on the cluster's name and environment.
3. Prints the password to the standard output.

The workflow depicted in the figure *Sample vault script workflow* shows what ansible will do on each cluster.

Environments

Environments provide a way to reuse ansible components (tasks, roles, playbooks, etc.) on multiple systems by maintaining different inventory files within the same project; which might also mean multiple `group_vars` and `host_vars` folders. Environments are usually used for testing purposes, such as verifying the integrity of features to be introduced in production servers.

Instead of being an ansible feature, environments are more of a concept materialized on the project's directory layout. Take the example from the *Roles* section:

```
playbook.yml
inventory
group_vars/
  lbnorth
  lbsouth
roles/
  nginx/
```

(continues on next page)

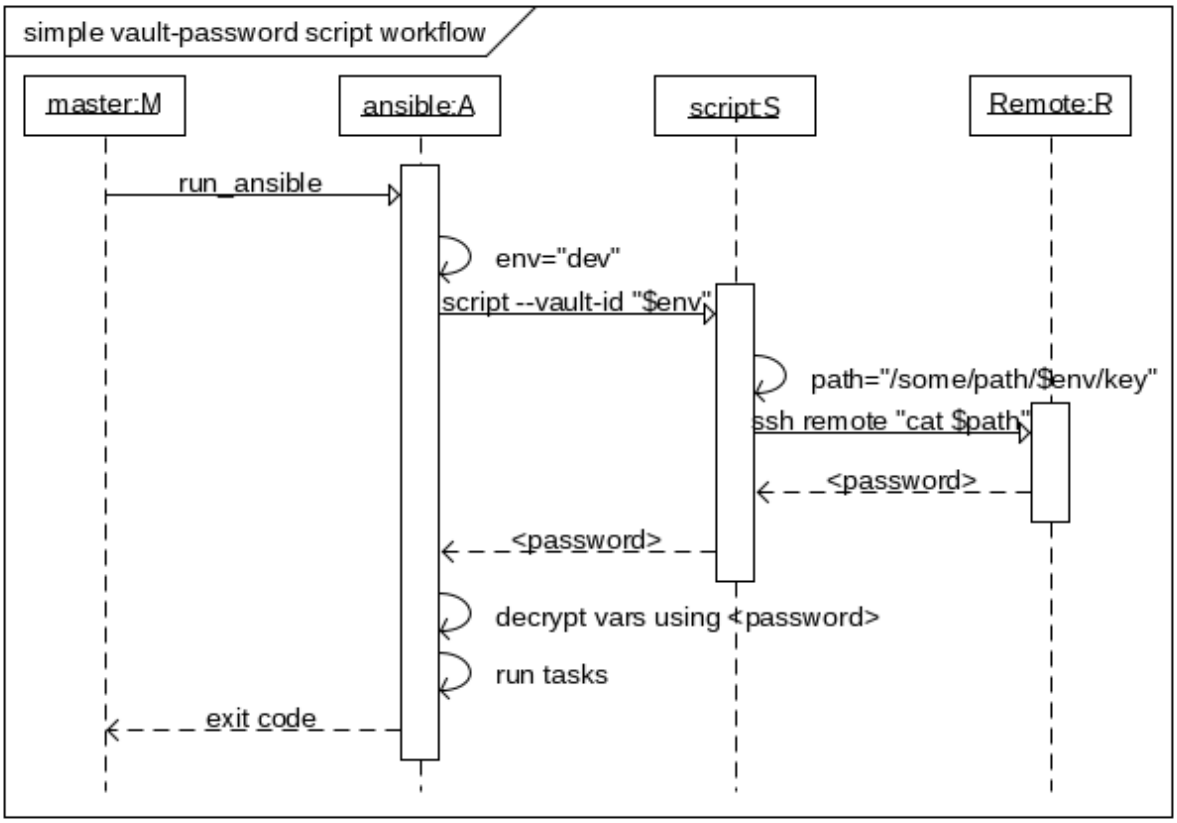


Fig. 2: Sample vault script workflow

(continued from previous page)

```
    tasks/
      packages.yml
      config.yml
      main.yml
    handlers/
      main.yml
    templates/
      /etc/nginx/conf/nginx.conf.j2
    vars/
      config.yml
```

If one were to constantly update the production servers, it would be wise to test changes on a system replica before releasing them. In order to create such replica, either using a virtual or physical machine, one must ensure equivalent operations to be applied over the testing servers, thus compelling them to become alike to their production counterparts. This is accomplished by reusing the entire project, but running ansible against different inventory files on each system:

```
playbook.yml
environments/
  production/
    inventory
    group_vars/
  development/
    inventory
    group_vars/
group_vars/
  lbnorth
  lbsouth
roles/
  nginx/
    tasks/
      packages.yml
      config.yml
      main.yml
    handlers/
      main.yml
    templates/
      /etc/nginx/conf/nginx.conf.j2
    vars/
      config.yml
```

- For production servers

```
ansible-playbook -i $ANSIBLE_HOME/environments/production/inventory \
$ANSIBLE_HOME/playbook.yml
```

- For testing servers

```
ansible-playbook -i $ANSIBLE_HOME/environments/development/inventory \
$ANSIBLE_HOME/playbook.yml
```

Control over tasks is done by employing variables, tags and/or other metaparameters. Ansible will load variables from the specified environment, expand them and run tasks accordingly, as depicted in figure *Simple multi-environment activity diagram*.

References

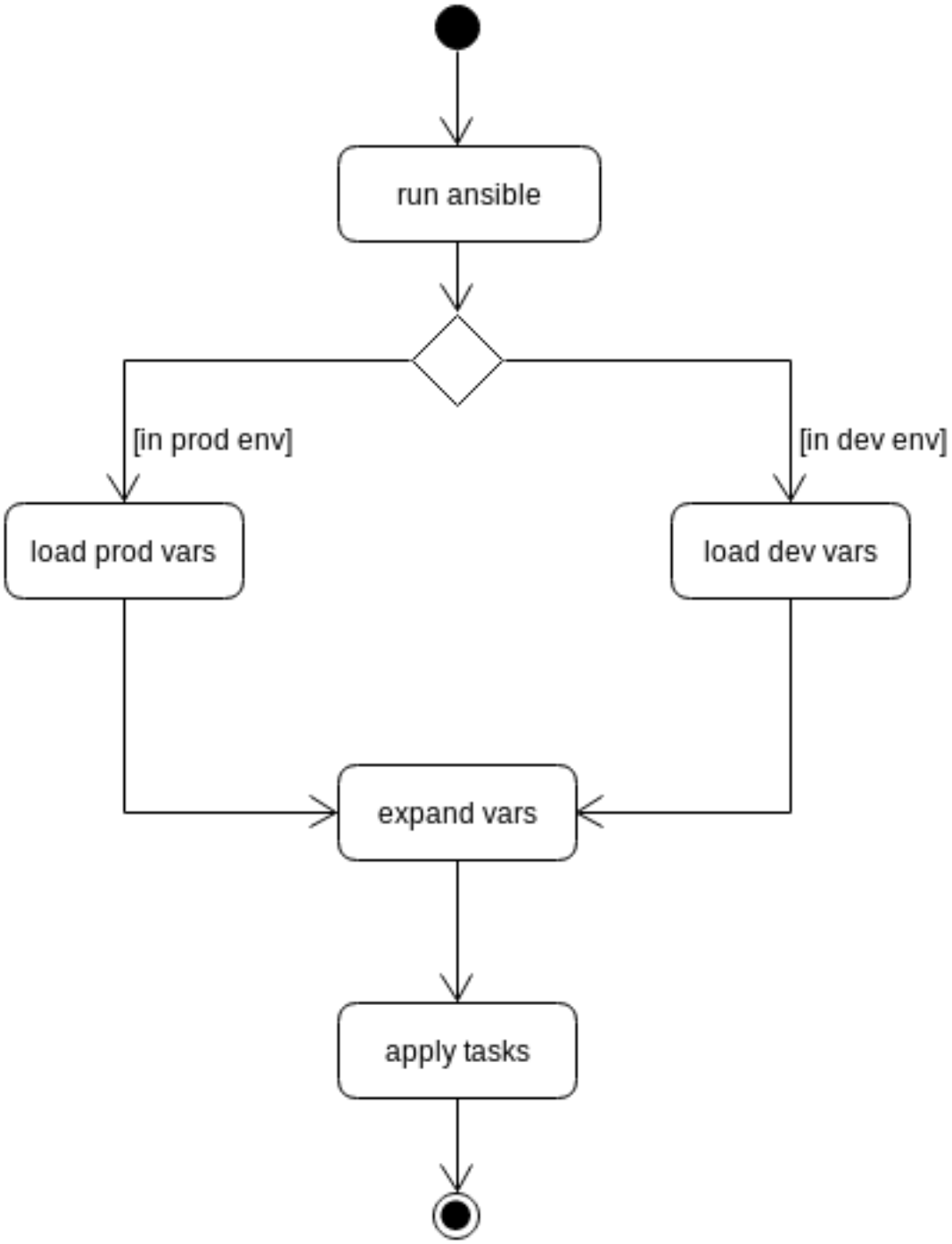


Fig. 3: Simple multi-environment activity diagram

Dynamic control

This section describes mechanisms to control which tasks are to be executed in a particular run.

Tags

Running specific parts of a playbook, role or taskfile is what tags are all about. Their purpose is to filter tasks based on a tag and then execute them. Take the playbook below for example,

```
---
# /tmp/playbook.yml
- hosts: all
  tasks:
    - name: Install emacs
      package:
        name: emacs-nox
        state: installed
      tags: good
    - name: Place emacs config file
      copy:
        src: .emacs
        dest: {{ ansible_env.HOME }}
      tags: evil
    - name: Install vim
      package:
        name: vim
        state: present
      tags: evil
    - name: Install nano
      package:
        name: nano
        state: installed
```

you could install emacs by running:

1. `ansible-playbook /tmp/playbook.yml --tags good`
2. `ansible-playbook /tmp/playbook.yml --skip-tags evil`

The former can be understood as "only run tasks tagged as good", hence tasks one and two will be the only ones to be executed. The latter, on the other hand, means "run every task not tagged as evil", therefore tasks one, two AND four will be executed.

Inheritance

Tags only affect tasks; this means tagging a task, role or playbook import will only serve the purpose of tagging all tasks within it.

> **Warning:** *include* statements DO NOT tag tasks within them. Multi-tagging only applies to dynamic includes, such as *import* statements.

```
---
# Tasks within taskfile_1.yml will not be tagged,
# but the include will.
- include_tasks: taskfile_1.yml
  tags: tag_1

# Tasks within taskfile_2 will be tagged, as well
# as the import statement itself.
- import_tasks: taskfile_2.yml
  tags: tag_2
```

Think of it as multi-tagging, where declaring one tag automatically tags a set of tasks. For example, the playbook below

```
---
- hosts: all
  roles:
    - role: apache
      vars:
        ssl_port: 443
      tags:
        - web
        - container
```

will tag all tasks within the role apache.

Special tags

Tag	Meaning	Explicit
always	always run a task; can be skipped using –skip-tags always.	yes
never	never run a task, unless explicitly told to do so.	yes
tagged	run tagged tasks only	no
untagged	run untagged tasks only	no
all	run all tasks (DEFAULT)	no

Example

Consider a provisioned cluster with 501 nodes (1 master, 500 slaves), where ansible's average running time is 25 to 30 minutes.

Suppose you are given the task of automating the creation of the folder `/scratch-local` and its subdirectories, so that each node has a directory per scientist, named after the convention `/scratch-local/<username>`.

In order to accomplish the task, you intend to read the usernames from the datacenter's FreeIpa manager and later use them to create the appropriate directories under `/scratch-local/`:

```
#/tmp/scratch_local.yml
---
- name: Get users from FreeIpa
  shell: ipa user-find --raw --pkey-only | awk '/uid:/{print $2}'
  register: get_users
```

(continues on next page)

(continued from previous page)

```
- name: Create dirs
  file:
    path: "/scratch-local/{{ item }}"
    state: directory
    owner: "{{ item }}"
  loop: "{{ get_users.stdout_lines }}"
```

Running the above taskfile will ensure all scientists have their own folder in the specified path. You quickly realize, however, that it will take 25 to 30 minutes for the changes to be applied and upon the creation of new user accounts at worst 30 minutes * N° of new users (if they are not created within ansible's run interval). So, you decide to tag the tasks:

```
#/tmp/playbook.yml
---
- hosts: computes
  tasks:
  - import_tasks: scratch_local.yml
    tags: scratch
```

Finally, you tell your boss incorporating your code to the git repo holding ansible data and running the command `ansible-playbook --tags scratch <playbook>` will do the job without further delay.

Deployment strategy

Ensuring the system's state remains as desired after deployment requires constant monitoring of configuration files, daemons (services), etc. Bearing this in mind, employing a self-aware architecture—in which tasks are not run once, but repeatedly and their actions are performed based on checking the system state—is a reasonable choice.

To further ease deployment one may find useful to provision a server, or set of servers, by using wrappers triggering ansible's execution, one-shot playbooks and scripts.

This section proposes an architecture integrating all the abovementioned ideas.

Overview

The proposed architecture aims to provision a system by following the steps below:

1. Create ansible's directory hierarchy, playbooks, roles, taskfiles, etc. (basically,the logic that will be used to provision the system), and maintain it under version control.
2. ssh into the master node and download the repo from version control. Consider using read-only deploy keys to download the repo without having to type an username and password; especially in unattended deployments.
3. Run a one-shot script to perform an initial setup of the environment required for ansible to run properly during and after deployment, followed by the actual software execution (see Section 3.5.1).
4. Check the bootstrap log and verify there were no errors and/or unexpected behaviors.
5. If errors arose, fix them and re-run the bootstrap script.

Provisioning logic

Directory hierarchy

The logic should consider constant creation of new playbooks, roles, taskfiles, etc., allowing to easy scale in the future. Take the example below:

```
# tree /etc/ansible
.
├── ansible.cfg
├── environments
│   ├── dev
│   │   ├── group_vars
│   │   └── inventory
│   └── prod
│       ├── group_vars
│       └── inventory
├── playbooks
│   ├── playbook_1.yml
│   ├── playbook_2.yml
│   └── playbook_n.yml
├── roles
│   ├── role_1
│   │   ├── handlers
│   │   │   └── main.yml
│   │   ├── tasks
│   │   │   └── main.yml
│   │   ├── templates
│   │   └── vars
│   │       └── main.yml
│   ├── role_2
│   │   ├── handlers
│   │   │   └── main.yml
│   │   ├── tasks
│   │   │   └── main.yml
│   │   ├── templates
│   │   └── vars
│   │       └── main.yml
│   └── role_n
│       ├── handlers
│       │   └── main.yml
│       ├── tasks
│       │   └── main.yml
│       ├── templates
│       └── vars
│           └── main.yml
├── scripts
│   ├── bootstrap.sh
│   └── vault-client.sh
└── site.yml
```

It is designed so that upon calling `site.yml` tasks from a particular set of playbooks, from the `playbooks` folder, are applied. This behavior can be accomplished by manually importing the playbooks or using variables:

Importing playbooks	Using variables

`# /etc/ansible/site.yml`
`---`
`- import_playbook: playbooks/playbook_1.yml`
`- import_playbook: playbooks/playbook_2.yml`
`- import_playbook: playbooks/playbook_n.yml`
	`# /etc/ansible/environments/prod/group_vars/group1`
	`---`
	`playbook: playbook_1`
	
	`# /etc/ansible/environments/prod/group_vars/group2`
	`---`
	`playbook: playbook_2`
	
	`# /etc/ansible/environments/prod/group_vars/groupN`
	`---`
	`playbook: playbook_n`
	
	`# /etc/ansible/site.yml`
	`---`
	`- import_playbook: "playbooks/{{ playbook }}.yml"`

One could couple all playbooks inside `site.yml`, but that would make future scalability difficult and potentially cause problems if a large number of people is working on the same project (take *git* merge conflicts for example).

If one-shot playbooks (playbooks that run only once, such as whose involving firmware updates) are to be managed, it is recommended to modify the directory hierarchy so that the `playbooks` folder holds them. For example:

```
.
└── playbooks
    ├── auto
    │   ├── playbook_1
    │   ├── playbook_2
    │   └── playbook_n
    └── manual
        ├── playbook_1
        ├── playbook_2
        └── playbook_n
```

which would be run like:

Note: More options can be used, but they will mostly depend on the playbook's functionality.

```
ansible-playbook -i /path/to/inventory \
                 /path/to/ansible/playbooks/manual/<playbook>
```

Ansible launcher

Using multiple environments, launching ansible from a non-standard location, among others may result in a large inconvinient command. Furthermore, if a run is to be triggered by an external entity, such as a script that requires ansible to run certain tasks within particular servers (see Section 3.5.1), additional concerns arise (e.g. What if I run ansible while it is already running? how to control the number of runs at a given time?).

To solve the abovementioned issues one can create a template ansible will render as a script:

```
#!/bin/bash
# Title       : run_ansible
# Description : Run ansible
# Author      : Tomas Felipe Llano Rios
# Date        : Nov 21, 2018
# Usage       : bash run_ansible [options]
# Help        : bash run_ansible -h
#===============================================================================

# tee only reads and prints from and to a file descriptor,
# so we need to use two execs to read and print from and to
# both stdout and stderr.
#
# Receives stdout, logs it and prints to stdout
exec > >(tee -ia /{{ ansible_log_dir }}/scheduled_run.log)
# Receive stderr, log it and print to stderr.
exec 2> >(tee -ia /{{ ansible_log_dir }}/scheduled_run.log >&2)

function log {
    echo "[$(date --rfc-3339=seconds)]: $*"
}

function print_help {
    echo -e "\nUsage: run_ansible [options]\n"
    echo -e "Where [options] include all ansible-playbook options,"
    echo -e "except for --vault-id and --inventory-file.\n"
    echo -e "Installed using the following configuration:"
    echo -e "\tEnvironment: $env"
    echo -e "\tAnsible home: $repo_dir"
    echo -e "\tAnsible config file: $cfg_file"
    echo -e "\tInventory file: $inv_file\n"

    command -v ansible-playbook > /dev/null 2>&1
    if [ "$?" -eq 0 ]; then
        echo -e "ansible-playbook options:\n"
        ansible-playbook -h | awk '/Options:/{y=1;next}y'
    else
        echo -e "See ansible-playbook help for more information.\n"
    fi
}

# Always release lock before exiting
function finish {
    flock -u 3
    rm -rf $lock
}
trap finish EXIT

# Create lock to prevent the script from being
# executed more than once at a given time.
declare -r script_name=`basename $0`
declare -r lock="/var/run/${script_name}"
if [ -f "$lock" ]; then
    echo "Another process (pid:`cat $lock`) is already running"
    trap - EXIT
    exit 1
```

(continues on next page)

(continued from previous page)

```
fi
exec 3>$lock
flock -n 3
log "Lock acquired"
declare -r pid="$$"
echo "$pid" 1>&3

declare -r env={{ env }}
declare -r repo_dir={{ repo_dir }}
declare -r cfg_file="$repo_dir/ansible.cfg"
declare -r inv_file="$repo_dir/environments/$env/inventory"

if [ "$1" = "-h" ] || [ "$1" = "--help" ]; then
    print_help
    exit 0
fi

log "Updating repository"
cd $repo_dir
git pull
cd -

log "Executing ansible"

export ANSIBLE_CONFIG="$cfg_file"
export DEFAULT_ROLES_PATH="$repo_dir/roles"
export ANSIBLE_EXTRA_VARS="env=$env repo_dir=$repo_dir $ANSIBLE_EXTRA_VARS"
ansible-playbook --inventory-file "$inv_file" \
                 --extra-vars "$ANSIBLE_EXTRA_VARS" \
                 --vault-id "$env@$repo_dir/scripts/vault-secrets-client.sh" \
                 $repo_dir/site.yml \
                 -vv \
                 $@
unset ANSIBLE_CONFIG
```

In order to allow for easy migration of the script to, for example, another folder while still pointing to the project sources the template obtains one variable, *ansible_log_dir*, from `group_vars` and the remaining two from the bootstrap script. This is corroborated in line 47 from Section 3.5.1, where there are two *extra vars* (env, repo_dir) passed to `ansible-playbook`; all of which are dynamically discovered just before the first run.

On subsequent runs, `run_ansible` will keep passing down these values recursively. One could argue it is better to just place the task inside a one-shot playbook; this implies, however, that modifications made to the template should be applied manually and if the rendered script is changed it would not be restored automatically.

Scheduled run

It would be tedious to manually run ansible every time a change is done to the project. A nice approach to schedule when one wants provisioning to occur is creating a task to install a cron managing the time at which to call ansible:

Hint: Given the limited environment in which crons are run, one may need to add a task such as:

```
- name: Adding PATH variable to cronfile
  cron:
    name: PATH
```

(continues on next page)

(continued from previous page)

```
    user: root
    env: yes
    value: /bin:/sbin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin
    cron_file: ansible_scheduled_run
```

or, if your launcher is written in bash, make it act as if it had been invoked as a login shell by using the `-l` option (`#!/bin/bash -l`).

```
- name: Schedule ansible to run every 30 minutes
  cron:
    name: "run ansible every 30 minutes"
    user: root
    minute: "*/30"
    job: "/path/to/run_ansible"
    cron_file: ansible_scheduled_run
```

bootstrap

By means of a few initial instructions the script should install ansible in the target system, prepare the environment it requires and trigger its first run. To further review whether the bootstrap failed or succeeded, and take apropriate actions, it is strongly recommended to log the output. Take the following program for example:

```
#!/bin/bash
# Title       : bootstrap.sh
# Description : Install and configure ansible
# Author      : Tomas Felipe Llano Rios
# Date        : Nov 21, 2018
# Usage       : bash bootstrap.sh <environment>
#================================================================================

# tee only reads and prints from and to a file descriptor,
# so we need to use two execs to read and print from and to
# both stdout and stderr.
#
# Receive stdout, log it and print to stdout.
exec > >(tee -ia ./bootstrap_run.log)
# Receive stderr, log it and print to stderr.
exec 2> >(tee -ia ./bootstrap_run.log >&2)

declare -r env="$1"
declare -r script_path="$(readlink -e $0)"
declare -r script_dir="$(dirname $script_path)"
declare -r repo_dir="${script_dir%/*}"
declare -r cfg_file="$repo_dir/ansible.cfg"
declare -r inv_file="$repo_dir/environments/$env/inventory"

# Check if the environment provided exists within ansible's
# directory hierarchy.
declare -r envs="$(ls $repo_dir/environments/)"
if [ "$envs" != *"$env"* ]; then
    echo -e "\nUnrecognized environment. Choose from:\n$envs\n"
    exit 1
fi
```

(continues on next page)

(continued from previous page)

```
# ansible-vault requires pycrypto 2.6, which is not installed by default
# on RHEL6 based systems.
declare -i centos_version=`rpm --query centos-release | awk -F'-' '{print $3}'`
if [ "$centos_version" -eq "6" ]; then
    /usr/bin/yum --enablerepo=epel -y install python-crypto2.6
fi
# Install ansible.
/usr/bin/yum --enablerepo=epel -y install ansible

# Run ansible.
export ANSIBLE_CONFIG="$cfg_file"
export DEFAULT_ROLES_PATH="$repo_dir/roles"
ansible-playbook \
    --inventory-file="$inv_file" \
    --extra-vars "env=$env repo_dir=$repo_dir" \
    --vault-id "$env@$repo_dir/scripts/vault-secrets-client.sh" \
    $repo_dir/site.yml \
    -vv
```

After running the script, there should be a cronfile in /etc/cron.d and a rendered version of the run_ansible script.

Example

1. Create directory tree

```
cd /some/dir/
mkdir -p ansible
cd ansible && git init
git remote add origin <uri>
mkdir -p {playbooks,environments,roles,scripts}
mkdir -p roles/master/{tasks,templates}
mkdir -p environments/production/group_vars/
# Create the appropriate files according to your needs.
# A good start would be:
#touch site.yml \ # Calls the master.yml playbook
#      playbooks/master.yml \ # Calls the master role
#      roles/master/tasks/main.yml \ # Renders template
#      roles/master/templates/run_ansible.j2
#      environment/production/inventory
#      environment/production/group_vars/all
```

2. Download repo

> **Note:** Consider using read-only deploy keys to download the repo without having to type an username and password; especially in unattended deployments.

```
ssh <user>@<server>
cd /usr/local/
git clone <uri>
```

3. Bootstrap. Suppose you run the bootstrap script from `/usr/local/ansible/scripts/`, which discovers and passes two variables to ansible: *env* and *repo_dir*:

```
declare -r env="$1"
declare -r script_path="$(readlink -e $0)"
declare -r script_dir="$(dirname $script_path)"
declare -r repo_dir="${script_dir%/*}"
declare -r cfg_file="$repo_dir/ansible.cfg"
declare -r inv_file="$repo_dir/environments/$env/inventory"
ansible-playbook \
    --inventory-file="$inv_file" \
    --extra-vars "env=$env repo_dir=$repo_dir" \
    --vault-id "$env@$repo_dir/scripts/vault-secrets-client.sh" \
    $repo_dir/site.yml \
    -vv
```

Executing the script in a production environment, like `bootstrap.sh prod`, will cause variables to be passed to ansible as `env=production` and `repo_dir=/usr/local/ansible/`; therefore producing a `run_ansible` script pointing to `/usr/local/ansible/`.

4. Check for errors

```
less bootstrap_run.log
```

Tips and tricks

Installation of multiple packages

Internet often indicate using anible's pseudo looping method to install multiple packages is the way to go. However, this can prove to be slow due to packages being installed one by one.

```
---
- name: Install multiple packages one by one
  yum:
    name: "{{ item }}"
    state: present
    update_cache: yes
  loop:
    - emacs
    - nano
    - vim
    - nmap
    - htop
```

An effective approach to install multiple packages in a single transaction is to provide a list of their names to the yum module:

```
---
- name: Install multiple packages in one transaction
  yum:
    name:
      - emacs
      - nano
      - vim
      - nmap
      - htop
    state: present
    update_cache: yes
```

User account creation

The *user* module ensures an user is created and their password setup properly by taking an optional *password* argument. This argument, however, is not the actual password, as most would think, but its hashed value. The hashing can be accomplished using ansible's hash-filters or other tools such as openssl or python.

Using hash-filters

```
- user:
    name: username
    password: "{{ '<password>' | password_hash('sha512', '<salt>') }}"
    shell: /usr/bin/nologin
```

Using openssl

```
openssl passwd -1 '<password>'
```

Troubleshooting

Ansible Vault

Newer pycrypto required

- **Error Message/s:**
  - "ERROR: ansible-vault requires a newer version of pycrypto than the one installed on your platform"
  - "ERROR! Vault password script path/to/script returned non-zero (1): None"
- **Reproduced on:** CentOS 6
- **Software versions affected:** ansible >= 2.4
- **How to reproduce?** Download ansible's RPM from the official website and install it. Thereupon, try using `ansible-vault` to encrypt something and the error message shall emerge.
- **Solution:** Install EPEL's ansible package. It is patched to support previous versions of python-crypto (See [Kura14]).

```
# Install epel repository
yum install epel-release
# Install pycrypto 2.6
yum install python-crypto2.6
# Install ansible from the EPEL repository
yum install --enablerepo=epel ansible
```

Playbooks

Invalid user password

- **Error Message/s:** None. The actual problem comes from a common misunderstanding of Ansible's *user* module.
- **Reproduced on:** CentOS 6
- **Software versions affected:** any ansible version

- **How to reproduce?** Create a playbook in which you use Ansible's *user* module to assign a password to any user (or just create one) and pass the password to the module's *password* argument. For example:

```
---
- hosts: all
  tasks:
  - name:
    user:
      name: root
      password: 1234
```

- **Solution:** Use the password's crypted value that would normally be placed inside `/etc/shadow`. For example:

```
# Creating an MD5 hash using openssl
openssl passwd -1
Password: 1234
Verifying - Password:
$1$PmZtHS1g$yjx.gQWWFduYPzN/j1jdY

# Creating a sha-256 hash using python 2
python -c "import random,string,crypt
randomsalt = ''.join(random.sample(string.ascii_letters,8))
print crypt.crypt('1234', '\$6\$%s\$' % randomsalt)"
$6$DivYqPSU$zWxSRQhe4ImWhKRFDAIu/PPG4Fp0LC3Cbv3n.wDHMaDsjF4ZSvjOt98j5/
↪qB7ONE3trcxtGeGgZqkYIKTKKJ1/
```

If your playbook is under version control consider using Ansible Vault to encrypt the hash either as a string or placing it inside a file and subsequentially encrypting it. If using the former, DO NOT press the return key after writing the hash, but [Ctrl] + [d] two times instead.

References

Authors

- Tomás Felipe Llano-Rios <tllanos@eafit.edu.co>

3.5.2 Vagrant

Basic information

- **Official website:** https://www.vagrantup.com/
- **Version:** 2.2.3
- **License:** MIT License

Installation

You can find the Vagrant's latest version in the link https://www.vagrantup.com/ downloads.html . Also, you can install it with your package manager (check the version before install it).

Usage

Vagrant commands usually have the following structure: `vagrant command [<args>]` but in this particular case, we have custom options that help with our ansible configuration so the structure changes to: `vagrant --custom-option=option -- command [<args>]`

Note:

- Be careful with the spaces between the double minus and the command.
- If you use a diferent configuration for vagrant remember to delete the custom option `--custom-option=option --` to use the vagrant's usual commands

---

1. Use the `up` command alongside the `--provision` argument to import the VM into VirtualBox and run ansible.

```
vagrant --machine=<single> -- up --provision
```

2. Check if ansible ran successfully. If it didn't, find the error and patch it.
3. Test your patch by syncing the changes and re-provision.

```
vagrant --machine=<single> -- rsync
vagrant --machine=<single> -- provision
```

Note: Our Vagrant's custom options are:

- machine: Machine hostname for ansible and the playbooks's name.
- vault-id: The password file for ansible.

Useful commands

- **ssh:** Connects to machine via SSH. `vagrant ssh`
- **reload:** Restarts vagrant machine, loads new Vagrantfile configuration. `vagrant reload`
- **halt:** Stops the vagrant machine. `vagrant halt`
- **destroy:** Stops and deletes all traces of the vagrant machine. `vagrant destroy`

For more help run `vagrant -h` or `vagrant <command> -h` for help on any individual command.

Vagrantfile

The primary function of the Vagrantfile is to describe the type of machine required for a project, and how to configure and provision these machines[1] . The syntax of Vagrantfile is Ruby.

This vagrantfile is based on a vagranfile with custom variables to ease the use of ansible_local handling and other options.

[1] Vagrantfile. (n.d.). Retrieved January 21, 2019, from https://www.vagrantup.com/docs/vagrantfile/

Listing 73: vagrantfile

```
# -*- mode: ruby -*-
#
# Title       : vagrantfile
# Description : A vagranfile example with some custom configuration.
# Author      : Manuela Carrasco
# Date        : Jan 21, 2019
# Usage       : vagrant [options] -- command [<args>]
# Help        : vagrant -h
#=============================================================================

# This example has custom options like "machine" but we won't show the code
# where the extra options were added.

Vagrant.configure("2") do |config| # The vagrantfile api version is 2
  # The virtual machine image. For more information
  # https://www.vagrantup.com/docs/vagrant-cloud/
  config.vm.box = "centos/7"
  # Custom configuration for virtualbox machine
  config.vm.provider :virtualbox do |vb|
    vb.customize ["modifyvm", :id,
                  "--memory", "2048"]
  end
  # Configure ansible in the VM
  config.vm.provision "ansible_local" do |ansible|
    ansible.playbook = "./site.yml"
    # Perform the playbook tasks as another user
    ansible.become = true
    # machine is a custom option for the hostname
    # and it's the playbooks name
    ansible.groups = {
      "#{machine}" => ["default"],
      "all:children" => ["#{machine}"]
    }
    # vault_id is a password file for ansible
    if not vault_id.empty?
      ansible.vault_password_file = "#{vault_id}"
    end
  end
  config.vm.hostname = "my-virtualmachine" # Set VM hostname
```

Note: For more information about vagrantfile and how to create it go to https://www.vagrantup.com/docs/vagrantfile/

Troublehooting

Disk Resize

The main purpose of this article is documenting how to face a problem with the machine storage in vagrant, and also we will see briefly how to install it.

Table of Contents

Basic information

- **Official documentation:** https://www.vagrantup.com/

Tested on (Requirements)

- **Base OS:** Fedora 31
- **OS for the virtual machine:** CentOS 8

Disk Resize Example in Vagrant

1. Create a directory where you will work with vagrant and enter into it.

```
$ mkdir Desktop/vagrant
$ cd Desktop/vagrant
```

2. Initialize the directory with Vagrant, it will generate a new file called Vagrantfile, it will be the base for vagrant.

```
$ vagrant init
```

3. Then download the plugin that will help for changing the size of our virtual machine

```
$ vagrant plugin install vagrant-disksize
```

4. Enter to the file and edit it with the options you want to your Vagrant machine such as hostname, network interfaces, bootstrap script, vagrant box, etc. (You can read more about boxes at[1]). For example:

```
BOX_IMAGE = "bento/centos-8.1"
NODE_COUNT = 2

Vagrant.configure("2") do |config|
  config.vm.box = BOX_IMAGE
  config.disksize.size = '50GB' #the plugin we've previously installed
  config.vm.define "controller" do |ctrlr|
    ctrlr.vm.hostname = "ctrlr.test.lan"
    ctrlr.vm.network "private_network", ip: "192.168.50.2"
    ctrlr.vm.provider "virtualbox" do |v|
      v.name = "Controller"
    end
```

(continues on next page)

[1] Vagrant documentation. Retrieved from https://www.vagrantup.com/intro/getting-started/boxes.html

(continued from previous page)

```
  end
end
```

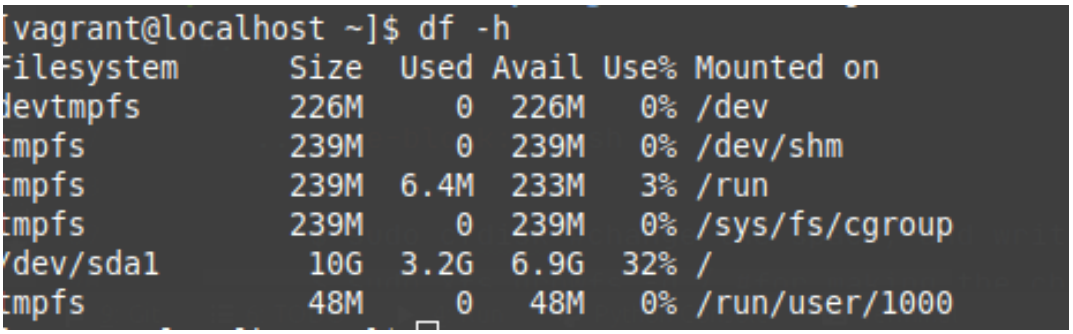
5. In the same folder where the Vagrant file is located run the following commands:

```
$ vagrant up
```

6. Go to the new machine using ssh.

```
$ vagrant ssh
```

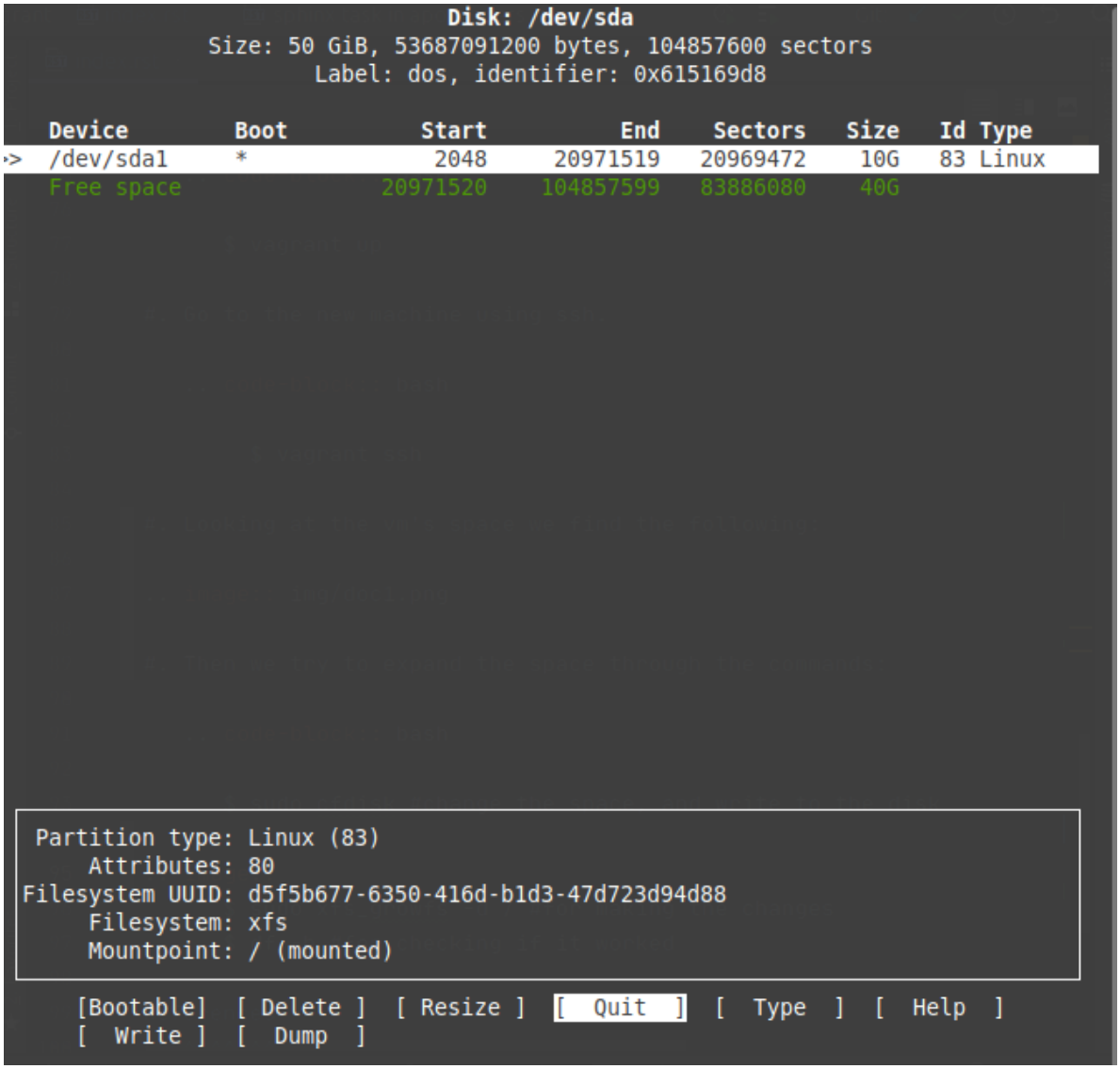
7. Looking at the vm's space we find out the following:



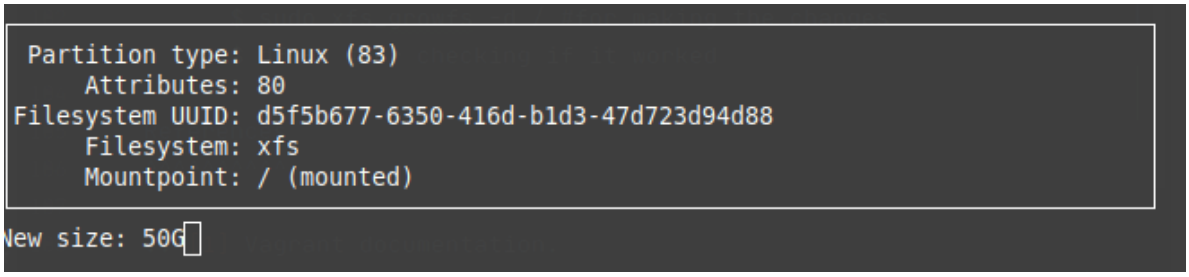
8. Then we try to expand the space through the commands:

```
$ sudo cfdisk #change the space, and write to the disk
```

- It should look like this:



- Here we have to click on the **resize** option and it should look like the following:



- After that it will ask for verification, just type "yes" and then click on **write**

9. Then enter and exit, after that process run the following commands:

```
$ sudo xfs_growfs -d / #for making the changes
$ df -h #for checking if it worked
```

References

Author Manuela Herrera-López <mherreral@eafit.edu.co>

Author

- Manuela Carrasco Pinzón <mcarras1@eafit.edu.co>
- Tomás Felipe Llano Rios <tllanos@eafit.edu.co>

References

3.6 Monitoring

This entry contains the configuration, installation and maintainance procedures needed to configure some monitoring systems used in the Scientific Computing Center APOLO.

3.6.1 Monitoring Architecture

Contents

Basic information

- **Deploy date:** 29th July, 2019

Sensu-Graphite-Grafana

Components

Monitoring Core: Provides a scalable event processing platform for processing check results and monitoring events.[1]

- **Sensu Server:** Schedules and publishes check execution requests to client subscriptions.[1]
- **Sensu API:** Provides access to monitoring data collected by Sensu through a RESTful API endpoint.
- **Redis:** In-memory Key-value data structure store, used by Sensu to store monitoring data.

[1] Sensu Server: Sensu Core 1.5. (n.d.). Retrieved July 30, 2019, from https://docs.sensu.io/sensu-core/1.5/reference/server/

Sensu Clients: This layer is composed of the agents that need to be monitored.

Communication: Sensu Services use a message bus for communication, sensu check requests and check results are published as messages to this layer, and the corresponding sensu services receive these messages by subscribing to the appropriate subscriptions[2].

> **Warning:** Using Redis as a transport greatly simplifies Sensu's architecture, however, Redis transport is NOT recommended for production environments.[3]

- **RabbitMQ:** Message Broker that communicates Sensu Clients with the Server.

Store Time-series data: Sensu doesn't store time-series natively, it's necessary to integrate it with Graphite to have historic registers of the results.

Carbon refers to various daemons that make up the storage backend of a Graphite installation[4]. This architecture only needs to use Carbon Cache, but later, Carbon Relay and Carbon aggregator can be added without problems.

- **Carbon Cache:** listens for metrics and writes them to disk as efficiently as possible. This requires caching metric values in RAM as they are received and flushing them to disk on an interval using the underlying whisper library. It also provides a query service for in-memory metric datapoints, used by the Graphite web app to retrieve "hot data".[4]
- **Graphite Web:** Graphite's user interface and RESTful API endpoint.
- **Whisper:** Fixed-size database, similar in design and purpose to RRD (round-robin-database). Whisper allows for higher resolution (seconds per point) of recent data to degrade into lower resolutions for long-term retention of historical data.

Visualization: Displays the configuration, current state and monitoring results.

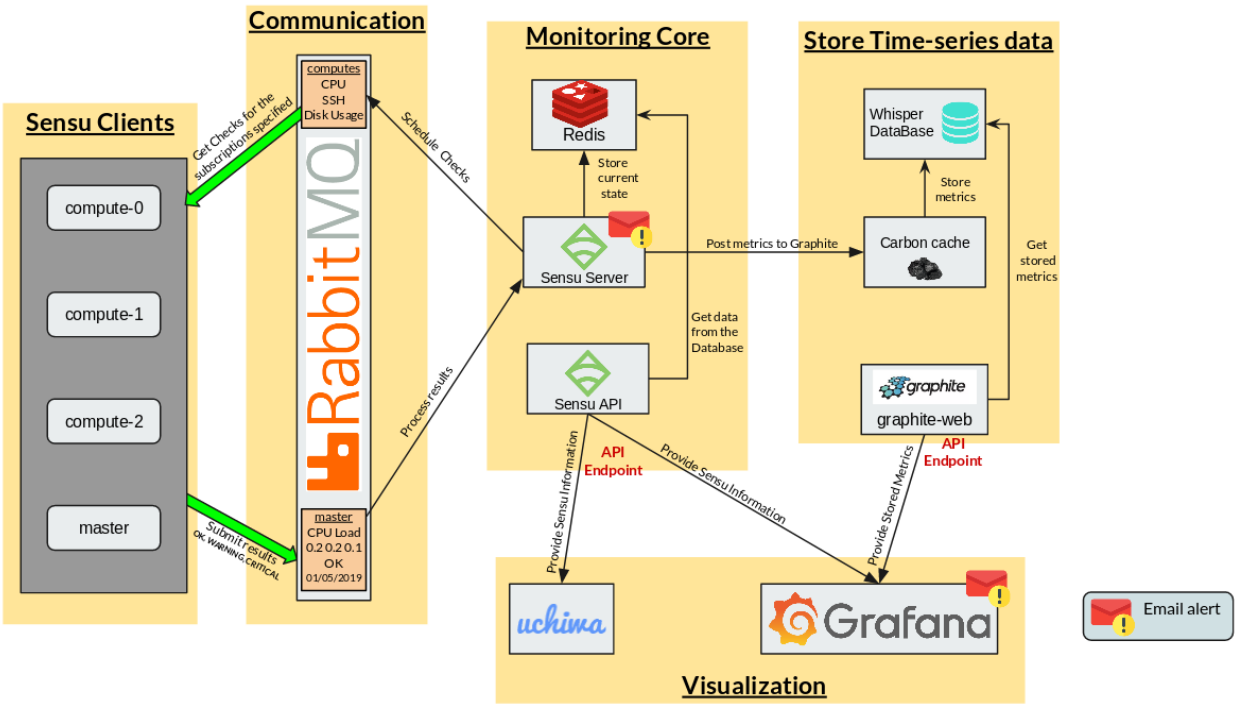
- **Uchiwa:** Web Dashboard for visualizing Sensu status and Configuration.
- **Grafana:** Shows time-series and monitoring results in dashboards and manage email alerts with established thresholds.

[2] Transport: Sensu Core 1.5. (n.d.). Retrieved July 30, 2019, from https://docs.sensu.io/sensu-core/1.5/reference/transport/

[3] Redis Configuration: Sensu Core 1.5. (n.d.). Retrieved July 30, 2019, from https://docs.sensu.io/sensu-core/1.5/reference/redis/#how-does-sensu-use-redis

[4] The Carbon Daemons. (n.d.). Retrieved July 30, 2019, from https://graphite.readthedocs.io/en/latest/carbon-daemons.html

WorkFlow



1. Sensu Server schedules the checks, posting pending checks in RabbitMQ for each subscription.
2. Each Sensu Client ask for the checks related to its subscriptions in RabbitMQ
3. After executing the Check, the clients post the results in RabbitMQ
4. Sensu Server processes the results and executes the respective handlers and mutators if necessary.
5. The last results are stored in Redis, registering ONLY the current state of the checks.
6. Sensu API provides that current state (configuration and events) to Grafana and Uchiwa.
7. If the check is configured, Sensu can report a bad execution or an unexpected situation sending an email.
8. The checks that have to be stored as TimeSeries pass their results to Carbon Cache.
9. Carbon Cache stores the results until the buffer is full, in this case, it flushes the results calling the Whisper Library and storing it as efficiently as possible.
10. Graphite Web provides this stored information through its API Rest endpoint.
11. Grafana takes the results stored in Graphite and creates the desired dashboard.
12. If a threshold is established and an alarm configured in Grafana, it can report an anomaly sending an email.

Authors

- Andrés Felipe Zapata Palacio <azapat47@eafit.edu.co>

References

3.6.2 Nagios

Nagios[1] monitors your entire IT infrastructure to ensure systems, applications, services, and business processes are functioning properly. In the event of a failure, Nagios can alert technical staff of the problem, allowing them to begin remediation processes before outages affect business processes, end-users, or customers.

This Tool is used in the computing center to monitor IPMI sensors, Fans, CPU usage, Temperature status, using the BMC interface provided by the servers.

Nagios Core - 4.4.1

Contents

Basic information

- **Deploy date:** 14th August, 2018
- **Official Website:** https://www.nagios.org/about/
- **License:** Nagios Open Software License, Nagios Software License, GNU GENERAL PUBLIC LICENSE

Directory Hierarchy

- **bin:** Nagios binaries.
- **dell:** Stores scripts, configuration files, images and resources of Dell Plugin.
- **etc:** Stores Nagios configuration files.
- **include**
- **libexec:** Contains most of the plugins installed.

[1] Nagios. Website Copyright © 2009-2018 Nagios Enterprises, LLC. All rights reserved. Retrieved 13:50, August 14th, 2018, from https://www.nagios.org/about/

- **sbin:** Nagios scripts
- **share:** Contains Web interface files.
- **var**

Installation

This entry covers the entire process performed for the installation and configuration of Nagios Core in Centos 7. This process of installation and configuration is automated using Ansible.

Installation

Table of Contents

Tested on (Requirements)

- **OS base:** CentOS 7 (x86_64)
- **Provisioner:** Ansible $\geq$ 4.4.1
- **Extra Libraries:**
  - PyCrypto 2.6.1

Note: It is important to check if the PyCrypto system version is greater than 2.6.1, because this is a pre-requisite that Ansible-vault needs to work correctly.

Directory Structure

```
.
├── ansible.cfg
├── inventory
│   └── hosts
├── playbooks
│   ├── healthcheck.yml
└── roles
    └── healthcheck
        ├── handlers
        ├── tasks
        ├── vars
        ├── templates
        └── files
```

Ansible Structure

We implemented a Role in Ansible that contains the whole process of installation and configuration of Nagios and it's integration with some plugins.

```
---
#################################################################
#                    INITIAL-CONFIGURATION                      #
#################################################################
- include_tasks: dell-repos.yml
- include_tasks: packages.yml
- include_tasks: nagios-users.yml
- include_tasks: apache-config.yml
- include_tasks: firewall-config.yml
- include_tasks: ipmi-config.yml
- include_tasks: mail-config.yml
- include_tasks: snmp-config.yml

#################################################################
#                          NAGIOS-CORE                          #
#################################################################
- include_tasks: nagios-installed.yml
- include_tasks: nagios-install.yml
  when: nagios_installed.stat.exists == false
```

(continues on next page)

(continued from previous page)

```
- include_tasks: nagios-config.yml
- include_tasks: nagios-post-install.yml
  when: nagios_installed.stat.exists == false
- include_tasks: selinux-config.yml

#################################################################
#                        NAGIOS-PLUGINS                         #
#################################################################
- include_tasks: nagios-plugins-installed.yml

- include_tasks: nagios-plugins.yml
  when: nagios_plugins_installed.matched == 0

- include_tasks: dell-plugin.yml
  when: dell_plugin_dir.stat.exists == false
- include_tasks: dell-plugin-config.yml

- include_tasks: ipmi-sensors-plugin.yml
  when: ipmi_sensor_plugin.stat.exists == false
- include_tasks: ipmi-plugin-status.yml

- include_tasks: ilo-plugin.yml
  when: ilo_plugins.matched == 0
- include_tasks: ilo-plugin-config.yml

- include_tasks: pnp4nagios-install.yml
  when: pnp_dir.stat.exists == false
- include_tasks: pnp4nagios-config.yml

#################################################################
#             CHECK-CORRECT-CONFIG & REMOVE TEMP DIR            #
#################################################################
- include_tasks: final-check.yml
```

Initial Configuration

dell-repos.yml

This procedure is necessary in order to install the package **srvadmin-idrac7** from the official Dell repo. This makes it easier to check the presence/absence of the packages using the ansible-module "yum" instead of writing manually the process of compilation and verification.

```
---
- name: Setup Dell Linux Independent Repository
  yum_repository:
    name: dell-system-update_independent
    state: present
    baseurl: "{{ baseurl_dell_independent_repo }}"
    description: dell-system-update_independent
    gpgkey: "{{ gpgkey_dell_independent_repo }}"

- name: Setup Dell Linux Dependent Repository
  yum_repository:
    name: dell-system-update_dependent
```

(continues on next page)

(continued from previous page)

```
    state: present
    baseurl: "{{ baseurl_dell_dependent_repo }}"
    description: dell-system-update_dependent
    gpgkey: "{{ gpgkey_dell_dependent_repo }}"
```

packages.yml

This taskfile contains the dependencies for using some Ansible modules and for installing Nagios core and it's plugins.

System Packages	Description
Python-passlib	Dependency of Ansible HTPASSWD Module
Python2-pip	PIP Installs OMSDK (Dependency of Dell Plugin)
LibSELinux-Python	Dependency of Ansible SELinux Module
PolicyCoreUtils-Python	Dependency of Ansible SELinux Module
mailx	Provides "mail" command, used in notify nagios commands
ipmiutil	Necessary for *IPMI Status: Critical [X system event log (SEL) entries present]*

The other dependencies are listed in the taskfile showed bellow.

Note: This solution uses a list of packages in the yum ansible module instead of an ansible iterator (item) because this specification improves the install operation, creating a complete dependency tree instead of calling "n times" the yum module.

Note: The @ syntax in yum module specifies the item is a package group.

Note: The Dell OpenManage Plugin has two lists of dependencies: The first one is installed with the "yum" module and the second one with the "pip" module.

```
---
- name: System Packages
  yum:
    name:
    - python-passlib      #Dependency for htpasswd ansible module
    - python2-pip         #PIP Installs OMSDK required for Dell plugin
    - libselinux-python          #SELinux Ansible module dependency
    - policycoreutils-python     #SELinux Ansible module dependency
    - mailx
    - ipmiutil
    state: present
    update_cache: yes

- name: NAGIOS Dependencies
  yum:
    name:
    - httpd
    - php
    - glibc
```

(continues on next page)

(continued from previous page)

```
    - gcc
    - glibc-common
    - gd
    - gd-devel
    - make
    - net-snmp
    - "@development"
    state: present
  tags: centos-7,nagios,packages

- name: IPMI_Sensor Monitoring Plugin Dependencies
  yum:
    name:
    - freeipmi
    - perl
    - perl-IPC-Run
    state: present
    update_cache: yes

- name: Dependencies for iLO REST Plugin
  yum:
    name:
    - curl
    - libcurl
    - libcurl-devel
    - nmap
    - libtdb-devel
    - python
    - openssl-devel
    state: present
    update_cache: yes

- name: Dependencies for Dell OME Plugin
  yum:
    name:
    - perl-Sys-Syslog      # SNMPTT Dependency
    - perl-Net-IP
    - perl-Net-SNMP
    - libwsman1
    - openwsman-perl
    - perl-Socket6
    - snmptt
    - net-snmp-perl
    - srvadmin-idrac7
    - java-1.8.0-openjdk
    - java-1.8.0-openjdk-devel
    - python-netaddr
    state: present
    update_cache: yes

- name: Python dependency for Dell OME Plugin
  pip:
    name:
    - omsdk
    - omdrivers
    - argparse
    state: present
```

(continues on next page)

(continued from previous page)

```
- name: Dependencies for PNP4Nagios
  yum:
    name:
    - rrdtool
    - php
    - perl
    - rrdtool-perl
    - php-gd
    state: present
```

nagios-users.yml

It is necessary before installing Nagios-Core to create a **Nagios** user, and a **nagcmd** group, whose members will be **apache** and **nagios** users. It's also necessary to let nagios execute /usr/sbin/ipmi-sensors and /usr/sbin/ipmi-sel with root permissions. This is assured making it explicit in the sudoers file.

```
---
- name: Group nagcmd needed by NAGIOS
  group:
    name: nagcmd
    state: present

- name: User nagios
  user:
    name: nagios
    groups: nagcmd
    password: "{{ nagios_passwd }}"

- name: User apache in group nagcmd
  user:
    name: apache
    groups: nagcmd

- name: SUDO permissions for ipmi execution
  lineinfile:
    path: /etc/sudoers
    regexp: '^nagios\s'
    line: 'nagios ALL=(root) NOPASSWD: /usr/sbin/ipmi-sensors, /usr/sbin/ipmi-sel'
    state: present
```

apache-config.yml

The objective is to configure Nagios to provide a Web interface, so it's necessary to write in the **httpd.conf** file the line **Listen <IP>:80**. In this generic installation, we will insert **Listen 80**, allowing every network interface to provide this service.

Finally, we will associate in **/etc/hosts** our **nagios_ip** with the **ServerName** set previously.

```
---
- name: Apache necessary line in httpd.conf
  lineinfile:
    path: /etc/httpd/conf/httpd.conf
```

(continues on next page)

(continued from previous page)

```
    line: "Listen 80"
  notify: apache_restart

- name: Define Host
  lineinfile:
    path: /etc/hosts
    line: "{{ nagios_ip }} {{ health_server_name }}"
```

firewall-config.yml

Note: It's important to remember that Firewalld is the firewall of the system in CentOS 7.

We will need to allow HTTP port in the firewall configuration. The SNMP ports (161-162) should be allowed for the correct operation of iLO REST Plugin. We decided to allow these firewall requirements in the **public** zone.

```
---
- name: Allow requests throw port 80
  firewalld:
    zone: public
    service: http
    state: enabled
    permanent: true
  notify:
  - apache_restart
  - firewalld_restart

- name: Allow SNMP ports for iLO REST Plugin
  firewalld:
    zone: public
    port: 161-162/udp
    state: enabled
    permanent: true
  notify:
  - firewalld_restart
```

ipmi-config.yml

Assures the existence of ipmi-config directory and synchronizes the ipmi.cfg file with **root** as owner, **nagcmd** as Group owner and permissions 640: read and write for Owner and read-only for group members. If the final state of the task is **changed**, Nagios daemon is restarted.

```
---
- name: Assures existence of ipmi-config Directory
  file:
    path: /etc/ipmi-config/
    state: directory

- name: Syncronize IPMI configuration
  template:
    src: "etc/ipmi-config/{{ item }}.j2"
    dest: "/etc/ipmi-config/{{ item }}"
```

(continues on next page)

(continued from previous page)

```
    owner: root
    group: nagcmd
    mode: 0640
  with_items:
    - ipmi-ilo.cfg
    - ipmi-dell.cfg
  notify:
    - nagios_restart
```

mail-config.yml

Synchronizes the mail configuration file with the version located in the repository.

Warning: Read the section *Mail Configuration* for more details.

```
---
- name: Synchronizes the mail configuration
  copy:
    src: etc/mail.rc
    dest: /etc/mail.rc
```

snmp-config.yml

The Dell plugin requires this previous SNMP configuration, read the section *Preconfiguration* for more details.

Synchronizes **/etc/snmp/snmptt.ini** and **/etc/snmp/snmptrapd.conf** snmp configuration files, with the version located in the repository. If there is a modification, snmptt and snmptrapd services are restarted. After that, those services are enabled in boot time if they were not enabled.

```
---
- name: Synchronize Nagios config files
  copy:
    src: "{{ item }}"
    dest: "/{{ item }}"
  with_items:
    - etc/snmp/snmptt.ini
    - etc/snmp/snmptrapd.conf
  notify:
    - snmptt_restart
    - snmptrapd_restart

- name: SNMP services enabled in boot time
  service:
    name: "{{ item }}"
    enabled: yes
  with_items:
    - snmptt
    - snmptrapd
```

Installing Nagios Core

nagios-install.yml and nagios-installed.yml

This taskfile is included only when the path /usr/local/nagios doesn't exist. This state is registered in nagios-installed.yml, with the module **stat**.

```
---
- name: Check if NAGIOS is installed
  stat:
    path: /usr/local/nagios
  register: nagios_installed
```

Nagios Core is downloaded from `{{ nagios_core_url }}` and stored in `{{ temp_dir }}`, then it is configured with **nagcmd** as the command group, and openssl enabled. Then, the MakeFile is executed as follows[1]:

Make options used	Descriptions
make all	.
make install	Install main program, CGI's and HTML files
make install-init	Install the init script
make install-commandmode	Install and configures permissions for holding external command file
make install-config	Generates templates for initial configuration

Note: The directive `make install-webconf` is executed in *nagios-post-install.yml*

```
---
- name: Create Temp Dir
  file:
    path: "{{ temp_dir }}"
    state: directory

- name: Download Nagios Core
  get_url:
    url: "{{ nagios_core_url }}"
    dest: "{{ temp_dir }}"

- name: Extract Nagios
  unarchive:
    src: "{{ temp_dir }}/nagios-4.4.1.tar.gz"
    dest: "{{ temp_dir }}"

- name: Exec configure
  shell: "./configure --build=x86_64-redhat-linux --with-command-group=nagcmd --with-
↪openssl"
  args:
    chdir: "{{ temp_dir }}/nagios-4.4.1"

- name: Make all
  make:
    chdir: "{{ temp_dir }}/nagios-4.4.1"
    target: all
```

(continues on next page)

[1] NagiosEnterprises/nagioscore. Retrieved August 17, 2018, from https://github.com/NagiosEnterprises/nagioscore

(continued from previous page)

```
- name: Make install
  make:
     chdir: "{{ temp_dir }}/nagios-4.4.1"
     target: install

- name: Install Nagios-init scripts
  make:
    chdir: "{{ temp_dir }}/nagios-4.4.1"
    target: install-init

- name: Install Nagios Command-mode
  make:
    chdir: "{{ temp_dir }}/nagios-4.4.1"
    target: install-commandmode

- name: Generates Templates for Nagios configure Files
  make:
    chdir: "{{ temp_dir }}/nagios-4.4.1"
    target: install-config
```

nagios-config.yml

This taskfile synchronize the Nagios config files with the ones stored in the repository, if there is a change in this synchronization, Nagios daemon is restarted with the handler `nagios_restart`.

Then, the module **htpasswd** assigns the password stored with Ansible Vault in the variable `{{ nagios_admin_passwd }}` using ldap_sha1 as crypt scheme and restarts Nagios daemon if the final state of the task is **changed**.

```
---
- name: Synchronize Nagios config files
  copy:
    src: "{{ item }}"
    dest: "/{{ item }}"
  with_items:
    - usr/local/nagios/etc/objects/contacts.cfg
    - usr/local/nagios/etc/objects/localhost.cfg
    - usr/local/nagios/etc/objects/commands.cfg
    - usr/local/nagios/etc/nagios.cfg
    - usr/local/nagios/etc/objects/common.cfg
    - usr/local/nagios/etc/objects/timeperiods.cfg
    - usr/local/nagios/etc/objects/common_services.cfg
  notify:
    - nagios_restart

- name: Assing password for the Web GUI user nagiosadmin
  htpasswd:
    path: /usr/local/nagios/etc/htpasswd.users
    name: nagiosadmin
    password: "{{ nagios_admin_passwd }}"
    crypt_scheme: ldap_sha1
  notify:
    - nagios_restart
```

nagios-post-install.yml

After **nagios-config.yml** is completed, `make install-webconf` is executed, generating the Apache config file for Nagios Web Interface. This step is executed only if Nagios Core was not installed before the current execution.

```
---
- name: Make install-webconf
  make:
    chdir: "{{ temp_dir }}/nagios-4.4.1"
    target: install-webconf
```

selinux-config.yml

Note: By default, the files under the directory **/usr/local/nagios/var/rw** don't belongs to the **httpd_sys_rw_content_t** context. It is necessary to add these contexts (this is what the taskfile does) because the way Web interface interacts with Nagios is with the Command File **/usr/local/nagios/var/rw/nagios.cmd** executing from **/usr/local/nagios/sbin/**

Warning: The error `Could not stat() command file /usr/local/nagios/var/rw/nagios.cmd` is fixed by this taskfile. The explanation is in the *Nagios Command Error* section.

It's necessary to execute `restorecon -r <directory>` in order to restart the SELinux configuration over these directories. This is executed by a handler in the Ansible role.

```
---
- name: Allow apache to modify nagios command file
  sefcontext:
    target: '/usr/local/nagios/var/rw(/.*)?'
    setype: httpd_sys_rw_content_t
    state: present
  notify: nagios_cmd_selinux_sync

- name: Allow apache to execute in /usr/local/nagios/sbin
  sefcontext:
    target: '/usr/local/nagios/sbin'
    setype: httpd_sys_script_exec_t
    state: present
  notify: nagios_sbin_selinux_sync
```

Installing Nagios Plugins

The taskfile `nagios-plugins-installed.yml` registers in ansible variables if the plugins are installed or not.

```
---
- name: Checks if NAGIOS basic plugins are installed
  find:
    paths: /usr/local/nagios/libexec
  register: nagios_plugins_installed

- name: Checks if IPMI-Sensor plugin is installed
```

(continues on next page)

(continued from previous page)

```
  stat:
    path: /usr/local/nagios/libexec/check_ipmi_sensor
  register: ipmi_sensor_plugin

- name: Checks if iLO REST plugin is installed
  find:
    paths: /usr/local/nagios/libexec
    patterns: "*hpeilo*"
  register: ilo_plugins

- name: Checks if Dell OME plugin is installed
  stat:
    path: /usr/local/nagios/dell
  register: dell_plugin_dir

- name: Checks if PNP4Nagios is installed
  stat:
    path: /usr/local/pnp4nagios
  register: pnp_dir
```

Read the following sections for more information about the installation and configuration process of the plugins.

- *Nagios Plugins*
- *IPMI Sensors*
- *Dell OpenManage*
- *iLO AgentLess Management Plugin*
- *PNP4Nagios*

Final Check

The final steps include removing `{{ temp_dir }}` and checking the Nagios configuration with the command `/usr/local/nagios/bin/nagios -v /usr/local/nagios/etc/nagios.cfg`.

This execution finishes assuring with handlers that nagios and apache services are started and enabled to start in boot time.

```
---
- name: Check integrity in Nagios configuration
  shell: "/usr/local/nagios/bin/nagios -v /usr/local/nagios/etc/nagios.cfg"
  notify:
    - nagios_started
    - apache_started
    - nagios_enabled
    - apache_enabled

- name: Remove temp dir
  file:
    path: "{{ temp_dir }}"
    state: absent
```

References

Authors

Andrés Felipe Zapata Palacio

Mail Configuration

Note: This configuration is automatically set up in the step of synchronization in the taskfile *nagios-config.yml* if the base configuration is included in the synchronization list.

The commands `notify-host-by-email` and `notify-service-by-email` were modified adding the flag -A:

```
/usr/bin/printf "MAIL_BODY" | /bin/mail -A nagios -s "SUBJECT" $CONTACTEMAIL$
```

In order to uncouple the mail definition from the command line, the flag `-A nagios` was added. With this option, **mail** will use the configuration defined in the account nagios, in the file `/etc/mail.rc`

Example:

```
account nagios {
set smtp=smtp.server.hostname:port
set sender=nagios@mail
set from="nagios@mail"
set ssl-verify=ignore
}
```

Plugins

Nagios-plugins

It is a set of useful plugins that are needed by other plugins like Dell OpenManage plugin.

Basic information

- **Official Website:** https://nagios-plugins.org/
- **License:** GNU General Public License v3.0
- **Description:** The official Nagios Plugins package contains over 50 plugins to get you started monitoring all the basics.[1]

Tested on (Requirements)

- **Nagios Core:** Version $\geq$ 3.5.0

[1] Nagios Plugins. (n.d.). Retrieved August 15, 2018, from https://www.nagios.org/downloads/nagios-plugins/

Installation

This taskfile is executed only if the folder `/usr/local/nagios/libexec` is not empty. This state is registered in the taskfile `nagios-plugins-installed.yml`, with the module **find**.

For more information about these registers read the section *Installing Nagios Plugins*.

The installation process consists of downloading, uncompressing, configuring and compiling the plugin.

```
---
- name: Assure existence of tempDir
  file:
    path: "{{ temp_dir }}"
    state: directory

- name: Download Nagios Plugins
  get_url:
    url: "{{ nagios_plugins_url }}"
    dest: "{{ temp_dir }}"

- name: Extract Nagios Plugins
  unarchive:
    src: "{{ temp_dir }}/nagios-plugins-2.2.1.tar.gz"
    dest: "{{ temp_dir }}"

- name: Exec configure
  shell: "./configure --with-nagios-user=nagios --with-command-group=nagcmd"
  args:
    chdir: "{{ temp_dir }}/nagios-plugins-2.2.1"

- name: Make
  make:
     chdir: "{{ temp_dir }}/nagios-plugins-2.2.1"

- name: Make install
  make:
     chdir: "{{ temp_dir }}/nagios-plugins-2.2.1"
     target: install
  notify:
    - apache_restart
```

References

IPMI Sensors - Nagios Plugin

Basic information

- **Official Website:** https://exchange.nagios.org/directory/Plugins/Hardware/Server-Hardware/IPMI-Sensor-Monitoring-Plugin/details
- **License:** GPL
- **Version:** 3.9

Installation

```
- include_tasks: ipmi-sensors-plugin.yml
  when: ipmi_sensor_plugin.stat.exists == false
- include_tasks: ipmi-plugin-status.yml
```

This taskfile is included only when the file `/usr/local/nagios/libexec/check_ipmi_sensor` doesn't exist. This state is registered in `nagios-plugins-installed.yml`, with the module **stat**.

More information in the section *Installing Nagios Plugins*.

The installation process consists on cloning the git repository and copying the file `check_ipmi_sensor` to the Nagios plugins directory.

```
---
- name: Assure existence of tempDir
  file:
    path: "{{ temp_dir }}/check_ipmi_sensor_v3"
    state: directory

- name: Clone git repo in temp dir
  git:
    repo: "{{ ipmi_plugin_url }}"
    clone: yes
    dest: "{{ temp_dir }}/check_ipmi_sensor_v3"

- name: Copy the IPMI sensors plugin
  copy:
    src: "{{ temp_dir }}/check_ipmi_sensor_v3/check_ipmi_sensor"
    dest: /usr/local/nagios/libexec/check_ipmi_sensor
```

After installing or not the plugin, the taskfile `ipmi-plugin-status.yml` is executed, checking the owner, the group and the permissions over the plugin.

```
---
- name: Correct setup of the executable
  file:
    path: /usr/local/nagios/libexec/check_ipmi_sensor
    owner: nagios
    group: nagcmd
    mode: 0550
```

Configuration

```
---
- name: Assures existence of ipmi-config Directory
  file:
    path: /etc/ipmi-config/
    state: directory

- name: Syncronize IPMI configuration
  template:
    src: "etc/ipmi-config/{{ item }}.j2"
    dest: "/etc/ipmi-config/{{ item }}"
    owner: root
```

(continues on next page)

(continued from previous page)

```
    group: nagcmd
    mode: 0640
  with_items:
    - ipmi-ilo.cfg
    - ipmi-dell.cfg
  notify:
    - nagios_restart
```

Synchronizes the ipmi-config file with the version present in the repo. The passwords are cyphered with **Ansible Vault**.

Usage

The following steps are required for setting up this plugin in a specific host:

1. Add the attribute **_ipmi_ip** in the host definition. This attribute is required by the check_ipmi_sensors plugin. The attribute `_ipmi_excluded_sensors` is necessary only when the error *IPMI Status: Critical [Presence = Critical, Presence = Critical]* occurs.

```
define host{
  host_name                host_1
      address                  192.168.1.1
      _ipmi_ip                 192.168.1.1
      _ipmi_excluded_sensors   56
```

Note: The names of these variables start with an underscore and are in lowercase. More info about the usage of custom object variables[1] .

---

2. Add the command definition. In this implementation, the command is added in `/usr/local/nagios/etc/objects/commands.cfg`

```
define command {
  command_name  check_ipmi_sensor
  command_line  $USER1$/check_ipmi_sensor -H $_HOSTIPMI_IP$ -f $ARG1$ -x $_
↪HOSTIPMI_EXCLUDED_SENSORS$ $ARG2$ $ARG3$
}
```

3. Add the service definition. In this implementation, the service is added in `/usr/local/nagios/etc/objects/common-services.cfg`

Note: If you want to ignore the SEL log entries warning, add the flag –nosel in the check_command field *(See example below)*

The plugin can be configured for checking each sensor type independently:

```
define service{
  use                  generic-service
  host_name            host1
  service_description  IPMI
```

(continues on next page)

[1] Custom Object Variables. (n.d.). Retrieved August 29, 2018,from https://assets.nagios.com/downloads/nagioscore/docs/nagioscore/3/en/customobjectvars.html

(continued from previous page)

```
  check_command        check_ipmi_sensor!/etc/ipmi-config/ipmi.cfg!--nosel!-T
↪<sensor_type>
}
```

Note: The sensor types are listed in the page: IPMI Sensor Types[2]

Or configured for checking everything in one Service definition:

```
define service{
  use                  generic-service
  host_name            host1
  service_description  IPMI
  check_command        check_ipmi_sensor!/etc/ipmi-config/ipmi.cfg
}
```

Note: If the IPMI plugin is configured for multiple nodes and there is not a common user/password between them, you can configure one service per each different credential, defining different ipmi-config files.

```
define service{
  use                  generic-service
  host_name            host1
  service_description  IPMI
  check_command        check_ipmi_sensor!/etc/ipmi-config/file1.cfg
}

    define service{
  use                  generic-service
  host_name            host2
  service_description  IPMI
  check_command        check_ipmi_sensor!/etc/ipmi-config/file2.cfg
}
```

Note: The user used for this IPMI monitoring doesn't need special permissions.

4. Create the file with the credentials and with the correct permissions.

```
username user
password passw0rd
privilege-level user
ipmi-sensors-interpret-oem-data on
```

- Owner: nagios
- Group: nagcmd
- Mode: 0640

Note: Read[3] for more information about freeIPMI configuration file.

[2] Krenn, T. (n.d.). IPMI Sensor Types. Retrieved November 20, 2018, from https://www.thomas-krenn.com/en/wiki/IPMI_Sensor_Types
[3] "freeipmi.conf(5) - Linux man page", FreeIPMI Core Team. Retrieved December 3, 2018, from https://linux.die.net/man/5/freeipmi.conf

Troubleshooting

IPMI Status: Critical [X system event log (SEL) entries present]

1. Read System Entry Logs before deleting them. It's important to see if there is a bad behavior registered in these logs.

   ```
   ipmiutil sel -N (host_ip|hostname) -F lan2 -U user -P passwd
   ```

2. Clear System Entry Logs with the credentials of a user with enough privileges. `ipmiutil sel -d -N (host_ip|hostname) -F lan2 -U user -P passwd`

Note: The password should be written between apostrophes (') if contains special characters.

IPMI Status: Critical [Presence = Critical, Presence = Critical]

1. Execute the following command to identify which sensors are absent.

   ```
   check_ipmi_sensor -H <Host-IP> -f <Archivo de configuración> -vvv | grep Critical
   ```

   Example of STOUT:

```
ID  | Name     | Type            | State    | Reading | Units | Lower NR  | Lower␣
↪C | Lower NC | Upper NC | Upper C | Upper NR | Event
56  | Presence | Entity Presence | Critical |   N/A   |  N/A  |    N/A    |   N/A␣
↪  |    N/A   |    N/A   |   N/A   |    N/A   | 'Entity Absent'
58  | Presence | Entity Presence | Critical |   N/A   |  N/A  |    N/A    |   N/A␣
↪  |    N/A   |    N/A   |   N/A   |    N/A   | 'Entity Absent'
IPMI Status: Critical [Presence = Critical ('Entity Absent'), Presence = Critical␣
↪('Entity Absent')] | 'Inlet Temp'=17.00;3.00:42.00;-7.00:47.00 'CPU Usage'=100.
↪00;~:101.00; 'IO Usage'=0.00;~:101.00;
'MEM Usage'=0.00;~:101.00; 'SYS Usage'=100.00;~:101.00; 'Pwr Consumption'=320.00;~
↪:452.00;~:540.00 'Current'=1.50 'Temp'=80.00 'Temp'=65.00
Presence = 'Entity Absent' (Status: Critical)
Presence = 'Entity Absent' (Status: Critical)
```

2. Add the attribute `_ipmi_excluded_sensors` which value is a comma-separated list of sensor IDs that contain the absent sensors discovered.

   Example:

```
define host{
  host_name               host-example
  address                 0.0.0.0
  _ipmi_ip                0.0.0.0
  _ipmi_excluded_sensors  56,58
}
```

References

Dell OpenManage - Nagios Plugin

Basic information

- **Official Website:** https://www.dell.com/support/home/co/es/cobsdt1/drivers/driversdetails?driverid=41y2v
- **License:** Dell Software License
- **Version:** 2.0

Tested on (Requirements)

- **Nagios Core:** Version $\geq$ 3.5.0

Dependencies

PIP:

- omsdk
- omdrivers
- argparse

YUM:

- perl-Sys-Syslog (SNMPTT Dependency)
- perl-Net-IP
- perl-Net-SNMP
- libwsman1
- openwsman-perl
- perl-Socket6
- snmptt
- net-snmp-perl
- srvadmin-idrac7
- java-1.8.0-openjdk
- java-1.8.0-openjdk-devel
- python-netaddr

Preconfiguration

Note: This procedure has been automated in our Ansible healthcheck role.

1. Edit `/etc/snmp/snmptt.ini` in order to enable the DNS resolution and enhanced logging options.

```
[General]
dns_enable = 1
net_snmp_perl_enable = 1
translate_log_trap_oid = 1

[Logging]
stdout_enable = 1
log_enable = 1
log_file = /var/log/snmptt/snmptt.log
log_system_enable = 1
log_system_file = /var/log/snmptt/snmpttsystem.log
unknown_trap_log_enable = 1
unknown_trap_log_file = /var/log/snmptt/snmpttunknown.log

[Debugging]
DEBUGGING = 1
DEBUGGING_FILE = /var/log/snmptt.debug
DEBUGGING_FILE_HANDLER = /var/log/snmptt/snmptthandler.debug
```

Note: The following lines are added to `/etc/snmp/snmptt.ini` by the Dell plugin installation process.

```
[TrapFiles]
# A list of snmptt.conf files (this is NOT the snmptrapd.conf file).  The COMPLETE␣
↪path
# and filename.  Ex: '/etc/snmp/snmptt.conf'
snmptt_conf_files = <<END
/usr/local/nagios/dell/config/templates/Dell_PowerVaultMD_Traps.conf
/usr/local/nagios/dell/config/templates/Dell_EqualLogic_Traps.conf
/usr/local/nagios/dell/config/templates/Dell_Compellent_Traps.conf
/usr/local/nagios/dell/config/templates/Dell_Chassis_Traps.conf
/usr/local/nagios/dell/config/templates/Dell_Agent_free_Server_Traps.conf
/etc/snmp/snmptt.conf
END
```

2. Edit `/etc/snmp/snmptrapd.conf` and add the following lines

```
traphandle default /usr/sbin/snmptthandler
disableAuthorization yes
```

3. Configure both SNMPTT and SNMPTRAPD services to start on boot time.

```
chkconfig snmptrapd on
chkconfig snmptt on
```

Installation

This taskfile is executed only if the directory `/usr/local/nagios/libexec` doesn't exist. This state is registered in the taskfile `nagios-plugins-installed.yml`, with the module **stat**.

For more information about this registers read the section *Installing Nagios Plugins*.

The installation process[1] consists of downloading and uncompressing the plugin, then the script `Dell_OpenManage_Plugin/Install/install.sh` is executed.

[1] https://www.dell.com/support/article/es/es/esbsdt1/sln310619/installation-of-dell-openmanage-plugin-for-nagios-xi-on-centos?lang=en

```
---
- name: Assure existence of tempDir
  file:
    path: "{{ temp_dir }}"
    state: directory

- name: Download Nagios Dell Plugin
  get_url:
    url: "{{ dell_plugin_url }}"
    dest: "{{ temp_dir }}"

- name: Extract Nagios Dell Plugin
  unarchive:
    src: "{{ temp_dir }}/Dell_OpenManage_Plugin_Nagios_Core_41Y2V_RHEL6_2.0_A00.tar.gz
↪"
    dest: "{{ temp_dir }}"

- name: Exec installation script
  shell: "bash install.sh"
  args:
    chdir: "{{ temp_dir }}/Dell_OpenManage_Plugin/Install"
```

Configuration

This playbook synchronizes the dell configuration files located in `/usr/local/nagios/dell/config/objects/` and the dell_contacts file.

```
---
- name: Synchronize configuration
  copy:
    src: "{{ item }}"
    dest: "/{{ item }}"
  with_items:
    - usr/local/nagios/dell/config/objects/file1.cfg
    - usr/local/nagios/dell/config/templates/dell_contacts.cfg
  notify:
    - nagios_restart
```

Usage

The plugin has a script that discovers and generates the necessary configuration files for Dell servers present in a given IP, IP range or subnet.

To configure a specific host:

```
$ /usr/local/nagios/dell/scripts/dell_device_discovery.pl -H host -P protocol -f
```

Params	Value
-h, –help	Display help text.
-H, –host <host>	IP or hostname.
-S, –subnet <subnet>	Subnet with mask.
-F, –filewithiplist	Absolute path of a file with of newline separated Hosts.
-P Protocol	1(SNMP) 2(WSMAN).
-f	Force rewrite of config file.

Note: If you need more information about the command, execute it with the flag `-h`

Troubleshooting

Incorrect hostname detection

It's possible that the `dell_device_discovery.pl` script detects an incorrect hostname in the discovery process (Ej: idrac8). It generates incorrect configurations because the host_name attribute in Nagios has to be unique for each Host definition.

The solution is to edit the host definition:

```
define host{
  use                     Dell Agent-free Server
  host_name               idrac8
  alias                   idrac8
  address                 192.168.1.1
  display_name            idrac8
  icon_image              idrac.png
  ...                     ...
}

define service{
  use                     Dell Traps
  host_name               idrac8
  service_description     Dell Server Traps
}
```

Update the fields host_name, alias, and display_name.

```
define host{
  use                     Dell Agent-free Server
  host_name               mgmt-master
  alias                   mgmt-master
  address                 192.168.1.1
  display_name            mgmt-master
  icon_image              idrac.png
  ...                     ...
}

define service{
```

(continues on next page)

(continued from previous page)

```
  use                     Dell Traps
  host_name               mgmt-master
  service_description     Dell Server Traps
}
```

References

iLO AgentLess Management - Nagios Plugin

Contents

Basic information

- **Official Website:** https://exchange.nagios.org/directory/Plugins/Network-and-Systems-Management/Others/A-Nagios-Plug-2Din-for-iLO-Agentless-Management-(HPE-ProLiant-Server)/details
- **License:** GPL
- **Version:** 1.5 - Nagios HPE iLO RESTful Plugin

Tested on (Requirements)

- HPE ProLiant Server
- **Nagios Core:** Version $\geq$ 3.5.0

Dependencies

- curl
- libcurl
- libcurl-devel
- nmap
- libtdb-devel
- python
- net-snmp-utils *(Also required by Nagios Core)*
- glibc-devel *(Also required by Nagios Core)*

Installation

This taskfile is executed only if there aren't any plugins in the directory `/usr/local/nagios/libexec` that matches the regular expression `*hpeilo*`. This state is registered in the taskfile `nagios-plugins-installed.yml`, with the module **find**.

For more information about these registers read the section *Installing Nagios Plugins*.

The installation process[1] consists of downloading the plugin from the original repository, then it is necessary to regenerate the configure files if the aclocal version is not 1.14. More information in the section *aclocal-1.14 is missing*. Finally, the configure, make and make install are executed.

```
---
- name: Create Temp Dir
  file:
    path: "{{ temp_dir }}"
    state: directory

- name: Download iLO REST Plugin
  git:
    repo: "{{ ilo_plugin_url }}"
    clone: yes
    dest: "{{ temp_dir }}/nagios-ilo-plugin"

- name: Reconfigure files for the aclocal in the system
  shell: "autoreconf -vfi"
  args:
    chdir: "{{ temp_dir }}/nagios-ilo-plugin"

- name: Exec configure
  shell: "./configure"
  args:
    chdir: "{{ temp_dir }}/nagios-ilo-plugin"

- name: Make
  make:
    chdir: "{{ temp_dir }}/nagios-ilo-plugin"
```

(continues on next page)

[1] "User's Manual Nagios Plug-in for HPE iLO RESTful Extension", Hewlett Packard Enterprise (HPE). Retrieved December 3, 2018, from https://goo.gl/knRFPr.

(continued from previous page)

```
- name: Make install
  make:
    chdir: "{{ temp_dir }}/nagios-ilo-plugin"
    target: install
```

Configuration

Synchronizes the iLO credential files and the iLO plugin configuration with the version present in the repo and configures the permissions over Nagios credential file.

```
---
- name: Assures the existance of the ilo plugin config directory.
  file:
    path: "/usr/local/nagios/etc/ilo"
    state: directory

- name: Assures the existance of the nagios credentials directory.
  file:
    path: "/etc/nagios"
    state: directory

- name: Synchronize configuration
  copy:
    src: "{{ item }}"
    dest: "/{{ item }}"
  with_items:
    - usr/local/nagios/etc/ilo/ilo.cfg
    - etc/nagios/.nagios_restful_credential.tdb
  notify:
    - nagios_restart

- name: Correct permissions of nagios-credentials file
  file:
    path: /etc/nagios/.nagios_restful_credential.tdb
    mode:  0600
    owner: nagios
    group: nagios
```

Warning: It's important to configure nagios as the owner of the nagios credential file as is described in this configuration taskfile.

Usage

Defining Hosts, commands and services

Warning: It's necessary to read the section *Installation in CentOS 7* before proceeding to configure.

For starting the configuration of the iLO plugin, run the following command:

```
/usr/local/nagios/libexec/hpeilo_nagios_config
```

Example of simple configuration using the script:

```
Do you wish to enable passive HPE iLO host discovery (y/n) (Blank is y):
Do you wish to enable active discovery (y/n) (Blank is y):
Do you wish to configure host groups (y/n) (Blank is y):
Enter host group name: ilo-4_example_group
Enter host group "ilo-4_example_group" IP range: 192.168.1.10
Enter host group "ilo-4_example_group" SNMP read-only community string (Blank default␣
↪value  public):
Enter host group "ilo-4_example_group" SNMP Trap community string (Blank default␣
↪value  public):
Do you wish to add one more host group (y/n) (Blank is n):
Enter host group check interval time in minutes (Blank default value - 5 minutes ): 1
 default is exists. Do you wish to update or not? (y/n): n
Do you wish to configure iLO credential for single server if there're different with
the default iLO credential (y/n) (Blank is n):

**************Configured Data****************
PASSIVE_DISCOVERY 1
ACTIVE_DISCOVERY  1
 HOSTGROUP: ilo-4_example_group,192.168.1.10,public,public
CHECK_INTERVAL 1
*****************************************

Do you wish to write above configured data to host-group configuration file. (y/n)␣
↪(Blank is y):
HPE iLO host-group configuration file saved at /usr/local/nagios/libexec/hpeilo_
↪nagios_config.cfg.
HPE iLO Hosts/Services discovered configuration file is existing:/usr/local/nagios/
↪etc/ilo/ilo.cfg
Do you wish to replace it?(y/n) (Blank is y):
```

With regard to the current question: If you answer NO, then the script will save a backup in the folder `/usr/local/nagios/etc/ilo/.backup/`, and will override the configuration file.

```
Do you wish to run configuration file to complete the process (y/n) (Blank is y):
Running configuration  file ....
Reading host-group configuration file : /usr/local/nagios/libexec/hpeilo_nagios_
↪config.cfg
Discovery using configuration file
passive configuration enabled
[Failed]
Restart snmptrapd
Redirecting to /bin/systemctl restart snmptrapd.service
[Done]
active configuration enabled
Scan the network range: 192.168.1.10-192.168.1.10   with the total hosts '1'
env: /etc/init.d/nagios: Permiso denegado
Progress: 100% (scanned 1 of 1 hosts)
Total hosts '1' added of '1' hosts
```

Warning: In version 1.5 of this plugin, there is an unimplemented service, so it's necessary to do the procedure explained in the section *Network Service not Implemented*

Note: If you created the temporal file /etc/init.d/nagios, it's important to delete it after generating the configuration of iLO plugin.

iLO Credentials

The iLO credentials used by the plugin are cyphered and stored in the file `/etc/nagios/.nagios_restful_credential.tdb` and can be modified with the following command:

```
/usr/local/nagios/libexec/credit_save -H <hostname> [-V] [-h]
```

Params	Value
-H HOST	Hostname, Host IP or default if you want to stablish default credentials
-V	Version number
-h	Prints this help information

Troubleshooting

aclocal-1.14 is missing

If your version of aclocal is different than 1.14 it is necessary to regenerate the **configure** files in order to compile correctly the plugin.

This is possible executing the command `autoreconf -vfi` in the source directory. If this is not executed, when `./configure` is executed the following error message will appear.

```
WARNING: 'aclocal-1.14' is missing on your system.
You should only need it if you modified 'acinclude.m4' or
'configure.ac' or m4 files included by 'configure.ac'.
The 'aclocal' program is part of the GNU Automake package:
<http://www.gnu.org/software/automake>
It also requires GNU Autoconf, GNU m4 and Perl in order to run:
<http://www.gnu.org/software/autoconf>
<http://www.gnu.org/software/m4/>
<http://www.perl.org/>
```

Installation in CentOS 7

Warning: Although the official page provides a compatible version with CentOS 7, the scripts for generating the configuration files are written for a CentOS version that still uses init, searching information in the files that match the following regular expression: `/etc/init.d/nagios*`

The error will show an error output during the execution of the configuration scripts as the showed below:

```
Do you wish to write above configured data to host-group configuration file. (y/n)␣
↪(Blank is y):
HPE iLO host-group configuration file saved at /usr/local/nagios/libexec/hpeilo_
↪nagios_config.cfg
```

(continues on next page)

(continued from previous page)

```
grep: /etc/init.d/nagios*: No existe el fichero o el directorio
dirname: falta un operando
Pruebe 'dirname --help' para más información.
```

In this installation, CentOS 7 configured Nagios Daemon with systemd specifications, so the file /etc/init.d/nagios* was not created.

In order not to modify the code of the iLO plugin scripts, the alternative proposed here is to generate a temporal file `/etc/init.d/nagios` during the generation of the configuration files with the following content:

```
NagiosIloCfgFile=/usr/local/nagios/etc/nagios.cfg
```

Network Service not Implemented

The services provided by this plugin are:

1. System Health
2. Fan
3. Memory
4. Network
5. Power Supply
6. Processor
7. Storage
8. Temperature

Although Network function is not implemented, the scripts written by the developers of the plugin generate configuration lines for this unimplemented function. So, after generating the configuration files with the iLO plugin script, it is necessary to remove manually this service from the service definitions generated.

It's necessary to remove the definitions of the service and the servicegroup.

Example:

```
define servicegroup {
  servicegroup_name    Network
  members              server1, Network, server2, Network
}

define service {
  use                 generic-iLO-service
  hostgroup_name      group-name
  service_description Network
  check_command       nagios_hpeilo_restful_engine!4
}
```

iLO Memory service in UNKNOWN state

The error message displayed in the **Status information** field in the Nagios **Service Status Details** is normal when the machine is powered off.

Message:

```
Check iLO credit is correct saved.(/usr/local/nagios/libexec/credit_save -H 10.150.4.
↪188)
```

References

PNP4Nagios

PNP is an addon to Nagios which analyzes performance data provided by plugins and stores them automatically into RRD-databases.[1]

Table of Contents

Basic information

- **Official Website:** https://docs.pnp4nagios.org/
- **License:** GNU General Public License v2.0

Tested on (Requirements)

- **Nagios Core:** Version $\geq$ 2.x

Dependencies

- RRDTool
- PHP

[1] PNP4Nagios Docs. Retrieved February 8, 2019, from https://docs.pnp4nagios.org/pnp-0.6/start

- Perl
- Perl RRDTool
- PHP-GD

Installation

This taskfile is executed only if the folder `/usr/local/pnp4nagios` doesn't exist. This state is registered in the taskfile `nagios-plugins-installed.yml`, with the module **stat**.

For more information about these registers read the section *Installing Nagios Plugins*.

The installation process consists of downloading, uncompressing, configuring and compiling the plugin.

Note: **fullinstall** option installs the main program, runlevel scripts, config and HTML files.

```
---
- name: Create Temp Dir
  file:
    path: "{{ temp_dir }}"
    state: directory

- name: Download PNP4Nagios Plugin
  get_url:
    url: "{{ pnp_plugin_url }}"
    dest: "{{ temp_dir }}/{{ pnp_plugin_file }}"

- name: Extract PNP4Nagios Plugins
  unarchive:
    src: "{{ temp_dir }}/{{ pnp_plugin_file }}"
    dest: "{{ temp_dir }}"

- name: Exec configure
  shell: "./configure"
  args:
    chdir: "{{ temp_dir }}/pnp4nagios-0.6.26"

- name: Make all
  make:
    chdir: "{{ temp_dir }}/pnp4nagios-0.6.26"
    target: all

- name: Make Full install
  make:
    chdir: "{{ temp_dir }}/pnp4nagios-0.6.26"
    target: fullinstall
```

Configuration

PNP4Nagios can be configured in 5 different modes:

- Synchronous
- Bulk

- Bulk with NPCD
- Bulk with NPCD and npcdmod
- Gearman

This procedure will configure PNP4Nagios in **Bulk with NPCD mode**. Bulk mode reduces the load on the Nagios Server because the data collection and Processing is not executed by a Nagios process but by NPCD.

Nagios Performance C Daemon (NPCD) processes the Performance Data produced by Nagios plugins.

Note: For more information about the different configuration modes, read[2].

Edit the following options in **nagios.cfg** file:

```
process_performance_data=1

service_perfdata_file=/usr/local/pnp4nagios/var/service-perfdata
service_perfdata_file_template=DATATYPE::SERVICEPERFDATA\tTIMET::$TIMET$\tHOSTNAME::
↪$HOSTNAME$\tSERVICEDESC::$SERVICEDESC$\tSERVICEPERFDATA::$SERVICEPERFDATA
↪$\tSERVICECHECKCOMMAND::$SERVICECHECKCOMMAND$\tHOSTSTATE::$HOSTSTATE
↪$\tHOSTSTATETYPE::$HOSTSTATETYPE$\tSERVICESTATE::$SERVICESTATE$\tSERVICESTATETYPE::
↪$SERVICESTATETYPE$
service_perfdata_file_mode=a
service_perfdata_file_processing_interval=15
service_perfdata_file_processing_command=process-service-perfdata-file

host_perfdata_file=/usr/local/pnp4nagios/var/host-perfdata
host_perfdata_file_template=DATATYPE::HOSTPERFDATA\tTIMET::$TIMET$\tHOSTNAME::
↪$HOSTNAME$\tHOSTPERFDATA::$HOSTPERFDATA$\tHOSTCHECKCOMMAND::$HOSTCHECKCOMMAND
↪$\tHOSTSTATE::$HOSTSTATE$\tHOSTSTATETYPE::$HOSTSTATETYPE$
host_perfdata_file_mode=a
host_perfdata_file_processing_interval=15
host_perfdata_file_processing_command=process-host-perfdata-file
```

It's necessary to redefine the `process-service-perfdata-file` and `process-host-perfdata-file` commands:

```
define command{
             command_name    process-service-perfdata-file
             command_line    /bin/mv /usr/local/pnp4nagios/var/service-perfdata /usr/
↪local/pnp4nagios/var/spool/service-perfdata.$TIMET$
}

define command{
             command_name    process-hosts-perfdata-file
             command_line    /bin/mv /usr/local/pnp4nagios/var/host-perfdata /usr/
↪local/pnp4nagios/var/spool/host-perfdata.$TIMET$
}
```

"Using these commands the file service-perfdata will be moved to `/usr/local/pnp4nagios/var/spool/` after the interval specified in `service_perfdata_file_processing_interval` has passed. The Nagios macro **$TIMET$** is appended to the filename to avoid overwriting of old files unintentionally."[2]

NPCD monitors `/usr/local/pnp4nagios/var/spool/` directory and `process_perfdata.pl` processes these files decoupled from Nagios.

[2] PNP4Nagios Configuration. Retrieved February 11, 2019, from https://docs.pnp4nagios.org/pnp-0.6/config

Note: This configuration process is intended to be automated with the task *nagios-config.yml*. These options should be written previously in the config files present in the ansible role.

After configuring PNP4Nagios, it's recommended to start NPCD as a daemon, executing:

```
/usr/local/pnp4nagios/bin/npcd -d -f /usr/local/pnp4nagios/etc/npcd.cfg
```

The task `pnp4nagios-config` will enable NPCD in boot time and will synchronize a PHP configuration file needed by PHP-GD, a dependency of PNP4Nagios.

Note: For more information about PHP-GD read *php-gd not detected*

```
---
- name: Service NPCD enabled at boot time
  service:
    name: npcd
    enabled: yes
    state: started

- name: Synchronize gd.ini file
  copy:
    src: etc/php.d/gd.ini
    dest: /etc/php.d/gd.ini
  notify: apache_restart

- name: Correct Group and Owner for PNP dirs
  file:
    path: /usr/local/pnp4nagios
    group: nagcmd
    owner: nagios
    recurse: yes
```

Test PNP4Nagios

1. Perl Script

PNP4Nagios can be tested executing a verification script provided by the official page. The main script options are:

Params	Value
-m <mode>	sync | bulk | bulk+npcd | npcdmod
-c <path>	Location of nagios.cfg
-p <path>	Path to PNP config dir

```
wget http://verify.pnp4nagios.org/verify_pnp_config
perl verify_pnp_config -m bulk+npcd -c /usr/local/nagios/etc/nagios.cfg -p /usr/local/
↪pnp4nagios/etc/
```

2. PHP Home Page Check

After installing and configuring PNP4Nagios, the Home webpage `http://<host>/pnp4nagios` will display a detailed environment test:

PNP4Nagios Environment Tests

The following options are determined by "configure". If any of the tests have failed, consult the documentation for more information on how to correct the problem.

PNP4Nagios Version	pnp4nagios-0.6.26
Prefix	/usr/local/pnp4nagios
Configure Arguments	./configure
RRD Storage	/usr/local/pnp4nagios/var/perfdata is readable.
RRDtool Binary	/usr/bin/rrdtool is executable by PHP
PHP GD extension	Pass
PHP function proc_open()	Pass
PHP zlib extension	Pass
PHP session extension	Pass
PHP JSON extension	Pass
PHP magic_quotes_gpc	Off
PHP socket extension	Pass
Apache Rewrite Module	Pass

Kohana Environment Tests

The following tests have been run to determine if Kohana will work in your environment. If any of the tests have failed, consult the documentation for more information on how to correct the problem.

PHP Version	5.4.16
System Directory	/usr/local/pnp4nagios/lib/kohana/system/
Application Directory	/usr/local/pnp4nagios/share/application/
Reflection Enabled	Pass
Iconv Extension Loaded	Pass
URI Determination	Pass

Your environment passed all requirements. Remove or rename the `/usr/local/pnp4nagios/share/install.php` file now.

If your environment passed all requirements, remove the `/usr/local/pnp4nagios/share/install.php` file.

Troubleshooting

php-gd not detected

PROBLEM: PHP-GD is installed but is unrecognizable by PHP.

SOLUTION: To create a file in `/etc/php.d/` that contains the extension information indicating the path to the gd.so library. Example:

```
[gd]
extension=/usr/lib64/php/modules/gd.so
```

Can't find Nagios Environment

PROBLEM: If PNP4Nagios is configured in Sync mode using Nagios Core 4.x, the following message appears on the web page:

```
Cant find Nagios Environment. Exiting ....
perfdata directory "/usr/local/pnp4nagios/var/perfdata/" is empty. Please check your␣
↪Nagios config.
```

SOLUTION: Apparently, it's a Nagios 4 bug, where environment data used by pnp4nagios is not provided in sync mode[3]. Bulk mode with NPCD is a better mode for configuring PNP4Nagios.

References

Usage

Before executing the role it's important to verify the value of the variables in the file `roles/healthckeck/vars/main.yml`. These variables were created in order to uncouple from the code things like IPs, URLs and passwords. In the case of passwords, we used **Ansible Vault** for ciphering them.

```
ansible-vault playbooks/healthcheck.yml --ask-vault-pass
```

Caution: This Ansible role was created thinking in the Ansible Philosophy: **The tool should be used to represent the state of the server, not as a procedural language but as a declarative one.**

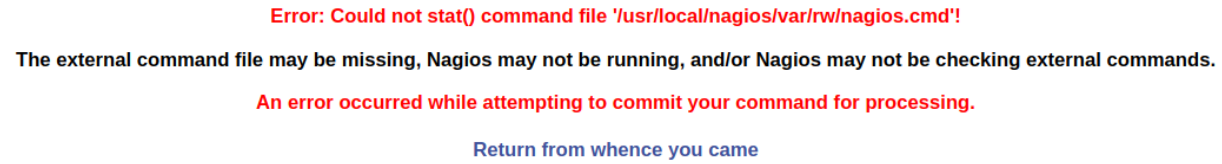
This role was developed to be run multiple times in the same server: If the real state doesn't matches with the role state, the server is modified in order to match both states. If the server has well configured and well installed Nagios and it's plugins, running the playbook will say **Ok** in most of the tasks, so it won't break any configuration.

Note: The flag `--ask-vault-pass` is used because this role uses ansible-vault for encrypting private data like passwords.

[3] Nagios Support Forum - PNP4nagios: Can't find Nagios Environment [SOLVED]. Retrieved February 11, 2019, from https://support.nagios.com/forum/viewtopic.php?f=7&t=29953

Troubleshooting

Nagios Command Error



This error occurs when the user generates a modification in the Nagios Command file executing different actions with the Nagios Web Interface *(eg. re-schedule checks)*. This error is corrected with the execution of the task *selinux-config.yml*.

By default, the Apache daemon cannot read/write the files under `/usr/local/nagios/var/rw` or execute from `/usr/local/nagios/sbin` because these directories doesn't belongs to the `httpd_sys_rw_content_t` SELinux context and `httpd_sys_script_exec_t` context respectively. This is what originates the problem.

Authors

- Andrés Felipe Zapata Palacio <azapat47@eafit.edu.co>

3.6.3 Sensu

The Sensu monitoring event pipeline[1] empowers businesses to automate their monitoring workflows and gain deep visibility into their multi-cloud infrastructure, from Kubernetes to bare metal.

This Tool is used in the computing center to monitor Memory usage, Service availability, CPU load, and Temperature status.

Sensu - 0.26.5

This documentation page describes the process of installation and configuration of an entire monitoring environment using Sensu as the Core and CentOS 7 as the base Operating System.

Contents

[1] Workflow automation for monitoring | Sensu. (n.d.). Retrieved June 11, 2019, from https://sensu.io/

Basic information

- **Deploy date:** 11th June, 2019
- **Official Website:** https://sensu.io/
- **License:** MIT License.

Directory Hierarchy

The two main Sensu directories are:

Installation directory

`/opt/sensu` is the installation directory. It contains services, binaries, libraries and its own Ruby installation with the plugins installed using the command `sensu-install`.

Configuration directory

`/etc/sensu` contains the configuration files, plugins, and handlers. The configuration definition can be present in `/etc/sensu/config.json` or as an independent JSON file in the directory `/etc/sensu/conf.d/`.

Introduction

The general sensu architecture is summarized as one or more servers that assign monitoring tasks to its clients through a message broker, and stores the results in a Database in Memory. Information like configuration and results is provided through an API.

Sensu Services

- **sensu-client:** It executes the tasks indicated in the Message Broker (RabbitMQ) by the Sensu server. It's necessary to restart it if the client configuration changes (local checks, address, etc) to refresh the configuration. If you change a configuration in the client, it's not necessary to restart the server.
- **sensu-server:** Distributes the monitoring tasks through the message broker, and reacts executing the specified handlers when a result has a critical or warning state. It's necessary to restart it if the server configuration changes (checks, handlers, etc) in order to refresh its execution parameters (send monitoring tasks and receive results).
- **sensu-api:** Manages the service information provided from the Sensu-server to external systems like Uchiwa. It's necessary to restart it if the server configuration changes (checks, handlers, etc) in order to update the informative layer.

Concepts

- **Clients:** Monitoring agents that execute checks and replies the results to its associated Sensu Server.
- **Subscriptions:** A name that groups 0 or more clients around one particular role or responsibility.
- **Checks:** Commands executed by the Sensu client which monitor a condition or collect measurements (server resources, services, etc).
- **Handlers:** Actions executed by the Sensu server on events, such as sending an email alert, creating or resolving an incident (e.g. in PagerDuty, ServiceNow, etc), or storing metrics in a time-series database (e.g. Graphite).
- **Mutators:** Transform event data prior to handling (i.e. add new attributes to the response)
- **Plugins:** Provide executable scripts or other programs that can be used as Sensu checks, handlers or mutators.

- **Filters:** Inspect event data and match its keys/values with filter definition attributes, to determine if the event should be passed to an event handler. Filters are commonly used to filter recurring events (i.e. to eliminate notification noise).

For more information, read[2].

Installation

This first installation procedure is the same in both: Sensu Server and Sensu Client. The procedure is explained for a machine whose Operating System is CentOS 7.

1. Install the Epel-Release repository

```
$ yum install epel-release
```

2. Add the Official Sensu repository, adding the following file to `/etc/yum.repos.d/sensu.repo`

```
[sensu]
name=sensu-main
baseurl=http://repositories.sensuapp.org/yum/el/7/x86_64/
gpgkey=https://repositories.sensuapp.org/yum/pubkey.gpg
gpgcheck=0
enabled=1
```

3. Install Sensu

```
$ yum install sensu
```

Sensu Server

Note: It's important to know that a Sensu Server can be also a Sensu Client.

After executing the previous steps, if you are not installing the Sensu Client, but the Sensu Server, proceed as follows:

1. Install the additional dependencies for managing the communication between clients-server.
   - **RabbitMQ** is the message broker that manages the communication.
   - **Erlang** is a programming language and a runtime dependency for RabbitMQ.
   - **Redis** works as Database in Memory and stores temporarily the monitoring information.
   - **Uchiwa** is a web Dashboard for visualizing Sensu status and Configuration.

```
$ yum install erlang redis uchiwa
```

   RabbitMQ can be installed from its official RPM:

```
$ yum install https://www.rabbitmq.com/releases/rabbitmq-server/v3.6.6/rabbitmq-
↪server-3.6.6-1.el6.noarch.rpm
```

[2] Sensu Reference | Sensu Core 0.29. (n.d.). Retrieved June 17, 2019, from https://docs.sensu.io/sensu-core/0.29/reference/

Configuration

Service configuration

You should start and enable at boot time the following services:

1. Sensu Client

- sensu-client

2. Sensu Server

- uchiwa
- sensu-server
- sensu-api
- redis
- rabbitmq-server

RabbitMQ Configuration

It's necessary to define authentication credentials to let Clients communicate in a secure way with the Sensu Server through the Message Broker RabbitMQ. This procedure is executed only once in the Sensu Server.

```
$ rabbitmqctl add_vhost /sensu
$ rabbitmqctl add_user sensu PASSWORD
$ rabbitmqctl set_permissions -p /sensu sensu ".*" ".*" ".*"
```

Sensu Configuration

1. Sensu Client

1. Add the Client definition in `/etc/sensu/config.json` or in any file with *json* extension into the directory `/etc/sensu/conf.d/`, specifying hostname, subscriptions, etc.

   Example: /etc/sensu/conf.d/client.json

```
{
  "client": {
    "name": "HOSTNAME",
      "bmc_address": "10.0.0.1",
      "subscriptions": [
        "subscription-1",
        "subscription-2"
      ]
  }
}
```

2. Add the Transport definition in the configuration directory:

   Example: /etc/sensu/conf.d/transport.json

```
{
  "transport": {
    "name": "rabbitmq",
    "reconnect_on_error": true
  }
}
```

3. Add the RabbitMQ definition specifying the credentials previously defined:

Example: /etc/sensu/conf.d/rabbitmq.json

```
{
  "rabbitmq": {
        "host": "SENSU_SERVER_IP",
        "port": 5672,
        "vhost": "/sensu",
        "user": "sensu",
        "password": "PASSWORD"
  }
}
```

2. Sensu Server

Add the Uchiwa configuration file:

Example: /etc/sensu/conf.d/uchiwa.json

```
{
  "sensu": [
    {
      "name": "Site Name",
      "host": "127.0.0.1",
      "port": 4567,
      "ssl": false,
      "path": "",
      "user": "",
      "pass": "",
      "timeout": 10
    }
  ],
  "uchiwa": {
    "host": "127.0.0.1",
    "port": 3000,
    "refresh": 10
  }
}
```

Plugins

Excluding the specific configurations that each plugin can have, the general process to add a plugin is described as follows:

1. Download the script into a directory readable by **sensu** user.
2. Define the check configuration in `/etc/sensu/config.json` or as an independent JSON file in `/etc/sensu/conf.d/`, specifying the execution line, frequency, and which subscribers will be associated with the check.
3. Restart **sensu-server** and **sensu-api**. If the check is defined as standalone (locally in the client) restart **sensu-client**.

Sensu Plugin - Remediator

This plugin reads configuration from a check definition and triggers specified remediation actions (defined as other checks) via the Sensu API when the occurrences and severities reach certain values.[1]

Contents

Basic information

- **License:** MIT License

Installation

Note: The entire process of installation and configuration has to be executed only in the Sensu Server.

The handler is a Ruby script that can be downloaded from the official repository: https://github.com/sensu-plugins/sensu-plugins-sensu/blob/master/bin/handler-sensu.rb. It can be located in `/etc/sensu/handlers/remediator.rb`

Configuration

Add the Remediator configuration file:

Example: `/etc/sensu/conf.d/handlers/remediator.json`

[1] Sensu-Plugins. (n.d.). Sensu-plugins/sensu-plugins-sensu. Retrieved June 12, 2019, from https://github.com/sensu-plugins/sensu-plugins-sensu/blob/master/bin/handler-sensu.rb

```
{
  "handlers": {
    "remediator": {
      "command": "/etc/sensu/handlers/remediator.rb",
      "type": "pipe",
      "severities": ["critical"]
    }
  }
}
```

Usage

Remediator is a Sensu handler that triggers actions when a defined check service changes to critical status. The way Remediator triggers this action is executing it in the hosts that follow a subscription with the hostname of the client that failed.

Example: A compute node called **test-node** fails in a defined monitoring check. Remediator triggers the defined action in the hosts that follow the subscription **test-node**.

That's why every host that wants to receive the trigger from **Remediator** has to be subscribed to its hostname.

The procedure for setting a handler for a check with remediator is:

1. Add the hostname as a subscription in each node that will execute the handler.

```
{
  "client": {
    "name": "HOSTNAME",
      "bmc_address": "10.0.0.1",
      "subscriptions": [
        "subscription-1",
        "subscription-2",
        "HOSTNAME"
      ]
  }
}
```

2. Create the definition of the check that will be executed to *remediate* a CRITICAL result.

   Example: If the service ssh is down, the check that will remediate that is the following:

```
{
  "checks": {
    "remediate-service-ssh": {
      "command": "service sshd start",
      "handlers": ["remediator"],
      "subscribers": ["ssh"],
      "standalone": false,
      "publish": false
    }
  }
}
```

3. Update the definition of the check that will trigger the remediator handler if changes its state to CRITICAL. You have to add an attribute called remediation, specifying which check will remediate the failure.

   Example: Checks if SSH is running. In this case, the check called *check-sshd* will be remediated by the check *remediate-service-ssh* if it changes to severity 2 (CRITICAL) one or more times.

```
{
  "checks": {
    "check-sshd": {
      "command": "check-process.rb -p /usr/sbin/sshd",
      "interval": 60,
      "handlers": ["remediator","mailer"],
      "subscribers": ["ssh"],
      "standalone": false,
      "remediation": {
        "remediate-service-ssh": {
          "occurrences": ["1+"],
          "severities": [2]
        }
      }
    }
  }
}
```

4. If the handler needs superuser permissions to run its action, it's necessary to add that rule in the sudoers configuration in the Sensu-Clients.

Troubleshooting

sudo: sorry, you must have a tty to run sudo

This message appears when you need to run the remediator's action as superuser, you defined the rule in sudoer's configuration but you haven't specified that the action requires a TTY.

SOLUTION: Add the following line in the sudoers file.[2]

```
Defaults:sensu !requiretty
```

References

Sensu Plugin - Mailer

This handler formats alerts as e-mails and sends them off to a pre-defined recipient.[1]

Contents

[2] Brousse, N. (2014, September 8). Sudo: Sorry, you must have a tty to run sudo. Retrieved June 13, 2019, from https://www.shell-tips.com/2014/09/08/sudo-sorry-you-must-have-a-tty-to-run-sudo/

[1] Sensu-Plugins. (n.d.). Sensu-plugins/sensu-plugins-mailer. Retrieved June 13, 2019, from https://github.com/sensu-plugins/sensu-plugins-mailer/blob/master/bin/handler-mailer.rb

- *References*

Basic information

- **License:** MIT License

Installation

Note: The entire process of installation and configuration has to be executed only in the Sensu Server.

The handler is a Ruby script that can be downloaded from the official repository: https://github.com/sensu-plugins/sensu-plugins-mailer/blob/master/bin/handler-mailer.rb.

It can be located in `/etc/sensu/handlers/mailer.rb`

Configuration

Add the Mailer configuration file and set correctly which will be the SMTP server, the sender, the mail recipients, etc:

Example: `/etc/sensu/conf.d/handlers/mailer.json`

```
{
  "mailer": {
    "mail_from": "sensu@example.com",
    "mail_to": [
      "mail1@example.com",
      "mail2@example.com"
    ],
    "smtp_address": "smtp.server.host.example.com",
    "smtp_port": "25",
    "smtp_domain": "example.com"
  },
  "handlers": {
    "mailer": {
      "type": "pipe",
      "filter": "state-change-only",
      "command": "/opt/sensu/embedded/bin/handler-mailer.rb"
    }
  }
}
```

In this example, the handler definition has the filter state-change-only associated. This filter executes the mailer handler to send mail only when there is a change in the state, that means, in the first occurrence of the state. When a check is in state OK, it doesn't count occurrences, that's why it's necessary to have both conditions in the conditional.

That filter is defined as follows:

Example: `/etc/sensu/conf.d/filters.json`

```
{
  "filters": {
    "state-change-only": {
```

(continues on next page)

(continued from previous page)

```
      "negate": false,
      "attributes": {
        "occurrences": "eval: value == 1 || ':::action:::' == 'resolve'"
      }
    }
  }
}
```

Usage

Follow these steps to add mailer as a handler that will send a mail when there is a state change in a specific monitoring check:

1. Add Mailer as a handler in the check definition:

```
{
  "checks": {
    "check_example": {
      "command": "check-example.rb",
      "subscribers": ["examples"],
      "interval": 60,
      "handlers": ["mailer"]
    }
  }
}
```

2. Restart sensu-server and sensu-api services.

References

Sensu Plugin - Check Process

Finds processes matching various filters (name, state, etc). The number of processes found will be tested against the Warning/critical thresholds. This handler formats alerts as emails and sends them off to a pre-defined recipient.[1]

Contents

[1] Sensu-Plugins. (n.d.). Sensu-plugins/sensu-plugins-process-checks. Retrieved June 14, 2019, from https://github.com/sensu-plugins/sensu-plugins-process-checks/blob/master/bin/check-process.rb

Basic information

- **License:** MIT License

Installation

Note: This installation process has to be performed in each Sensu Client that will execute this monitoring task.

Install the plugin executing:

```
$ sensu-install -p process-checks
```

It will be located in `/opt/sensu/embedded/bin/check-process.rb`, that directory is included in the Sensu user PATH, so in the check definition it's not necessary to write the full path in the configuration file.

Usage

Add the Check-Process configuration, specifying which will be its subscribers.

The -p argument is for a pattern to match against the list of running processes reported by ps.[1]

Example: Checks if there is any process running that contains **/usr/sbin/sshd** in its name, status, etc from the output of the command `ps`.

```
{ "checks":
  {
    "service_sshd": {
      "command": "check-process.rb -p /usr/sbin/sshd",
      "interval": 60,
      "handlers": ["mailer"],
      "subscribers": ["ssh"],
      "standalone": false,
    }
  }
}
```

Note: It's important to set correctly the argument to filter correctly from the running processes. Read more in *The check shows an incorrect number of running processes*.

Troubleshooting

The check shows an incorrect number of running processes

In the previous example of SSH, the specified filter was **/usr/sbin/sshd**, because the check wanted to know if the service ssh was up.

If the filter is changed to **sshd**, it will find other ssh processes that are not related to sshd, for example, a remote connection as a client, not as a server. That's why it's important to define the filter correctly, and verify that it finds what you really want.

References

Sensu Plugin - Graphite

This plugin provides native Graphite instrumentation for monitoring, including: replication status, various Graphite data queries, mutators, and handlers.[1]

Contents

Basic information

- **Official Page:** https://github.com/sensu-plugins/sensu-plugins-graphite
- **License:** MIT License

Installation

Note: This installation process has to be performed in the Sensu Server.

Install the plugin executing:[2]

```
$ sensu-install -p graphite
```

This command will install the corresponding Gem. Its scripts are located in `/opt/sensu/embedded/bin/`, that directory is included in the Sensu user's PATH, so in the check definition, it's not necessary to write the full path.

The scripts are:

- bin/check-graphite-data
- bin/check-graphite-replication
- bin/check-graphite-stats
- bin/check-graphite
- bin/extension-graphite
- bin/handler-graphite-event

[1] Sensu Plugins - Graphite. (2018, December 17). Retrieved July 17, 2019, from https://github.com/sensu-plugins/sensu-plugins-graphite
[2] Eves, M. (n.d.). Sensu Metrics Collection. Retrieved July 31, 2019, from https://blog.sensu.io/sensu-metrics-collection-beafdebf28bc

- bin/handler-graphite-notify
- bin/handler-graphite-status
- bin/handler-graphite-occurrences
- bin/mutator-graphite

Configuration

Handler definition:

```
{
  "handlers": {
    "graphite": {
      "type": "tcp",
      "mutator": "only_check_output",
      "timeout": 30,
      "socket": {
        "host": "127.0.0.1",
        "port": 2003
      }
    }
  }
}
```

Mutator definition:

```
{
  "mutators": {
    "graphite_mutator": {
      "command": "/opt/sensu/embedded/bin/mutator-graphite.rb",
      "timeout": 10
    }
  }
}
```

Usage

To store the metrics of a plugin as time series in Graphite, execute the following steps:

1. Add **graphite** to the list of handlers in the check definition.
2. Set the **type** attribute of the check as **metric**.

Note: Graphite handler will communicate Sensu with carbon-cache. In order to activate the handler with an event, it's necessary to set the attribute type to metric. Without that option, the handler will only be executed with a CRITICAL or WARNING state.

```
{
 "checks": {
  "check_ipmi_temperature": {
    "command": "check-sensors.rb -h BMC_ADDRESS -u IPMI_USER -p IPMI_
↪PASSWORD -s inlet_ambient",
    "type": "metric",
```

(continues on next page)

(continued from previous page)

```
    "subscribers": ["example"],
    "handlers":["graphite"],
    "interval": 60
  }
 }
}
```

1. Restart the Sensu Server and Sensu API services.

References

Authors

- Andrés Felipe Zapata Palacio <azapat47@eafit.edu.co>

Sensu Plugin - CPU Load

Checks the average CPU load in the last 1, 5 and 15 minutes, establishing a warning and a caution threshold for each one of these time intervals.

Contents

Basic information

- **Official Page:** https://github.com/sensu-plugins/sensu-plugins-cpu-checks
- **License:** MIT License

Installation

Note: This installation process has to be performed in each Sensu Client that will execute this monitoring task.

Install the plugin executing:

```
$ sensu-install -p loads-checks
```

This command will install the corresponding Gem. Its scripts are located in `/opt/sensu/embedded/bin/`, that directory is included in the Sensu user's PATH, so in the check definition, it's not necessary to write the full path.

The scripts are:

- bin/check-cpu.rb
- bin/check-cpu.sh
- bin/metrics-cpu-mpstat.rb
- bin/metrics-cpu-pcnt-usage.rb
- bin/metrics-numastat.rb
- bin/metrics-user-pct-usage.rb

Usage

Add the Check-CPU-load configuration, specifying which will be its subscribers and its warning and critical thresholds. The units of the thresholds is load/core.

Example: If the machine has 16 cores and we want 18 for the critical threshold, the value is 1.125 (18/16).

```
{
  "checks": {
    "check_cpu_load": {
      "command": "check-load.rb -c 1.25, 1.125, 1.125 -w 0.9365, 0.9365, 0.
↪875",
      "subscribers": ["example"],
      "interval": 60,
      "handlers": ["mailer"]
    }
  }
}
```

In this example, the check will be in WARNING state if the CPU load is greater than 15,15,14 in the last 1,5 and 15 minutes respectively, and CRITICAL state if the CPU load is greater than 20,18,18 in the last 1,5 and 15 minutes respectively.

So, with a machine of 16 cores, the values are 0.9365, 0.9365, 0.875 for the Warning threshold and 1.25, 1.125, 1.125 for the Critical threshold.

Authors

- Andrés Felipe Zapata Palacio <azapat47@eafit.edu.co>

Sensu Plugin - Disk Usage

Uses the sys-filesystem gem to get filesystem mount points and metrics[1]. Check disk capacity and inodes in the specified partitions.

Contents

[1] Sensu-Plugins. (n.d.). Sensu-plugins/sensu-plugins-disk-checks. Retrieved June 14, 2019, from https://github.com/sensu-plugins/sensu-plugins-disk-checks

Basic information

- **Official Page:** https://github.com/sensu-plugins/sensu-plugins-disk-checks
- **License:** MIT License

Installation

Note: This installation process has to be performed in each Sensu Client that will execute this monitoring task.

Install the plugin executing:

```
$ sensu-install -p disk-usage
```

It will be located in `/opt/sensu/embedded/bin/check-disk-usage.rb`, that directory is included in the Sensu user PATH, so in the check definition it's not necessary to write the full path in the configuration file.

Usage

Add the Check-Disk-Usage configuration:

The most important flags are:

Params	Value
-w	Doesn't throw a critical failure even if exceeds the thresholds.
–warn-space PERCENT	Warn if PERCENT or more of disk space used
–warn-inodes PERCENT	Warn if PERCENT or more of inodes used
–crit-space PERCENT	Critical if PERCENT or more of disk space used
–crit-inodes PERCENT	Critical if PERCENT or more of inodes used
–mount MOUNTPOINT	Comma separated list of mount point(s) (default: all)

Warning: If MOUNTPOINT contains an invalid partition (i.e /this_doesnt_exist) the plugin will not return Error, it will ignore it and check the others.

Execute `/opt/sensu/embedded/bin/check-disk-usage.rb --help` to obtain the full list of flags supported by the plugin.

Example: Checks if the partitions /,/tmp,/var are over 90% or 95% of it's capacity. The inode threshold is the default because was not specified.

```
{
  "checks":{
    "check-disk-usage": {
      "command": "/opt/sensu/embedded/bin/check-disk-usage.rb --mount /,/tmp,
↪/var --warn-space 90 --crit-space 95",
      "interval": 60,
      "subscribers": ["example"],
      "handlers": ["mailer"]
    }
  }
}
```

References

Sensu Plugin - Memory Checks

This plugin provides native memory instrumentation for monitoring and metrics collection.

Contents

Basic information

- **Official Page:** https://github.com/sensu-plugins/sensu-plugins-memory-checks
- **License:** MIT License

Installation

Note: This installation process has to be performed in each Sensu Client that will execute this monitoring task.

Install the plugin executing:

```
$ sensu-install -p memory-checks
```

This command will install the coresponding Gem. It's scripts are located in `/opt/sensu/embedded/bin/`, that directory is included in the Sensu user's PATH, so in the check definition it's not necessary to write the full path.

The scripts are:

- bin/check-memory.rb
- bin/check-memory.sh
- bin/check-memory-percent.rb
- bin/check-memory-percent.sh
- bin/check-ram.rb
- bin/check-swap-percent.rb
- bin/check-swap.sh
- bin/check-swap.rb
- bin/metrics-memory-percent.rb
- bin/metrics-memory.rb

Usage

Memory Usage Percent

Add the Check-Memory-Percent configuration, specifying which will be its subscribers and its warning and critical thresholds.

```
{
  "checks": {
    "check_memory_usage": {
        "command": "check-memory-percent.rb -w 90 -c 95",
        "subscribers": ["example"],
        "interval": 60
    }
  }
}
```

In this example, the check will be in WARNING state if memory usage is over 90% and CRITICAL over 95%.

Swap Usage Percent

Add the Check-Swap-Percent configuration, specifying which will be its subscribers and its warning and critical thresholds.

```
{
  "checks": {
    "check_swa_usage": {
        "command": "check-swap-percent.rb -w 10 -c 15",
        "subscribers": ["example"],
        "interval": 60
    }
  }
}
```

In this example, the check will be in WARNING state if swap usage is over 10% and CRITICAL over 15%.

Authors

- Andrés Felipe Zapata Palacio <azapat47@eafit.edu.co>

Sensu Plugin - Network Interface

A sensu plugin to efficiently monitor network interfaces on Linux allowing to track metrics like speed, duplex, operational status, link carrier, etc.[1]

Contents

Basic information

- **Official Page:** https://github.com/m4ce/sensu-plugins-network-interface
- **Author:** Matteo Cerutti
- **License:** MIT License

Installation

Note: This installation process has to be performed in each Sensu Client that will execute this monitoring task.

The plugin is a Ruby script that can be downloaded from the official repository: https://github.com/m4ce/sensu-plugins-network-interface/blob/master/bin/check-network-interface.rb.

It can be located in `/etc/sensu/plugins/check-network-interface.rb`.

Usage

The plugin has the following options:

[1] Cerutti, M. (2017, April 10). Sensu plugin for monitoring network interfaces. Retrieved June 17, 2019, from https://github.com/m4ce/sensu-plugins-network-interface

Params	Value
–carrier <STATE>	Indicates the current physical link state of the interface (default: up)
-c, –config <PATH>	Optional configuration file (default: ./network-interface.json)
–dryrun	Do not send events to sensu client socket
-d, –duplex <STATE>	Check interface duplex settings (default: full)
-x <INTERFACES>,	Comma separated list of interfaces to ignore
–ignore-interface	
-i, –interface <INTERFACES>	Comma separated list of interfaces to check (default: ALL)
–handlers <HANDLERS>	Comma separated list of handlers
-m, –mtu <MTU>	Message Transfer Unit
–operstate <STATE>	Indicates the interface RFC2863 operational state (default: up)
-s, –speed <SPEED>	Expected speed in Mb/s
-t, –txqueuelen <TXQUEUELEN>	Transmit Queue Length
-w, –warn	Warn instead of throwing a critical failure

Note: If not specified, the plugin will check by default everything (duplex, carrier, operstate, MTU, speed).

Note: The plugin will execute many independent checks that send it's individual results to Sensu Server. The flag `--dryrun` avoids that behavior. It's recommended to use this flag because it's not necessary to be that descriptive.

Configuration

Add the configuration file in a valid directory. Ej: `/etc/sensu/conf.d/checks/check-interfaces.json`

Example: Checks if the Infiniband interface has a speed of 56 Gbps, and an MTU of 65520.

```
{
  "checks": {
    "check_ib0_iface": {
      "command": "check-network-interface.rb --interface ib0 --mtu 65520 --
↪speed 56000 --dryrun",
      "subscribers": ["example"],
      "interval": 60
        }
  }
}
```

Troubleshooting

MTU never fails

For an unknown reason, the flag for setting the desired MTU doesn't work. The script takes dynamically the desired MTU. For InfiniBand interfaces, it expects 65520, so, specifying an MTU value doesn't make the difference.

Authors

- Andrés Felipe Zapata Palacio <azapat47@eafit.edu.co>

References

Sensu Plugin - IPMI Sensors Plugin

This plugin collects sensor data from an IPMI endpoint, using the rubyipmi gem[1].

Contents

Basic information

- **Official Page:** https://github.com/sensu-plugins/sensu-plugins-ipmi
- **Author:** Matt Mencel
- **License:** MIT License

Dependencies

- Gem: rubyipmi
- ipmitool or freeipmi package

You have to install rubyipmi gem using the binary located in the Sensu directories, executing:

```
$ /opt/sensu/embedded/bin/gem install rubyipmi
```

It will install the gem in the directory `/opt/sensu/embedded/lib/ruby/gems/2.3.0`, which is used by Sensu client.

[1] Mencel, M. (2018, November 01). Sensu-Plugins-ipmi. Retrieved June 17, 2019, from https://github.com/sensu-plugins/sensu-plugins-ipmi

Installation

Note: This installation process has to be performed in each Sensu Client that will execute this monitoring task.

The plugin is a Ruby script that can be downloaded from the official repository: https://github.com/sensu-plugins/sensu-plugins-ipmi/blob/master/bin/check-sensor.rb

It can be located in `/etc/sensu/plugins/check-sensor.rb`.

Usage

The plugin has the following options:

Params	Value
-h, –host IPMI_HOST	IPMI Hostname or IP (required)
-p, –password IPMI_PASSWORD	IPMI Password (required)
-v, –privilege PRIVILEGE	IPMI privilege level: CALLBACK, USER, OPERATOR, ADMINISTRATOR (defaults to USER)
-i, –ipmitool IPMI_PROVIDER	IPMI Tool Provider (ipmitool OR freeipmi). Default is ipmitool.
–scheme SCHEME	Metric naming scheme, text to prepend to .$parent.$child
-s, –sensor SENSOR_NAME	IPMI sensor to gather stats for. Default is ALL
-t, –timeout TIMEOUT	IPMI connection timeout in seconds (defaults to 30)
-u, –username IPMI_USERNAME	IPMI Username (required)

Note: The sensor name depends on the BMC version. It's different the nomenclature in iLO 4 and in iDRAC.

Configuration

Add the configuration file in a valid directory. Ej: `/etc/sensu/conf.d/checks/ipmi-temp.json`

Example: Check the Ambient temperature in iLO4

```
{
 "checks": {
  "check_ipmi_temperature": {
    "command": "check-sensors.rb -h BMC_ADDRESS -u IPMI_USER -p IPMI_
↪PASSWORD -s inlet_ambient",
    "type": "metric",
    "subscribers": ["example"],
    "handlers":["graphite"],
    "interval": 60
  }
 }
}
```

You can obtain the full list of sensors, executing the command without specifying sensors:

```
$ /opt/sensu/embedded/bin/ruby /etc/sensu/plugins/check-sensors.rb -h BMC_ADDRESS -u␣
↪USER -p PASS
```

Authors

- Andrés Felipe Zapata Palacio <azapat47@eafit.edu.co>

References

Integration with Ansible

We implemented a Role in Ansible that contains the whole process of installation and configuration of a Sensu Client, and it's integration with some plugins. Configuration files are generated dynamically using Ansible templates and pre-defined variables in the role.

The structure is:

```
---
sensu_clients_data:
  compute-1:
    subscriptions: ["compute-node","nvidia-gpu"]
  compute-2:
    subscriptions: ["compute-node"]
```

These subscriptions define the values present in the configuration file `client.json`.

Later in this documentation, these subscriptions are used also as conditionals in Ansible for the plugins installation and configuration.

Add a new Plugin

This procedure explains the general steps required to support a new plugin within the Sensu-Clients ansible role.

Note: The steps show the configuration of GPU Nvidia Plugin as an example.

1. Add a new subscription or use an existing one to associate the desired nodes with this plugin:

   ```
   sensu_clients_data:
     compute-1:
       subscriptions: ["nvidia-gpu"]
     compute-2:
       subscriptions: ["nvidia-gpu"]
   ```

2. Install the dependencies if needed. You can add a conditional that checks if the current machine has the corresponding subscription defined.

   Example:

   ```
   when: '"nvidia-gpu" in sensu_clients_data[inventory_hostname].
   ↪subscriptions'
   ```

3. Check if the plugin is installed (i.e verifying the presence of the script) and register this state into an Ansible variable:

```
- name: Nvidia plugin present
  stat:
    path: /opt/sensu/embedded/bin/metrics-nvidia.rb
  register: nvidia_plugin
```

4. Install the plugin if its presence was not registered in the last step:

```
- name: Install Nvidia Plugin
  shell: /usr/bin/sensu-install -p sensu-plugins-nvidia
  when: 'nvidia_plugin.stat.exists == false'
```

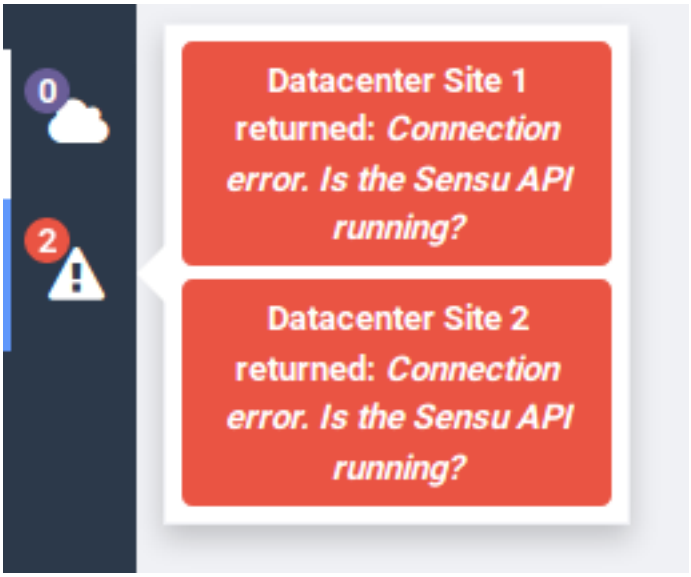
You can add a second conditional that executes this installation only if the current machine has the corresponding subscriptions defined.

Example:

```
when '"nvidia-gpu" in sensu_clients_data[inventory_hostname].subscriptions'
```

Troubleshooting

Connection error. Is the Sensu API running?



PROBLEM: Datacenter Site 1 returned: Connection error. Is the Sensu API running?

REASON 1: uchiwa.json has the default configuration (Two generic Datacenter configurations)

SOLUTION: Edit uchiwa.json with real information and restart uchiwa service.

REASON 2: Redis doesn't support ipv6 (localhost resolves to ::1). Using "localhost" instead of 127.0.0.1 for the host configuration on systems that support IPv6 may result in an IPv6 "localhost" resolution (i.e. ::1) rather than an IPv4 "localhost" resolution[1]

[1] Redis Configuration | Sensu Docs. Retrieved June 12, 2019, from https://docs.sensu.io/sensu-core/1.0/reference/redis/#redis-definition-specification

SOLUTION: Update the Redis configuration (by default located in `/etc/sensu/config.json`), changing the attribute "host" as follows:

```
{
    ...
    "redis": {
      "host": "127.0.0.1",
      "port": 6379
    },
    ...
}
```

After that, please restart the sensu-server service.

Authors

- Andrés Felipe Zapata Palacio <azapat47@eafit.edu.co>

References

3.6.4 ELK

"ELK" is the acronym for three open source projects: Elasticsearch, Logstash, and Kibana. Elasticsearch is a search and analytics engine. Logstash is a server-side data processing pipeline that ingests data from multiple sources simultaneously, transforms it, and then sends it to a "stash" like Elasticsearch. Kibana lets users visualize data with charts and graphs in Elasticsearch.

The Elastic Stack is the next evolution of the ELK Stack. For more information go to ELK's main page.

ELK Stack 6.x Installation and Configuration

Contents

Note:

- Install Java JDK 8, more recent versions may have compatibility problems.

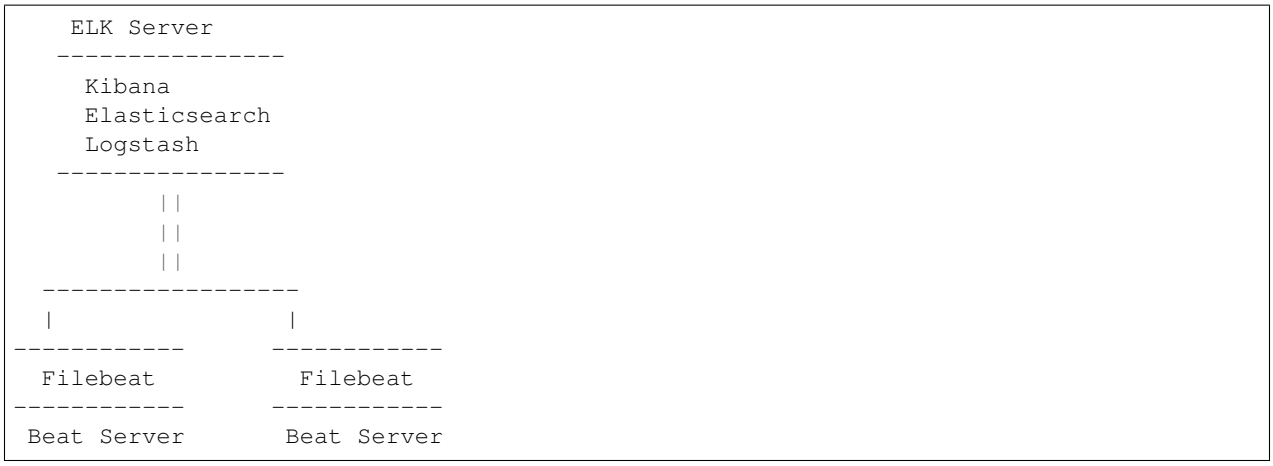
- You must use the same version for Logstash, Elasticsearch, Kibana to avoid compatibility problems.
- Also, when reading the guides check that the guide version is compatible with the version of your ELK stack.
- Consider the amount of RAM used by Logstash and Elasticsearch. By default 1GB for heap, plus the JVM which is about 2.5 GB of RAM.
- Install, configure, and start the services in the following order, then you will avoid repeating some steps, and also some problems. Note that this order is the same as the one in the role *elk*.
  - Elasticsearch: https://www.elastic.co/guide/en/elasticsearch/reference/6.6/index.html
  - Logstash: https://www.elastic.co/guide/en/logstash/6.6/index.html
  - Kibana: https://www.elastic.co/guide/en/kibana/6.6/index.html
  - Filebeat: https://www.elastic.co/guide/en/beats/filebeat/6.6/index.html

Basic information

- **Official website:** https://www.elastic.co/elk-stack

Installation

The architecture in which the ELK Stack was installed is the following.



Also, it is important to note that the stack of applications was installed on CentOS 7 using Ansible. Therefore, in the next subsections, there will be an explanation of the tasks used to install each of the ELK Stack components.

Before proceeding to the installation of each main component, it is needed to add the ELK's repository to the rpm's repositories.

```
---
- name: Download public signing key
  shell: rpm --import https://artifacts.elastic.co/GPG-KEY-elasticsearch

- name: Add elk repository file
  template:
    src: "etc/yum.repos.d/elk.repo.j2"
    dest: "/etc/yum.repos.d/elk.repo"
```

(continues on next page)

(continued from previous page)

```
    owner: root
    group: root
    mode: 0644
```

This playbook basically adds the ELK's gpg signing key and takes a template to render it in the `/etc/yum/repos.d/` directory, which is where `rpm` looks for its repositories. The template file is this:

```
[elastic-{{ elk_version }}]
name=Elastic repository for {{ elk_version }} packages
baseurl=https://artifacts.elastic.co/packages/{{ elk_version }}/yum
gpgcheck=1
gpgkey=https://artifacts.elastic.co/GPG-KEY-elasticsearch
enabled=1
autorefresh=1
type=rpm-md
```

The `{{ elk_version }}` jinja variable refers to the version of your desired stack. In this case 6.c. This variable must be passed as an argument when running ansible or have it defined somewhere in your ansible project. For more information about variables go to the Ansible's documentation.

Also, it is needed to install Java 8 and the main components (elasticsearch, logstash, kibana) packages.

```
---
- name: Install Java
  yum:
    name: java-{{ java_version }}-openjdk-devel
    state: present

- name: Install logstash-elasticsearch-kibana
  yum:
    name:
      - logstash
      - elasticsearch
      - kibana
    state: present
```

The variable `{{ java_version }}` represents the java version used, in our case (and due to compatibility) 1.8.0.

Elasticsearch

After installing the needed package, Elasticsearch is configured like this:

```
---
- name: Configure elasticsearch
  template:
    src: "etc/elasticsearch/elasticsearch.yml.j2"
    dest: "/etc/elasticsearch/elasticsearch.yml"
    owner: root
    group: root
    mode: 0644
  notify: enable_restart_elasticsearch
```

The Elasticsearch main configuration file, which is a template, is rendered in `/etc/elasticsearch/`. The template can be found `here`. In that template, you will find a variable called `{{ machine }}`, which is rendered as the hostname of our ELK server, in our case *elk*. So in your case, you can use whatever you want, but from now on

in this guide, we will use the hostname *elk*. Also, when the configuration file is placed, a notify is made so that the Elasticsearch service is started/restarted. The handler looks like this:

```
---
- name: enable_restart_elasticsearch
  systemd:
    name: elasticsearch
    state: restarted
    enabled: yes
```

Logstash

After installing the needed package, Logstash is configured like this:

```
---
- name: Configure logstash
  copy:
    src: "etc/logstash/{{ item }}"
    dest: "/etc/logstash/{{ item }}"
    mode: 0644
  with_items:
    - pipelines.yml
    - logstash.yml
  notify: enable_restart_logstash

- name: Logstash main pipeline configuration file
  template:
    src: "etc/logstash/conf.d/main_pipeline.conf.j2"
    dest: "/etc/logstash/conf.d/main_pipeline.conf"
    owner: root
    group: root
    mode: 0644
  notify: enable_restart_logstash
```

The first task copies two configuration files, *pipelines.yml* and *logstash.yml*. The first file indicates to Logstash where to find our pipelines configuration files. You can find it `here`. The second one is the main configuration file for Logstash. You can find it `here`.

The second task takes a template and renders it in the pipelines directory. The template represents the description of our main pipeline, that is, inputs, filters, and outputs. You can find it `here`.

Note: Logstash Filters: It is important to know the version of the filter plugins that you are using so you will be able to search for the proper documentation.

Kibana

After installing the needed package, Kibana is configured like this:

```
---
- name: Configure kibana
  template:
    src: "etc/kibana/kibana.yml.j2"
    dest: "/etc/kibana/kibana.yml"
```

(continues on next page)

(continued from previous page)

```
    owner: root
    group: root
    mode: 0644
  notify: enable_restart_kibana
```

The Kibana main configuration file, which is a template, is rendered in `/etc/kibana/`. The template can be found `here`. Also, when the configuration file is placed, a notify is made so that the Kibana service is started/restarted. The handler looks like this:

```
---
- name: enable_restart_kibana
  systemd:
    name: kibana
    state: restarted
    enabled: yes
```

After installing and configuring Kibana, it is time to give structure to our logs and create/import the dashboards and visualizations needed:

1. Access the web interface through *http://elk:5601*. To access it using the domain name *elk* remember to add elk to your *hosts* file.
2. Organize the information. This will help you plot all your data easily.

   Note: Create the indexes, and the mappings BEFORE sending any data to Elasticsearch.

   a) Create *indexes* and *mappings*, that is, give types and formats to your data.

      - In the *Dev Tools* section, copy and paste the content of the index and mappings `file`, then select it all and click on RUN. Note that these mappings are the ones that we use, you can take them as an example and create yours, for more information go to ELK's documentation about mappings.
      - To easily see your mappings go to: Management -> Index management -> Select your index -> Mapping.

   b) Continue with **c** and **d** steps after *Filebeat* is sending information to logstash. So please go to *Filebeat*.

      - You can check that it is already done if you can create *index patterns*, that is, it won't let you create them if you don't have any data.

   c) Create the dashboard and visualizations.

      - Go to *Management*, then, under the Kibana section go to *Saved Objects*, then, *Import*, and import the dashboards and visualizations `file`.
      - If you want to export the visualizations to a JSON format, remember to export every saved object, because some visualizations may depend on other objects and they won't work if you don't export them all.

   d) In the section *Management -> Index Patterns* select one (no matter which one) index pattern and press the start button to make it the default one.

Filebeat

Remember that in our case Filebeat is installed in servers different from the ELK Server, see *Installation*.

So, the installation playbook looks like this:

```
---
- name: Download public signing key
  shell: rpm --import https://artifacts.elastic.co/GPG-KEY-elasticsearch

- name: Add elk repository file
  template:
    src: "etc/yum.repos.d/elk.repo.j2"
    dest: "/etc/yum.repos.d/elk.repo"
    owner: root
    group: root
    mode: 0644

- name: Install filebeat
  yum:
    name: filebeat
    state: present

- name: Configure filebeat
  template:
    src: etc/filebeat/filebeat.yml.j2
    dest: /etc/filebeat/filebeat.yml
    owner: root
    group: rot
    mode: 0644
  notify: enable_restart_filebeat
```

As previously explained, the three first tasks are for adding the ELK's repository and installing the main component package. The last task is for configuring Filebeat. It takes a template file, which contains the Filebeat main configuration, that is, where it will take the logs from. You can find the template file `here`. Then, after Filebeat is configured, a notification is sent to a handler to start/restart the Filebeat service. The handler looks like this:

```
---
- name: enable_restart_filebeat
  service:
    name: filebeat
    state: restarted
    enabled: yes
```

Testing

If you want to test all the ELK Stack locally you can easily do it using Vagrant.

Note: Elasticsearch and Logstash use together at least 5 GB of RAM when not in idle state.

The vagrantfile looks like this:

```
# -*- mode: ruby -*-
# vi: set ft=ruby :

# All Vagrant configuration is done below. The "2" in Vagrant.configure
# configures the configuration version (we support older styles for
# backwards compatibility). Please don't change it unless you know what
```

(continues on next page)

(continued from previous page)

```
# you're doing.
Vagrant.configure("2") do |config|

  config.vm.define "cr0n05", autostart: false do |cronos|

    cronos.vm.box = "centos/7"
    cronos.vm.network "private_network", ip: "192.168.1.2"
    cronos.vm.hostname = "cr0n05"

    cronos.vm.provider "virtualbox" do |v|
      v.memory = 1024
    end

    cronos.vm.provision "ansible_local" do |ansible|
      ansible.become = true
      ansible.playbook = "site.yml"
      ansible.verbose = "vv"
      ansible.extra_vars = {
        machine: "cr0n05"
      }
    end

  end

  config.vm.define "4p0l0", autostart: false do |apolo|

    apolo.vm.box = "centos/7"
    apolo.vm.network "private_network", ip: "192.168.1.3"
    apolo.vm.hostname = "4p0l0"

    apolo.vm.provider "virtualbox" do |v|
      v.memory = 1024
    end

    apolo.vm.provision "ansible_local" do |ansible|
      ansible.become = true
      ansible.playbook = "site.yml"
      ansible.verbose = "vv"
      ansible.extra_vars = {
        machine: "4p0l0"
      }
    end

  end

  config.vm.define "elk", autostart: false do |elk|

    elk.vm.box = "centos/7"
    elk.vm.network "private_network", ip: "192.168.1.4"
    elk.vm.hostname = "elk"

    elk.vm.provider "virtualbox" do |v|
      v.memory = 4096
    end

    elk.vm.provision "ansible_local" do |ansible|
      ansible.become = true
```

(continues on next page)

(continued from previous page)

```
      ansible.playbook = "site.yml"
      ansible.verbose = "vv"
      ansible.extra_vars = {
        machine: "elk"
      }
    end

  end

end
```

In the configuration of each virtual machine, there is a subsection for provisioning. In that subsection, there is a variable that is accessed as `ansible.playbook`. You have to set it to the path to your ansible playbook. You should use the playbook that was explained in the previous section, *Installation*. Also in this provisioning subsection, note that the `ansible.extra_vars` defines a variable called `machine`, so if you are using the playbook explained before, this variable must match the hostname of the virtual machine. The hostname of the virtual machine can be changed with the variable `vm.hostname`. For more information read the Vagrant documentation about vagrantfiles.

To start up the virtual cluster use the following bash script with the argument `up`:

```
#!/bin/bash

if [ "$1" == "up" ]; then
    vagrant up elk cr0n05 4p0l0 --no-provision
elif [ "$1" == "provision-elk" ]; then
    vagrant provision elk
elif [ "$1" == "provision-filebeat" ]; then
    vagrant provision cr0n05 4p0l0
else
    echo "Usage: ./run.sh up|provision-elk|provision-filebeat"
fi
```

Note: Change elk, cr0n05, 4p0l0, to the virtual machine names that you set up in your Vagrantfile. If you are using the vagrantfile from above, you do not have to change them.

Make the virtual machines visible between them by their hostname. You just have to change the `/etc/hosts` file and add the ip address of the virtual machine that you want to see followed by its hostname. For example, make elk visible by others and in the *elk* machine.

```
# file /etc/hosts
0.0.0.0         elk      # allow others to use the elk hostname instead of the ip
192.168.1.2     cr0n05   # make cr0n05 visible to elk by its hostname not just its ip
192.168.1.3     4p0l0
```

After making them visible, run the script with the argument `provision-elk` so that Elasticsearch, Logstash, and Kibana will be installed. Configure Kibana as explained in *Kibana*. Then run the script with the argument `provision-filebeat`. When it finishes you should be able to open your browser in the elk machine's ip address port 5601.

Authors

- Hamilton Tobon Mosquera <htobonm@eafit.edu.co>

ELK Stack 7.x

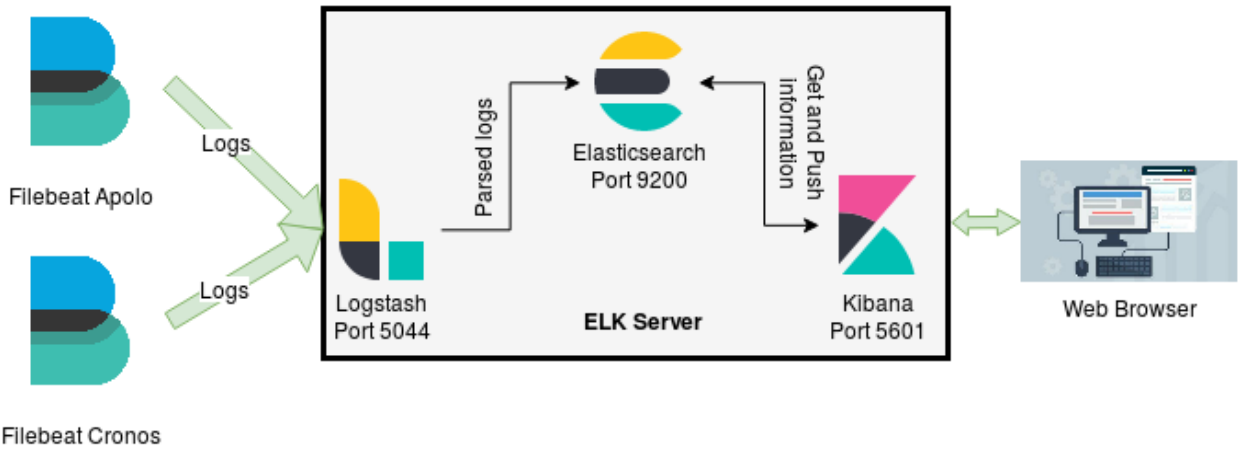
Installation and Configuration

Contents

Note:

- It is **very important** to make the cluster nodes visible between them by their DNS. Otherwise, the services might not properly start. If this happens, dig into the log files and find out the problem.
- Correctly configure the date and time of the cluster. Otherwise, visualizing the logs in Kibana might be a problem. This due that the log timestamp will be on a different date of that of the Kibana server.
- Install Java JDK 8, more recent versions may have compatibility problems.
- Use the same version for Logstash, Elasticsearch, Kibana, and Filebeat to avoid compatibility problems.
- Also, when reading the guides check that the guide version is compatible with the version of the ELK stack in use.
- In case of having more than 500MB in logs, the RAM used by Logstash and Elasticsearch is something to consider. By default 1GB for heap, plus the jvm (which runs some other things), so more or less 2.0 - 3.0 GB of RAM. Therefore ensure at least 4-6GB of RAM (even more when having gigabytes of logs) in the server that Elasticsearch and Logstash will be installed.
- If RAM is not a problem, and there is a need for more RAM on Elasticsearch or Logstash, increase the heap size by modifying the `jvm.options` file, which is usually located under `/etc/<service>` or under `/usr/share/<service>/config`. Look for the options `-Xms1g` and `-Xmx1g` and change the *1g* to an the value needed. The chosen value should be the same in both options, for more information about the heap read here.
- Install, configure, and start the services in the following order, then repeating some steps will be avoided, as well as some problems. Note that this order is the same as the one in the Ansible role `elk`.
  - Elasticsearch: https://www.elastic.co/guide/en/elasticsearch/reference/7.1/index.htm
  - Logstash: https://www.elastic.co/guide/en/logstash/7.1/index.html
  - Kibana: https://www.elastic.co/guide/en/kibana/7.1/index.html
  - Filebeat: https://www.elastic.co/guide/en/beats/filebeat/7.1/index.html
- If any service fails or is working in an unexpected way check the logs. They are usually under `/usr/share/<service>/logs` or under `/var/log/<service>`.

The architecture in which the ELK Stack was installed is the following.



Although we are using this architecture, it may be modificated in any way, just pay close attention to the configuration files. Also, it is important to note that the stack of applications was installed on CentOS 7 using Ansible. Therefore, in the next subsections, there will be an explanation of the Ansible roles used to install and configure each of the ELK Stack components. The directory structure of the Ansible project is described `here`.

1. Before proceeding to the installation of each main component, it is needed to add the ELK's repository to the rpm's repositories.

```
# roles/elk/tasks/add-elk-repo.yml

---
- name: Download public signing key
  shell: rpm --import https://artifacts.elastic.co/GPG-KEY-elasticsearch

- name: Add elk repository file
  template:
    src: "etc/yum.repos.d/elk.repo.j2"
    dest: "/etc/yum.repos.d/elk.repo"
    owner: root
    group: root
    mode: 0644
```

This playbook basically adds the ELK's gpg signing key to rpm and takes a template, `roles/elk/templates/etc/yum.repos.d/elk.repo.j2`, to render it in the `/etc/yum/repos.d/` directory, which is where `rpm` looks for its repositories. The template file is this:

```
# roles/elk/templates/etc/yum.repos.d/elk.repo.j2

[elastic-{{ elk_version }}]
name=Elastic repository for {{ elk_version }} packages
baseurl=https://artifacts.elastic.co/packages/{{ elk_version }}/yum
gpgcheck=1
gpgkey=https://artifacts.elastic.co/GPG-KEY-elasticsearch
enabled=1
autorefresh=1
type=rpm-md
```

The `{{ elk_version }}` jinja variable refers to the version of the desired stack. In this case 7.x. This variable must be passed as an argument when running Ansible or have it defined somewhere in the Ansible project. For more information about variables go to the Ansible's documentation.

1. Also, it is needed to install Java 8 and the main components (Elasticsearch, Logstash, Kibana) packages.

```
# roles/elk/tasks/install-elk.yml

---
- name: Install Java
  yum:
    name: java-{{ java_version }}-openjdk-devel
    state: present

- name: Install logstash-elasticsearch-kibana
  yum:
    name:
      - logstash
      - elasticsearch
      - kibana
    state: present
```

The variable `{{ java_version }}` represents the java version used, in this case (and to ensure compatibility) 1.8.0.

Elasticsearch

After installing the needed package, Elasticsearch is configured like this:

```
# roles/elk/tasks/config.yml

---
- name: Configure elasticsearch
  template:
    src: "etc/elasticsearch/elasticsearch.yml.j2"
    dest: "/etc/elasticsearch/elasticsearch.yml"
    owner: root
    group: root
    mode: 0644
  notify: enable_restart_elasticsearch
```

The Elasticsearch main configuration file, which is the template `roles/elk/templates/etc/elasticsearch/elasticsearch.yml.j2`, is rendered in `/etc/elasticsearch/`. The template can be found `here`. In that template, there is a variable called `{{ machine }}`, which is rendered as the hostname of the ELK server, in this case `elk`. It can be changed as needed, but from now on in this guide, `elk` is what will be used. Note that, when the configuration file is placed, a notify is made so that the Elasticsearch service is started/restarted. The notify handler looks like this:

```
# roles/elk/handlers/main.yml

---
- name: enable_restart_elasticsearch
  systemd:
    name: elasticsearch
    state: restarted
    enabled: yes
```

Note: Note that the identifier given to the notify action in the configuration task must be same as the identifier given to the handler.

Logstash

After installing the needed package, Logstash is configured like this:

```
# roles/elk/tasks/config.yml

---
- name: Configure logstash
  copy:
    src: "etc/logstash/{{ item }}"
    dest: "/etc/logstash/{{ item }}"
    mode: 0644
  with_items:
    - pipelines.yml
    - logstash.yml
  notify: enable_restart_logstash

- name: Logstash main pipeline configuration file
  template:
    src: "etc/logstash/conf.d/main_pipeline.conf.j2"
    dest: "/etc/logstash/conf.d/main_pipeline.conf"
    owner: root
    group: root
    mode: 0644
  notify: enable_restart_logstash
```

The first task copies two configuration files, `pipelines.yml`, and `logstash.yml` to `/etc/logstash`. The former indicates to Logstash where to find our pipelines configuration files. It can be found `here`. The latter is the main configuration file for Logstash. It can be found `here`.

The second task takes a template, `roles/elk/templates/etc/logstash/conf.d/main_pipeline.conf.j2`, and renders it in the pipelines directory, `/etc/logstash/conf.d`. The template represents the description of the main pipeline, that is, inputs, filters, and outputs. It can be found `here`.

Note: **Logstash Filters:** It is important to know the version of the filter plugins being used, so that the proper documentation can be found.

Kibana

After installing the needed package, Kibana is configured like this:

```
# roles/elk/tasks/config.yml

---
- name: Configure kibana
  template:
    src: "etc/kibana/kibana.yml.j2"
    dest: "/etc/kibana/kibana.yml"
    owner: root
    group: root
    mode: 0644
  notify: enable_restart_kibana
```

The Kibana main configuration file, which is the template `roles/elk/templates/etc/kibana/kibana.yml.j2`, is rendered in `/etc/kibana/`. The template can be found `here`. Also, when the configuration file is

placed, a notify is made so that the Kibana service is started/restarted. The notify handler looks like this:

```
# roles/elk/handlers/main.yml

---
- name: enable_restart_kibana
  systemd:
    name: kibana
    state: restarted
    enabled: yes
```

Indexes and Mappings

After installing and configuring Kibana, it is time to give structure to our logs and create/import the dashboards and visualizations needed:

1. Access the web interface through *http://elk:5601*. To access it using the domain name `elk` remember to make the cluster nodes visible by their DNS', in this case, the node where Kibana is installed.
2. Organize the information. This will help plotting all the data easily. To add a new logging source see the section below *Adding a new Logging Source*.

> **Warning:** Create the indexes and the mappings BEFORE sending any data to Elasticsearch, otherwise, data would end up in unexpected indexes.

a) Create *indexes* and *mappings*, that is, give types and format to the data.

- In the **Dev Tools** section, copy and paste the contents of the index and mappings `file`, then select it all and click on RUN. These mappings can be used as a reference to create more mappings.
- To easily see the mappings go to: **Management -> Index management -> Click on the desired index -> Mapping**.

b) Create the dashboard and visualizations.

- Go to the **Management** section, then, go to the subsection **Saved Objects**, then, **Import**, and import the dashboards and visualizations `file`.

> **Note:** To export the visualizations and dashborads to a json format, remember to check the *every saved object* option. Otherwise some visualizations may depend on other objects that might not be exported, ending up with errors when importing them again.

c) In the section **Management -> Kibana -> Index Patterns** select one (no matter which one) index pattern and press the star button to make it the default one.

Filebeat

Recall that in this case Filebeat is installed in nodes different from the ELK Server, the architecture used here is on *Installation and Configuration*.

So, the installation playbook looks like this:

```
# roles/master/tasks/config.yml

---
- name: Download public signing key
  shell: rpm --import https://artifacts.elastic.co/GPG-KEY-elasticsearch

- name: Add elk repository file
  template:
    src: "etc/yum.repos.d/elk.repo.j2"
    dest: "/etc/yum.repos.d/elk.repo"
    owner: root
    group: root
    mode: 0644

- name: Install filebeat
  yum:
    name: filebeat
    state: present

- name: Configure filebeat
  template:
    src: etc/filebeat/filebeat.yml.j2
    dest: /etc/filebeat/filebeat.yml
    owner: root
    group: rot
    mode: 0644
  notify: enable_restart_filebeat
```

As previously explained, the three first tasks are for adding the ELK's repository and installing the main component package. The last task is for configuring Filebeat. It takes a template file that contains the Filebeat main configuration, basically, where it will take the logs from. The template file can be found `here`. Then, after Filebeat is configured, a notify is sent to a handler to start/restart the Filebeat service. The handler looks like this:

Note: Note that the folder under `roles/` is not `elk/` anymore. It is `master/` because the idea is to install the log collector in the master nodes, which then, will start sending logs to the `elk` server.

```
# roles/master/handlers/main.yml

---
- name: enable_restart_filebeat
  service:
    name: filebeat
    state: restarted
    enabled: yes
```

Make sure that the paths to the logs given in `filebeat.yml` are correct. Everythin is correct if data can be seen in Kibana. Go to the **Discover** section, select some index pattern and select a time window, something similar to 1 month ago, or change it as needed. This time window represents the time range that Kibana uses to query the logs to Elasticsearch. For example, if to analyze the last year logs, choose **1 year ago -> now**.

Testing

Testing of the whole ELK Stack can be easily done using Vagrant.

Note: Recall that when having a relatively large amount of logs, Elasticsearch and Logstash use about 4-6GB RAM (or even more) when filtering and indexing data.

1. The vagrantfile looks like this:

```
# -*- mode: ruby -*-
# vi: set ft=ruby :

# All Vagrant configuration is done below. The "2" in Vagrant.configure
# configures the configuration version (we support older styles for
# backwards compatibility). Please don't change it unless you know what
# you're doing.
Vagrant.configure("2") do |config|

  config.vm.define "cr0n05", autostart: false do |cronos|

    cronos.vm.box = "centos/7"
    cronos.vm.network "private_network", ip: "192.168.1.2"
    cronos.vm.hostname = "cr0n05"

    cronos.vm.provider "virtualbox" do |v|
      v.memory = 1024
    end

    cronos.vm.provision "ansible_local" do |ansible|
      ansible.become = true
      ansible.playbook = "site.yml"
      ansible.verbose = "vv"
      ansible.extra_vars = {
        machine: "cr0n05",
        elk_ip: "192.168.1.4",
        elk_hostname: "elk"
      }
    end

  end

  config.vm.define "4p0l0", autostart: false do |apolo|

    apolo.vm.box = "centos/7"
    apolo.vm.network "private_network", ip: "192.168.1.3"
    apolo.vm.hostname = "4p0l0"

    apolo.vm.provider "virtualbox" do |v|
      v.memory = 1024
    end

    apolo.vm.provision "ansible_local" do |ansible|
      ansible.become = true
      ansible.playbook = "site.yml"
      ansible.verbose = "vv"
      ansible.extra_vars = {
        machine: "4p0l0",
        elk_ip: "192.168.1.4",
        elk_hostname: "elk"
      }
```

(continues on next page)

(continued from previous page)

```
    end

  end

  config.vm.define "elk", autostart: false do |elk|

    elk.vm.box = "centos/7"
    elk.vm.network "private_network", ip: "192.168.1.4"
    elk.vm.hostname = "elk"

    elk.vm.provider "virtualbox" do |v|
      v.memory = 4096
    end

    elk.vm.provision "ansible_local" do |ansible|
      ansible.become = true
      ansible.playbook = "site.yml"
      ansible.verbose = "vv"
      ansible.extra_vars = {
        machine: "elk"
      }
    end

  end

end
```

For the purpose of this guide use the Ansible project `here`. This project is explained in *Installation and Configuration*. Although, in case of setting up another configuration, read the explanation below of the Vagrantfile above so that it can be replicated.

- In the configuration of each virtual machine, there is a subsection for provisioning. In that subsection, there is a variable that is accessed as `ansible.playbook`. Set it to the path of the main ansible playbook.
- Take a look at the provisioning subsection in the vagrantfile, note that the `ansible.extra_vars` defines a variable called `machine`, this variable must match the hostname of the virtual machine.
- The hostname of the virtual machine can be changed with the variable `vm.hostname`. For more information, read the Vagrant documentation about vagrantfiles.
- The variables `elk_ip` and `elk_hostname` under the configuration of `4p0l0` and `cr0n05`, are used to make `elk` visible by its hostname automatically.

The `site.yml` uses one playbook or another depending on the value of the variable `machine`:

```
# site.yml
---
- import_playbook: "playbooks/{{ machine }}.yml"
```

The `playbooks/4p0l0.yml`, `playbooks/cr0n05.yml`, and `playbooks/elk.yml` playbooks are simple too:

```
# playbooks/4p0l0.yml
---
- hosts: 4p0l0
  roles:
    - ../roles/master
```

```
# playbooks/cr0n05.yml
---
- hosts: cr0n05
  roles:
    - ../roles/master
```

```
# playbooks/elk.yml
---
- hosts: elk
  roles:
    - ../roles/elk
```

The roles `elk` and `master` are responsible for setting up ELK and Filebeat respectively. Go to *Installation and Configuration*, for a more detailed explanation.

- Before starting the virtual cluster please see the directory structure that should be matched in order to run the tests `here`.

2. To start up the virtual cluster use the following bash script:

```
#!/bin/bash

if [ "$1" == "up" ]; then
    vagrant up elk cr0n05 4p0l0 --no-provision
elif [ "$1" == "provision-elk" ]; then
    vagrant provision elk
elif [ "$1" == "provision-filebeat" ]; then
    vagrant provision cr0n05 4p0l0
elif [ "$1" == "halt" ]; then
    vagrant halt elk cr0n05 4p0l0
else
    echo "Usage: ./run.sh up|provision-elk|provision-filebeat"
fi
```

From the root of the project run:

```
$ ./scripts/run.sh up
```

3. Now provision `elk`, run:

```
$ ./scripts/run.sh provision-elk
```

After correctly provisioning `elk`, set up the *Indexes and Mappings* in Kibana.

> **Warning:** Before provisioning filebeat it is very important to set up the *indexes and mappings* in Kibana.

4. After setting up Kibana run:

```
$ ./scripts/run.sh provision-filebeat
```

5. If everything is ok, new logging sources can be addded, as well as, create visualizations and dashboards, etc.
6. To stop the cluster run:

```
$ ./scripts/run.sh halt
```

Adding a new Logging Source

Adding a new logging source is simple if the ELK stack is alreary installed. Follow the following steps:

Note: Even though this guide uses Filebeat as log collector, the principles should be the same for others as well.

Contents

1. Logging Source

1. Identify the logging source. For example, it required to track the users that log in to an SSH server. Identify where the service writes its logs. In this case `/var/log/secure`.
2. Knowing where to take the logs from, identify the messages that are useful. For example:

```
Jun 25 15:30:02 elk sshd[5086]: pam_unix(sshd:session): session closed for user␣
↪vagrant
Jun 25 17:08:11 elk sshd[5185]: Accepted publickey for vagrant from 10.0.2.2 port␣
↪54128 ssh2: RSA SHA256:64u6q4IdjxSFhVGdqwJa60y/nMx7oZWb0dAsNqMIMvE
Jun 25 17:08:11 elk sshd[5185]: pam_unix(sshd:session): session opened for user␣
↪vagrant by (uid=0)
```

The first and third logs might not be useful in this case, but the second log is the one that helps.

2. Filtering

Now it's time to parse and filter the important information. It can be accomplished using Logstash and the Kibana's Grok debugging tool. Grok is a Logstash filtering plugin used to match patterns and extract useful information from the logs. For more information about Grok read the documentation. Follow these steps:

1. Open Kibana. Go to **Dev Tools -> Grok Debugger**. Here can be found three main text boxes:

- The first one is where to put the log that will be filtered.
- The second one is where a regular expression is written. This regular expression tells Grok how to filter the log.
- The third one is where Grok shows its results in JSON format, which is the format used by Elasticsearch to inxed everything.

2. But, how to filter an arbitrary log?. Grok uses a regular expression library called Oniguruma. This library has a way match patterns in the log. These patterns can be tagged with a name. That name is important because that is how the

information will be found in Elasticsearch. Here is the regular expression that matches the `timestamp`, the `event` state (if the user could or couldn't log in), the `user` that tried to log in, the `ip address` that is trying to log in, the `port` number and the user `signature`. Check out the following regular expression:

```
%{SYSLOGTIMESTAMP:log_timestamp} %{SYSLOGHOST:system_hostname} sshd(.*)?: (?
↪<sshd_event>[a-zA-Z]+) %{DATA:sshd_method} for (invalid user )?%{DATA:sshd_
↪user} from %{IPORHOST:sshd_guest_ip} port %{NUMBER:sshd_guest_port} ssh2(:
↪%{GREEDYDATA:sshd_guest_signature})?
```

With the log in the first text box and the regular expression in the second text box, press **Simulate**. Up to this point the Kibana's Grok debugger should look like this:

3. Let's break down the regular expression into chunks:

   - **%{SYSLOGTIMESTAMP:log_timestamp}**: *SYSLOGTIMESTAMP* is a Grok built-in regular expression. These and many more built-in regular expressions can be found in this repository. **log_timestamp** is how was decided to *tag* the matched string. Therefore, this expression will match from **Jun ...** to **... 17:08:11**.
   - **%{SYSLOGHOST:system_hostname}**: *SYSLOGHOST* matches the log hostname and identifies it as **system_hostname**. Note that this is the sshd server's hostname, not the user's hostname.
   - **sshd(.*)?**: This expression matches the literal string 'sshd', followed by anything except new lines (the dot) '.'. The parentheses are grouping operators, therefore, they group the expression '.*', and this whole expression is optional, '?', which means it might or might not appear in the log. In other words, there might not be something after the word 'sshd', if so, then it won't match anything. Note that this expression doesn't have any identifier, that's because what's matched here is not important.
   - **(?<sshd_event>[a-zA-Z]+)**: This is an important expression. The expression '(?<xxx>...)' can be used when there isn't a default Grok pattern for what is needed. Instead of 'xxx', type the name/tag that will be given to the matched string. Instead of '...' put the regular expression that matches the needed string. In this case, the **event** is composed only by letters, so '[a-zA-Z]' means any lowercase or uppercase letter, the '+' means one or more times. This expression can be replaced by the Grok default pattern **%{DATA:sshd_event}**, but for the purpose of this guide, '(?<xxx>...)' was used so that it can be used whenever needed.
   - **%{DATA:sshd_method}**: *DATA* matches anything (but new lines). The key is that this *anything* may or may not appear, in other words, it's optional. But **sshd_method** is always needed, why to let it as optional?. Well, it's just for simplicity, instead of creating a new regular expression it's simpler to just use the built-in **%{DATA:...}**.
   - **(invalid user)?**: If the event is 'Invalid' instead of 'Accepted' or 'Failed' this string appears, so that's why it is optional.
   - **%{DATA:sshd_user}**: *DATA* matches anything (but new lines), but that anything may or may not appear.

- **%{IPORHOST:sshd_guest_ip}**: *IPORHOST* matches IP addresses, including IPv6. That IP address is given the identifier *sshd_guest_ip*.
- **%{NUMBER:sshd_guest_port}**: *NUMBER* matches numbers, in this case, the client's port number.
- **(: %{GREEDYDATA:sshd_guest_signature})?**: *GREEDYDATA* matches anything (but new lines). In this case, it matches the guest signature, but sometimes it might not appear, so that's why it is enclosed in an optional construct '(...)?'.
- The other expressions, 'sshd', 'for', 'from', 'port', and 'ssh2' are literal strings, so Grok has to find them in the string that is being parsed, **otherwise the whole string is rejected**.

4. Already having a way to parse the new log, it's time to change the Logstash pipeline configuration. Before proceeding, it's recommended to read this short guide about how a pipeline configuration file looks. Also, it would be very useful to read about what is the purpose of Logstash. Go to the end of the *filter* section and add the following:

```
if [fromsecure] {

}
```

If this this logging source, `/var/log/secure`, was added before, don't add that `if` sentence, surely it is somewhere else in the *filter* section. But, why `[fromsecure]`?, what does that mean?. It checks if the JSON received has a field called `fromsecure`. The existence of that field will be explained later in *4. Adding the log path to Filebeat*.

5. Under the `if` sentence add a `grok` block. This is the way of asking to Logstash to use a *filter* plugin, in this case *Grok*. So, add the following:

```
grok {
  match => {
    "message" => []
  }
  add_field => {}
}
```

The `match` and `add_field` sub-blocks ask Grok to use those options. The `match` option is used to parse fields, what was explained two subsections before. Those fields are passed to the `filter` section by the `input` section, which in turn receives messages from a *Filebeat* service, or a Dead letter queue. The `add_field` adds fields to the JSON message in case that the match option successfully matched a string. This is useful in the `output` section of the pipeline. This is useful to send to Elasticsearch only what was successfully parsed, and not everything that arrives at the `input` section.

6. Under the *match* sub-block and the brackets, and between double quotes, add the regular expression built with the Kibana's Grok debugger. Under the `add_field` sub-block add the following too:

```
grok {
  match => {
    "message" => [
      "%{SYSLOGTIMESTAMP:log_timestamp} %{SYSLOGHOST:system_hostname} sshd(.
↪*)?: (?<sshd_event>[a-zA-Z]+) %{DATA:sshd_method} for (invalid user )?%
↪{DATA:sshd_user} from %{IPORHOST:sshd_guest_ip} port %{NUMBER:sshd_guest_
↪port} ssh2(: %{GREEDYDATA:sshd_guest_signature})?"
    ]
  }
  add_field => {
    "type" => "secure_sshd_login_attempt"
    "secure_correctly_filtered" => "true"
```

(continues on next page)

(continued from previous page)

```
  }
}
```

The `type` field serves to differentiate logs in the same *index* in Elasticsearch. For example, `/var/log/secure` also stores logs about the system security (e.g who executes sudo commands), not only logs about ssh. The `secure_correctly_filtered` is used in the `output` section to send only the information that was correctly filtered.

7. The following filter plugin is **extremely important** to correctly visualize the information. Kibana uses a **metafield**, called `@timestamp`, to organize and show the information based on dates. Logstash adds that field by default when a log is received in the `input` section. The problem is that the `log_timestamp` field that we added before has a different date, it has the timestamp that corresponds to the log creation. The time when the log arrives to Logstash is likely to be very different from the time that the log was generated by the service (in this case sshd). There might be a difference of months, even years, because the log that is being indexed might be from the last month/year. To solve this problem Logstash has a plugin called `date`. This plugin can be used to replace the information in the metafield `@timestamp` with any other field that has a timestamp, in this case `log_timestamp`. It has more options than the two presented here. The basic usage is the following:

```
date {
  match => [ "log_timestamp", "MMM dd yyyy HH:mm:ss", "MMM d yyyy HH:mm:ss",
↪"MMM dd HH:mm:ss", "MMM d HH:mm:ss" ]
  timezone => "America/Bogota"
}
```

The `match` option tells the plugin to parse the field in the first string given in the array, `log_timestamp`. The following strings are the format in which the field to parse might be built. For example, "MMM dd yyyy HH:mm:ss", means that the `log_timestamp` field might be in the format: Three letter month, MMM. A two digit day, dd. A four digit year, yyyy. A two digit hour, HH. A two digit minutes, mm. And a two digit seconds, ss. The rest of the options tells to the plugin that the `log_timestamp` field might have those variants.

The `timezone` option tells the plugin to update the timezone in the `@timestamp` field to the given timezone. Elasticsearch uses UTC as timezone. It cannot be changed, that is, Elasticsearch uses it to work properly. Even though we cannot change it, we can update the `@timestamp` field with our real timezone because Kibana converts it underneath to the browser's timezone. Therefore, it is important to have the **same timezone in the browser and in the logs**.

Note: This plugin is **used by Grok only** in case of successful parse of the log.

---

8. The following filter plugin is used to remove unnecessary fields from the JSON that will be sent to Elasticsearch. This is how to use it:

```
mutate {
    remove_field => ["fromsecure", "log_timestamp"]
}
```

The `remove_field` option is given a list of fields that will be removed.

- The `fromsecure` field is used in the `if` sentence above, so it's not needed anymore. The procedence of this field is explaned later in *4. Adding the log path to Filebeat*.
- The `log_timestamp` is not needed anymore because we already have a field that contains the a timestamp, `@timestamp`.

9. Up to this point there is no need for more Logstash filters. Putting everything together should look like this:

```
# /etc/logstash/conf.d/main_pipeline.conf

input {
  beats {
    port => "5044"
  }

  # Events go to the dead_letter_queue when Elasticsearch response has 400/402 code
  dead_letter_queue {
    path => "/var/log/logstash/dead_letter_queue"
    tags => ["recovered_from_dead_letter_queue"]
  }
}

filter {
  if [fromsecure] {

    grok {
      match => {
        "message" => [
            # Login attemps
          "%{SYSLOGTIMESTAMP:log_timestamp} %{SYSLOGHOST:system_hostname} sshd(.*)?: %{DATA:sshd_event} %{DATA:sshd_method} for (invalid user )?%{DATA:sshd_user} from %{IPORHOST:sshd_guest_ip} port %{NUMBER:sshd_guest_port} ssh2(: %{GREEDYDATA:sshd_guest_signature})?"
          ]
      }
      add_field => { "type" => "secure_sshd_login_attempt"
                     "secure_correctly_filtered" => "true" }
    }

    # In case of successful parsing, the @timestamp field is updated with the parsed info
    date {
      match => [ "log_timestamp", "MMM dd yyyy HH:mm:ss", "MMM d yyyy HH:mm:ss", "MMM dd HH:mm:ss", "MMM d HH:mm:ss" ]
      timezone => "America/Bogota"
    }

    mutate {
      remove_field => ["fromsecure", "log_timestamp"]
    }
  }
}

output {
  if [secure_correctly_filtered] {
    elasticsearch { # Dead Letter Queue is only supported for elasticsearch output
      index => "secure"
      hosts => ["elk:9200"]
    }
  }
}
```

In summary:

- The first section, `input`, indicates to Logstash where it will receive logs from. In this case Filebeat, on port 5044, and something called the **Dead Letter Queue**. This is where logs that couldn't be indexed go. For example, Logstash received a log, but Elasticsearch crashed, so the log couldn't be indexed, then the log is written to the Dead Letter Queue allowing it to be reindexed later.
- The last block, `output`, indicates to Logstash where it will send logs to. In this case Elasticsearch, which is in the host `elk` on port `9200`, to the **index** `secure`. Elasticsearch indexes will be explained in *3. Creating Indexes and Mappings*, think about them as tables where the logs will be registered.
- Note the `if` sentence in line 42. Recall the `add_field` option explained in the Grok filter, well it is used here to send logs to the proper index if and only if, they were correctly filtered by Grok.

10. Restart the Logstash service and hopefully, everything will work perfectly. Sometimes, the service seems to start correctly but it failed reading the pipeline configuration file (what was just written). To check that everything is perfect check out the log when Logstash is starting, commonly `/usr/share/logstash/logs/logstash-plain.log`. Logs similar to these are a good signal:

```
[2019-07-03T10:04:46,238][INFO ][logstash.agent           ] Pipelines␣
↪running {:count=>1, :running_pipelines=>[:main], :non_running_pipelines=>
↪[]}
[2019-07-03T10:04:46,705][INFO ][org.logstash.beats.Server] Starting server␣
↪on port: 5044
[2019-07-03T10:04:50,337][INFO ][logstash.agent           ] Successfully␣
↪started Logstash API endpoint {:port=>9600}
```

3. Creating Indexes and Mappings

Indexes are used by Elasticsearch to store the information sent by Logstash. Mappings are a way to structure that data using a JSON format. Let's see an example to structue the log parsed above, for more information about mappings read here:

```
PUT /secure
{
  "mappings":{
    "properties":{
      "type": { "type" : "keyword" },
      "system_hostname":{ "type": "keyword" },
      "sshd_guest_ip":{ "type": "ip" },
      "sshd_guest_port":{ "type": "integer" },
      "sshd_guest_signature":{ "type": "text" },
      "sshd_event":{ "type": "keyword" },
      "sshd_method":{ "type": "keyword" },
      "sshd_user":{ "type": "keyword" }
    }
  }
}
```

Elasticsearch offers a REST API to manage the data. So, *PUT* inserts new information into Elasticsearch. Therefore, if there exists an index with the name *secure*, Elasticsearch will throw an error. In that case use *POST*, which is used to update the existing information. So, what does all that stuff mean?:

- "`mappings`" refers to the property that describes the structure of the index.
- "`properties`" as its name says, is used to describe the properties of the mappings.
- The rest of the items are the fields and its types. These fields describe the types of the information parsed in Logstash. For example:

- "sshd_guest_ip" is the field that represents the ip address parsed from the logs. Its type is "ip". Elasticsearch has a built-in type called "ip" which eases the indexation and visualization of ip addresses.
- The "type" field is useful to differentiate the logs sent from a single source, in this case /var/log/secure. Recall the add_field option under the Grok plugin in *2. Filtering*, it was added the field: "type" => "sshd_login_attempt". Therefore, in case of indexing the sudo commands logs, change this field to something like: "type" => "secure_sudo_command". This is how to differentiate them easily.

4. Adding the log path to Filebeat

Now that the data is filtered and properly structured, it's time to start sending it to Logstash. Go to the machine that has the Filebeat service, edit the file /etc/filebeat/filebeat.yml. Under the section filebeat.inputs: add:

```
- type: log
  paths:
    - /var/log/secure*
  fields:
    fromsecure: true
  fields_under_root: true
```

What does it mean?:

- The first line indicates the type of information that will be collected.
- The second line indicates the paths where the new logging source is located, in this case /var/log/, and secure* matches all the logs that start with the name *secure*. This wildcard is used becase some logs have a date at the end of its name, so it will be painful to add over and over again a path when a log appears in /var/log/.
- The fourth line, fields, indicates to Filebeat to add a new field in to the JSON sent to Logstash. Recall the first if sentence in the *2. Filtering* section. Well, this field is added so that all the different logging sources can be differentiated in Logstash.
- The last option, fields_under_root, is used to add the fields under the root of the JSON, and not nested into a field called beat, which is the default behavior.

Restart the Filebeat service and hopefully everything will work perfectly. Otherwise, recall to check the logs usually under /usr/share/<service>/logs or under /var/log/<service>.

5. Create Index Patterns

With some data indexed in Elasticsearch, create **Index Patterns**. These are used by Kibana to match (using regular expressions) indexes and take the data that will be plotted from those indexes matched by some pattern.

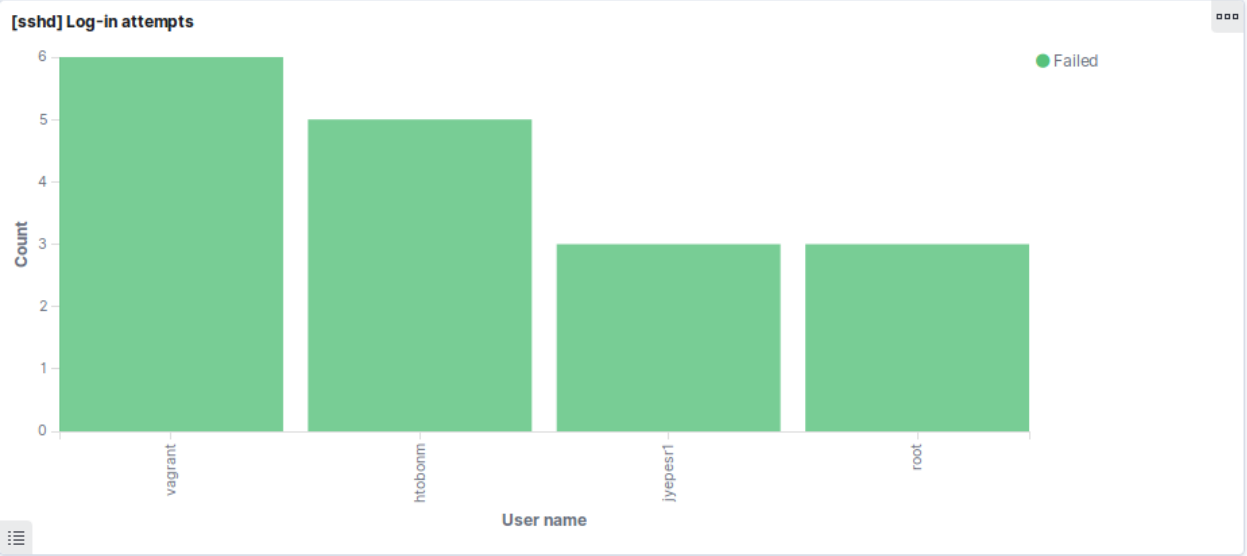
Go to **Management -> Index Pattern -> Create index pattern**. Select its name/pattern, and as time filter field select "@timestamp".

6 . Plot the data

One of the easiest plots that can be created is a frequency histogram. Nevertheless, there are lots of more features that Kibana offers.

In Kibana go to **Visualize**, press the **+** button, select the type of visualization, in this case, **Vertical Bar**. Afther this, select the index pattern that corresponds to the **secure** logs. Then, to create a frequency historgram of the users that failed logging in follow these steps:

1. In the left hand side of the Kibana web page, there is a subsection called **Buckets**. Click on **X-Axis**.
2. As aggregation select **Terms**. For more information about Term aggregation.
3. As field select **sshd_user**.
4. As custom label write: User name.
5. Now instead of **X-Axis** select **Add sub-buckes**. Then select **Split Series**.
6. Here as aggregation select **Terms** again.
7. As field select **sshd_event**.
8. Now type the following in the bar that is in the upper part of the Kibana's GUI, the **Filters** bar: `sshd_event : "Failed"`. This is called **Kibana Query Language**, it can be used to filter the data and plot only what is be useful. More information on this query language here, Kibana Query Language.
9. Click on the **play** button in the left hand side of the Kibana's GUI.
10. Save the visualization with a descriptive name, something like: *[sshd] Failed attempts to log in*.
11. In case of not having a **Dashboard**, create a new one, then add the visualization. Up to this point it should look something like:



Basic information

- **Official website:** https://www.elastic.co/elk-stack

Authors

- Hamilton Tobon Mosquera <htobonm@eafit.edu.co>

3.6.5 Grafana

Grafana allows you to query, visualize, alert on and understand your metrics no matter where they are stored.[1]

Grafana is an open source metric analytics & visualization suite. It is most commonly used for visualizing time series data for infrastructure and application analytics but many use it in other domains including industrial sensors, home automation, weather, and process control.[2]

Grafana 5.4

This documentation page describes the process of installation and configuration of Grafana using CentOS 7 as the base Operating System.

Contents

Directory Hierarchy

Description	Path
Installs binary	/usr/sbin/grafana-server
init.d script	/etc/init.d/grafana-server
Default Configuration	/etc/sysconfig/grafana-server
Configuration file	/etc/grafana/grafana.ini
Default Log file	/var/log/grafana/grafana.log
Database definition	/var/lib/grafana/grafana.db
Systemd Service definition	/usr/lib/systemd/system/grafana-server.service

[1] Grafana Labs. (n.d.). Grafana - The open platform for analytics and monitoring. Retrieved June 20, 2019, from https://grafana.com/
[2] Grafana Labs. (n.d.). Grafana Documentation. Retrieved June 20, 2019, from https://grafana.com/docs/v4.3/

Note: Default configuration specifies an sqlite3 database.

Note: The installation creates the grafana-server service in /usr/lib/systemd only if systemd is available.[1]

Basic information

- **Deploy date:** 11th February, 2019
- **Official Website:** https://grafana.com
- **License:** Apache License 2.0

Concepts

- **Example:** ABC.

Installation

1. Download the Official RPM:

```
$ wget https://dl.grafana.com/oss/release/grafana-5.4.3-1.x86_64.rpm.
```

2. Install the downloaded RPM:

```
$ yum install grafana-5.4.3-1.x86_64.rpm
```

Configuration

HTTP Configuration

Instead of running Grafana Server directly in the port 80, it can be provided through port 80 or 443 using a Web server or a reverse proxy.

The procedure to configure it with apache is:

1. Define the Location configuration that will stablish a reverse proxy to redirect HTTP requests from port 80 to local port 3000 (default port for Grafana) in a valid configuration directory (i.e `/etc/httpd/conf.d/grafana.conf`)

```
<Location "/grafana">
  ProxyPreserveHost On
  ProxyPass http://10.10.10.10:3000
  ProxyPassReverse http://10.10.10.10:3000
</Location>
```

1. Enable Apache to redirect Grafana App port

[1] Grafana Labs. (n.d.). Installing on RPM-based Linux. Retrieved June 20, 2019, from http://docs.grafana.org/installation/rpm/

```
$ setsebool -P httpd_can_network_connect 1
```

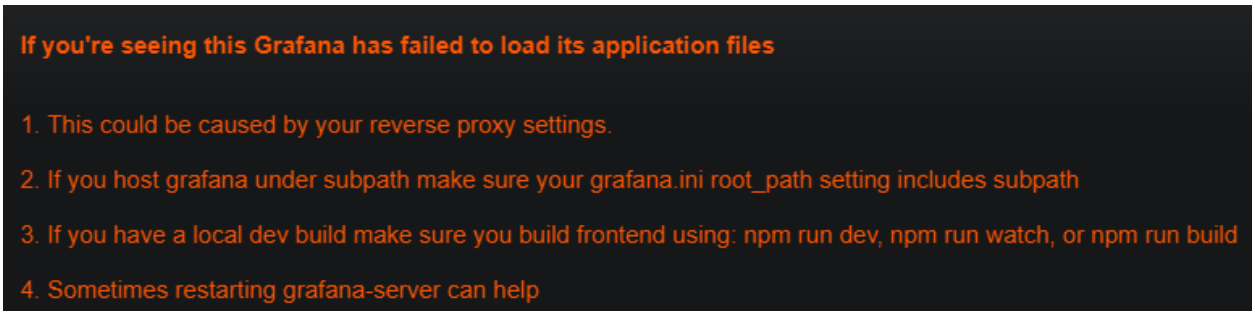
2. Enable Grafana Service to start in boot time

```
$ systemctl enable grafana-server
```

Plugins

Troubleshooting

If you're seeing this Grafana has failed to load its application files



PROBLEM 2: Using a web server (i.e Apache) to provide the grafana web interface through a public IP generates an incorrect redirection with the default values present in `grafana.ini`.

SOLUTION: Edit the variable `root_url` in `/etc/grafana/grafana.ini` as follows:

```
root_url = %(protocol)s://%(domain)s:%(http_port)s/grafana
```

Restart grafana-server to refresh the configuration:

```
$ systemctl grafana-server restart
```

Basic Authentication Failed

PROBLEM: The Login process fails in Grafana Webpage displaying the following JSON:

```
{"message":"Basic auth failed"}
```

SOLUTION: Disable Basic Authentication in `/etc/grafana/grafana.ini`:

```
[auth.basic]
enabled = false
```

Restart grafana-server to refresh the configuration:

```
$ systemctl grafana-server restart
```

Cannot Bind to Port

PROBLEM: The variable *http_port* is set with a reserved port (lower than 1024) but grafana-server cannot start correctly.

> **Warning:** It's not recommended to run Grafana in a reserved port. Instead use a Reverse Proxy or a web server (Apache, Nginx).

SOLUTION: Grafana Server is a service that runs as grafana user, it's neccesary to add it the linux capability to bind to reserved ports running with a non-root user.

Starting with kernel 2.2, Linux divides the privileges traditionally associated with superuser into distinct units, known as capabilities, which can be independently enabled and disabled. Capabilities are a per-thread attribute.[2]

Execute:

```
$ setcap 'cap_net_bind_service=+ep' /usr/sbin/grafana-server
```

Authors

- Andrés Felipe Zapata Palacio <azapat47@eafit.edu.co>

References

3.6.6 Apolo Monitoring

Apolo monitoring is a web application designed to monitor the different resources consumed by the jobs running in the Apolo cluster. The resources monitored by the web application are the following: Percentage of cores availability per node, amount of RAM consumed and amount of SWP memories consumed. It also has a log of jobs in execution with their respective information, it has sections of history of percentage of utilization updated every 24 hours, percentage of utilization updated every hour. Apolo monitoring is structured to add new functionalities.

Contents

[2] Kerrisk, M. (2019, May 11). Capabilities Manual. Retrieved June 20, 2019, from http://man7.org/linux/man-pages/man7/capabilities.7.html

Basic information and dependencies

- **Deploy date:** 05th May, 2021
- **Tested in:** Apolo test environment
- **Installed in:** Apolo II
- **Programming language:** PHP 7.4
- **Composer:** Composer 2.0.12
- **Framework:** Laravel 5.8
- **Data Base:** Mongo DB
- **Web server:** Apache2
- **DNS server:** Bind9
- **Continuous improvement and deployment:** Travis CI
- **External files:** Scripts sh

¿What is PHP?

PHP (recursive acronym for PHP: Hypertext Preprocessor) is a very popular open source language especially suitable for web development and can be embedded in HTML.

PHP 7.4 Installation and its dependencies

Steps to install php 7.4

```
sudo apt update

sudo apt install apache2

sudo systemctl start apache2.service

sudo systemctl enable apache2.service

sudo apt-get install software-properties-common
```

(continues on next page)

(continued from previous page)

```
sudo add-apt-repository ppa:ondrej/php

sudo apt update

sudo apt-get install php7.4 libapache2-mod-php7.4 php7.4-cli php7.4-mysql php7.4-gd␣
↪php7.4-imagick php7.4-recode php7.4-tidy php7.4-xmlrpc

php -v
```

¿What is Composer?

Composer is a package manager for PHP that provides a standard for managing, downloading and installing dependencies and libraries. Similar to NPM in Node.js and Bundler in Ruby, Composer is the ideal solution when working on complex projects that rely on multiple installation sources. Instead of having to download each dependency manually, Composer does this automatically for us.

Composer installation

1. Installation of dependencies

It will install the dependencies. We will need curl to download Composer and php-cli to install and run it. The php-mbstring package is needed to provide functions for a library we will use. Composer uses git to download project dependencies and unzip to extract compressed packages. It is possible to install everything with the following command:

```
sudo apt update

sudo apt install curl php-cli php-mbstring git unzip
```

2. Downloading and installing Composer

Composer provides an installer written in PHP. We will download it, check that it is not corrupted and use it to install Composer.

```
cd ~

curl -sS https://getcomposer.org/installer -o composer-setup.php
```

Next, verify that the installer matches the SHA-384 hash for the most recent installer found on the Composer Public Keys/Signatures page. Copy the hash from that page and store it as a shell variable:

```
HASH=544e09ee996cdf60ece3804abc52599c22b1f40f4323403c44d44fdfdd586475ca9813a858088ffbc1f233e9b180f061
```

Now, run the following PHP script to verify that the installation script runs safely:

```
php -r "if (hash_file('SHA384', 'composer-setup.php') === '$HASH') { echo 'Installer␣
↪verified'; } else { echo 'Installer corrupt'; unlink('composer-setup.php'); } echo␣
↪PHP_EOL;"
```

You will see the following result:

```
Installer verified
```

Warning: If you see the Installer corrupt message, you will need to double-check if you used the correct hash and re-download the installation script. Then run the command to verify the installer again. Once you have a verified installer, you can continue.

3. To install composer globally, use the following command which will download and install it system-wide as a command named composer, in /usr/local/bin:

```
sudo php composer-setup.php --install-dir=/usr/local/bin --filename=composer
```

To check your installation, run the following:

```
composer
```

You will see that the Composer version and arguments are displayed in this output.

```
Output
   ______
  / ____/___  ____ ___  ____  ____  ________  _____
 / /   / __ \/ __ `__ \/ __ \/ __ \/ ___/ _ \/ ___/
/ /___/ /_/ / / / / / / /_/ / /_/ (__  )  __/ /
\____/\____/_/ /_/ /_/ .___/\____/____/\___/_/
                    /_/
Composer version 2.0.12 2021-03-03 11:44:59

Usage:
  command [options] [arguments]

Options:
  -h, --help                     Display this help message
  -q, --quiet                    Do not output any message
  -V, --version                  Display this application version
      --ansi                     Force ANSI output
      --no-ansi                  Disable ANSI output
  -n, --no-interaction           Do not ask any interactive question
      --profile                  Display timing and memory usage information
      --no-plugins               Whether to disable plugins.
  -d, --working-dir=WORKING-DIR  If specified, use the given directory as working␣
↪directory.
  -v|vv|vvv, --verbose           Increase the verbosity of messages: 1 for normal␣
↪output, 2 for more verbose output and 3 for debug
. . .
```

¿What is Laravel?

Laravel is one of the easiest to assimilate open source frameworks for PHP. It is simple, very powerful and has an elegant and fun to use interface.

Create a laravel project

```
composer create-project laravel/laravel="5.8.*" NameProject
```

To start a Laravel project locally we execute:

```
cd NameProject

php artisan serve
```

¿What are migrations in Laravel?

In Laravel, it is said that migrations are a version control of our database, but in reality they are more than that. It allows us to create tables, establish relationships, modify them and of course delete them, and all this through the command console.

Creating a migration

First we will have to open the console or command line and position ourselves in the path of our project. After that, execute the following command to create our first migration.

```
php artisan make:migration create_users_table
```

¿What are controllers in laravel?

Controllers are a mechanism that allows us to group related HTTP request logic and thus better organize our code.

Generate a controller

We generate a new controller with the Artisan command make:controller passing it the name we want to give it. In the example the name is UserController:

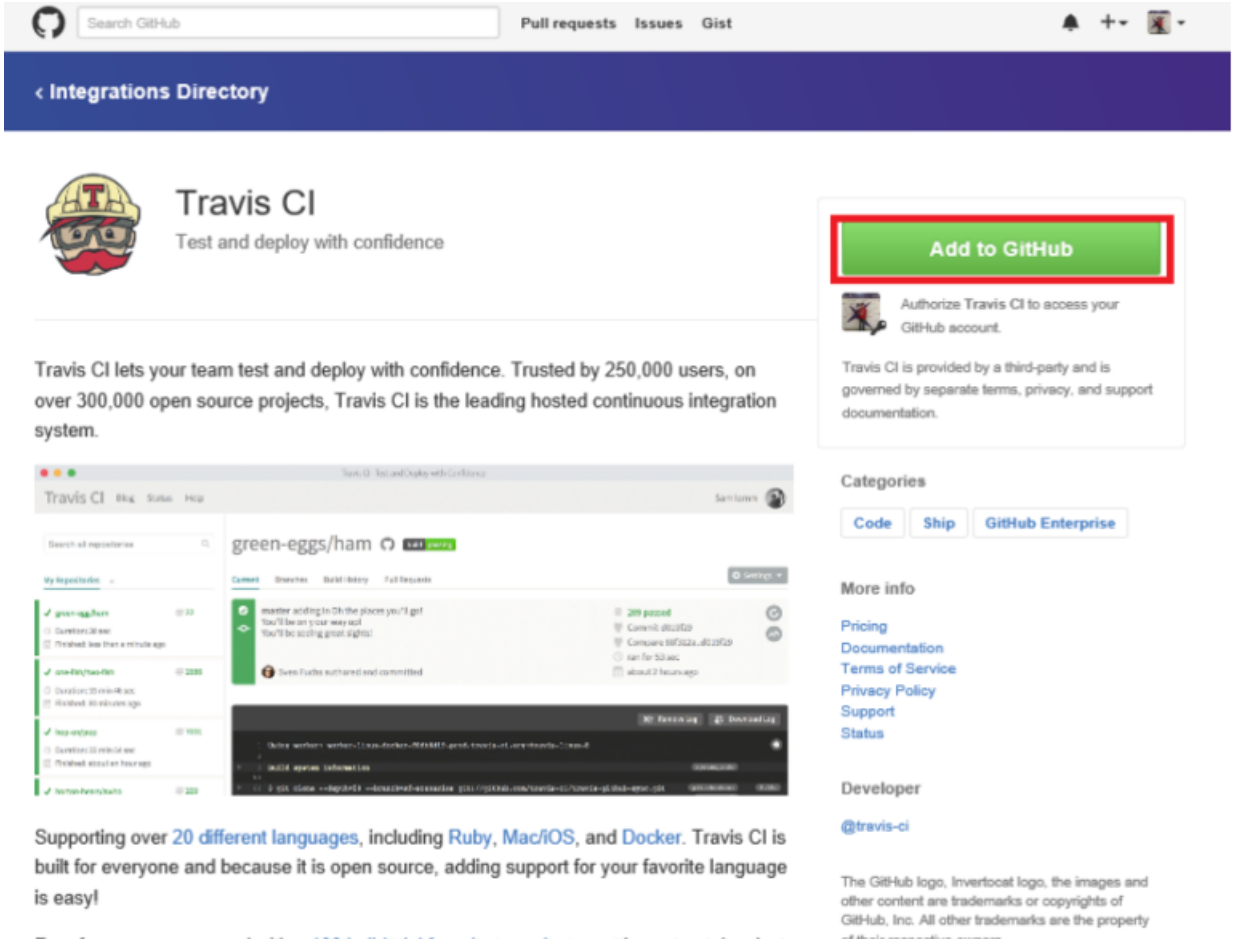
```
php artisan make:controller UserController
```

¿What is travis-CI?

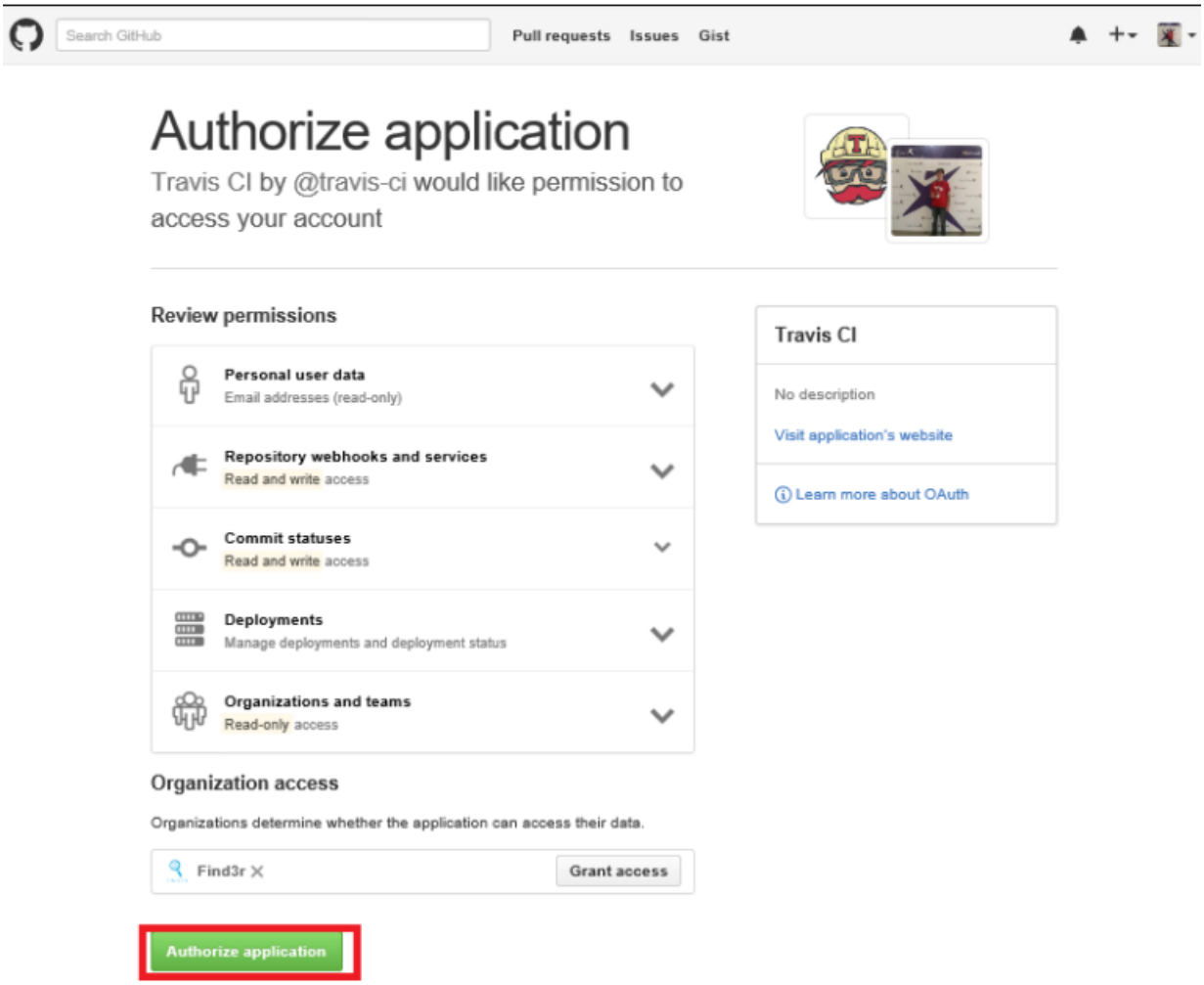
Travis-CI is a Continuous Integration system, free for Open Source projects and paid for private projects. It integrates seamlessly with GitHub and automatically executes the pipeline defined in each push or pull request. It tests and builds applications written in Ruby, Node, Objective-C, Go, Java, C#, F# and PHP, among others (running on Linux).

Steps to follow to use Travis-CI

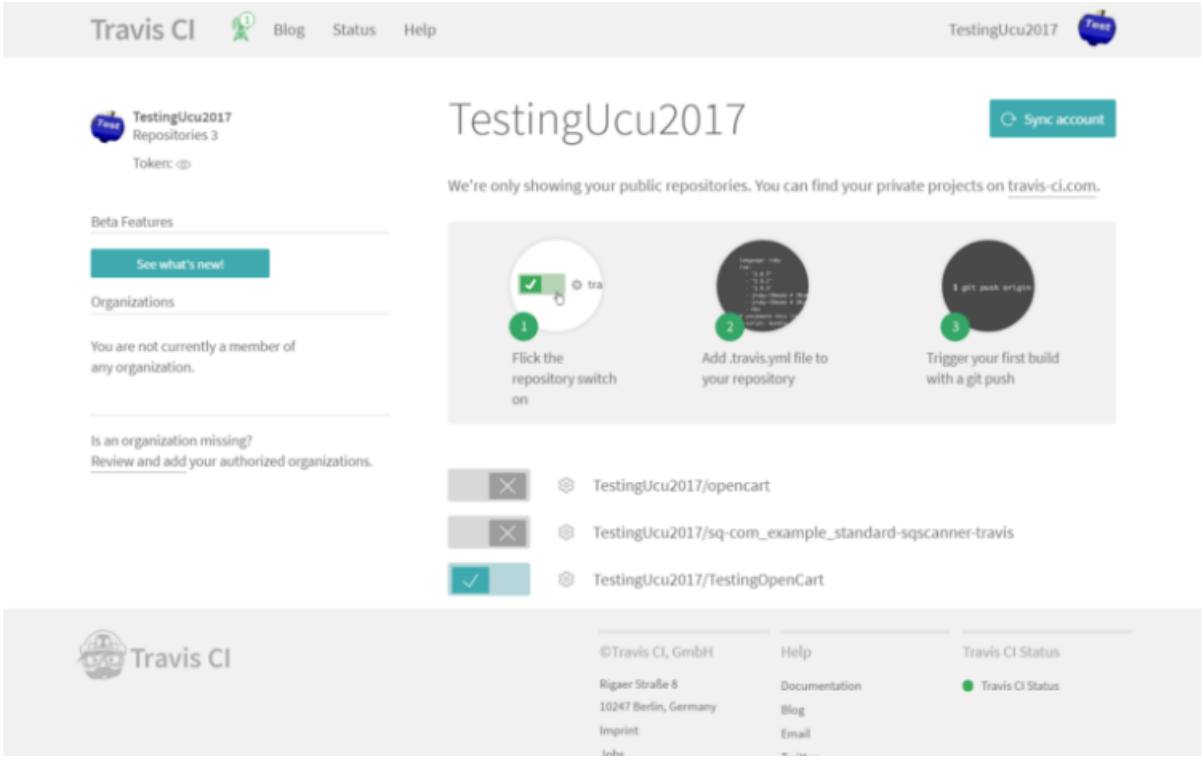
1. The first step is to create a GitHub repository containing the code of the application we want to incorporate into Travis-CI.
2. Once the repository is created, we must give permissions to Travis to connect to it. To do this we must go to the repository settings, click on "Services" and select Travis-CI. Once there select "Add to GitHub".



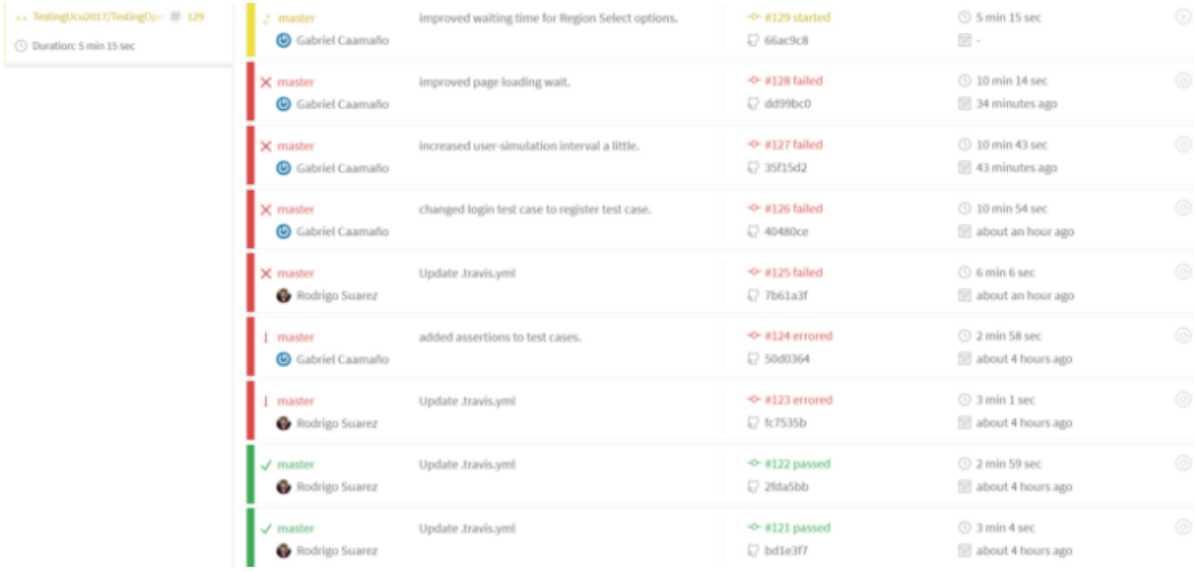
2. After this, on the next page we must grant all required permissions to Travis-CI.



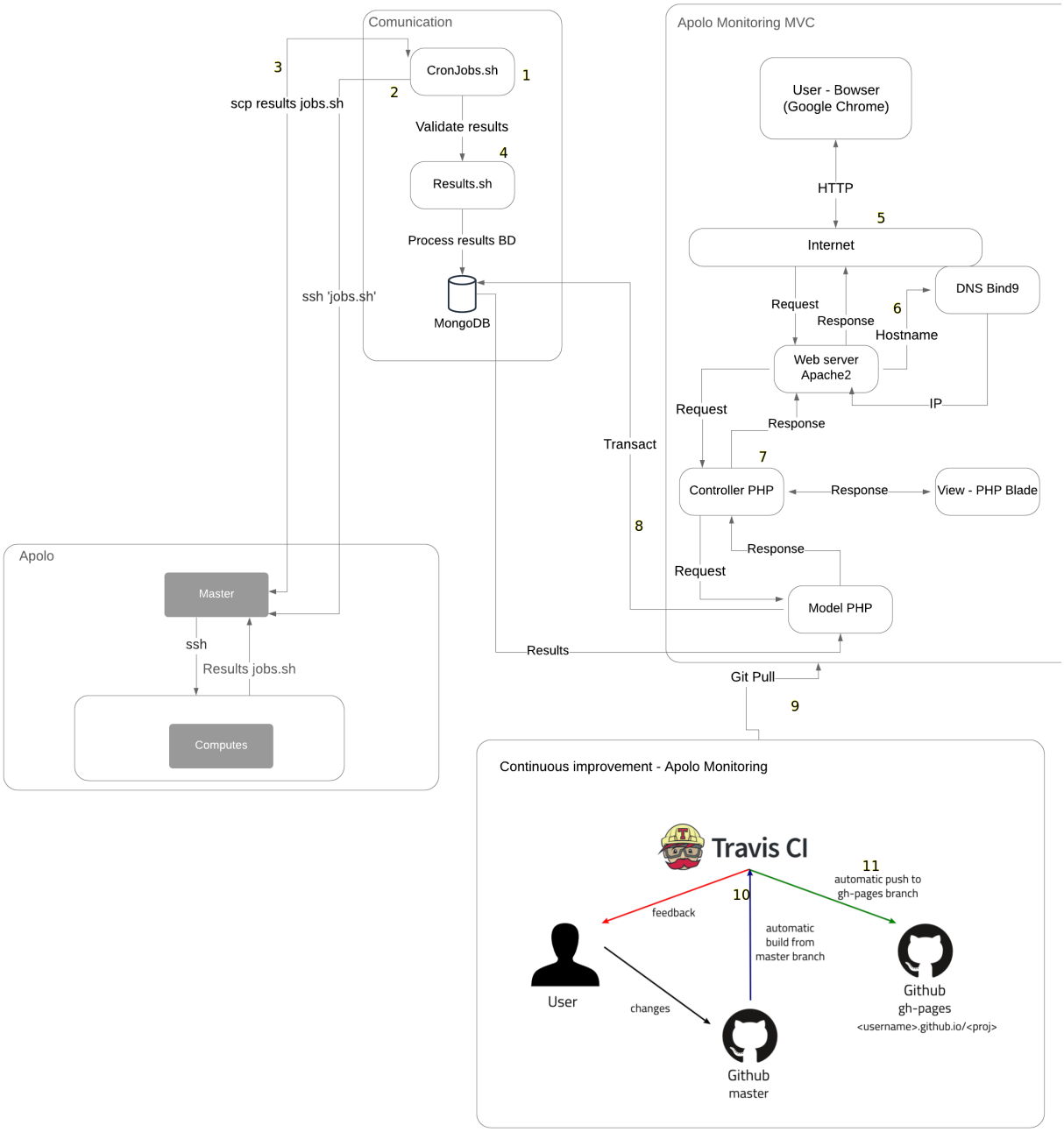
3. Once in Travis-CI, select the "+", find the repository you want to work on and click on the switch.



4. Go back to the GitHub repository and create the .travis.yml file. This file should be hosted in the root of the project. It will tell Travis-CI what to do every time it runs a Build.
5. If the Build is satisfactory, Travis-CI will display the results in green, otherwise the results will be red.



Technology diagram of Apolo Monitoring



1. First we have the communication, which is the basis for the whole system built, here we have the cron table where several srcipts.sh are executed at certain times.
2. When any of the scripts in the cron table is executed, it connects to the Apolo master through SSH to execute scripts that are specifically programmed to obtain information such as: cluster utilization percentage, RAM memory consumed, SWP memory consumed, core availability percentage and running jobs information.
3. Once the information is ready in the Apolo master, another cron is executed to SCP the application home with the information generated in the previous step.
4. At the end of the SCP, another cron is run to review the downloaded data and then inject the information into

the database.

5. When an Apolo administrator wants to use the web application, he/she will have to search from a web browser (Google Chrome), hostname to be able to use it.
6. When looking up the web application in the browser, the Internet will communicate with the web server, then the web server will communicate with the DNS server to resolve the name of the web application and give a response.
7. After the DNS server gives its response, a request will be made to the controller from the web server, then a request will be made to the view, thus returning the view to the user.
8. If the view needs any query in the database, simply from the model it communicates with the database, to return the data needed by the view.
9. If you want to add a new functionality to the web application, you must make its respective unit tests to git push to the remote repository.
10. Upon git push to the remote repository, Travis CI will go live and make the decision whether the changes are right or wrong.
11. If they are correct, all that needs to be done is a git pull to upload the changes to the repository in which it is deployed.

Authors

- Bryan López Parra <blopezp@eafit.edu.co>

References

- GitHub: Where the world builds software. (2021). Retrieved 11 March 2021, from https://github.com/
- Slurm Workload Manager - Documentation. (2021). Retrieved 11 March 2021, from https://Slurm.schedmd.com/documentation.html
- MySQL. (2021). Retrieved 11 March 2021, from https://www.mysql.com/
- Laravel - The PHP Framework For Web Artisans. (2021). Retrieved 28 July 2021, from https://laravel.com/

3.7 Operating Systems

Our supercomputers (*Apolo II* and *Cronos*) are built with the operating system Rocks 6.2 which is based on CentOS 6.6.

3.7.1 Rocks

Rocks is an open-source Linux cluster distribution that enables end users to easily build computational clusters, grid endpoints and visualization tiled-display walls. Hundreds of researchers from around the world have used Rocks to deploy their own cluster.[1]

[1] Rocks Cluster - Open Source Toolkit for Real and virtual Clusters. Retrieved February 1, 2019, from http://www.rocksclusters.org/

Rocks 6.2

Table of Contents

Basic information

- **Official Website:** http://www.rocksclusters.org/downloads/2015-05-11-download-rocks-6-2-sidewinder.html
- **License:** http://www.rocksclusters.org/docs/licensing.html
- **Installed on:** *Apolo II* , *Cronos*

Add NAS Appliance

This section describes the required steps in Rocks to add a new NAS system appliance.

```
# The first part will configure an IB interface for the NAS

rocks add host nas-1 membership='NAS Appliance' cpus=48 rack=0 rank=1
rocks add host interface nas-1 iface=ib1
rocks set host interface subnet nas-1 iface=ib1 subnet=mpi-nfs
rocks set host interface ip nas-1 iface=ib1 ip=<nas ip>
rocks set host interface mac nas-1 iface=ib1 mac=<mac>
rocks set host interface module nas-1 iface=ib1 module=ib_ipoib
rocks set host attr nas-1 arch x86_64
rocks set host attr nas-1 os linux
rocks set host attr nas-1 managed false
rocks set host attr nas-1 kickstartable false
rocks set host installaction nas-1 action=os
rocks set host boot nas-1 action=os
rocks set attr Info_HomeDirSrv nas-1.mpi-nfs
rocks sync config

# This second part will configure a second interface for the NAS (admin network)

rocks add network admin subnet=<subnet ip> netmask=<mask> dnszone=<domain> mtu=1500␣
↪servedns=False
rocks add host interface nas-1 iface=em3
rocks set host interface mac nas-1 iface=em3 mac=<mac>
rocks set host interface ip nas-1 iface=em3 ip=<ip>
rocks set host interface subnet nas-1 iface=em3 subnet=admin
```

Troubleshooting

This section is intended to be a quick reference for the most common errors produced in Rocks and its solutions made by us.

Infiniband

Issue 1: Permanent connected-mode If we add the `CONNECTED_MODE=yes` parameter to ib0 interface configuration, it works well but this parameter is removed when the command `rocks sync network` is run, therefore the connected mode is disabled.

Solution There is an official way to enable the connected mode using the parameter `SETIPOIBCM=yes` on (/etc/infiniband/openib.conf) file if you are using Mellanox OFED.

3.8 Scientific Libraries

3.8.1 ParallelNetCDF

Table of Contents

Description

Parallel netCDF (PnetCDF) is a parallel I/O library for accessing NetCDF files in CDF-1, 2, and 5 formats. The CDF-5 file format, an extension of CDF-2, supports more data types and allows users to use 64-bit integers to define large dimensions, attributes, variables (> 2B array elements).[1]

ParallelNetCDF 1.6.1

Table of Contents

[1] http://cucis.ece.northwestern.edu/projects/PnetCDF/

Basic information

- **Official Website:** http://cucis.ece.northwestern.edu/projects/PnetCDF/
- **License:** Copyright (c) 2003 Northwertern University and Argonne National Laboratory. All rights reserved.
- **Installed on:** Apolo II and Cronos
- **Installation date:** 13/02/2018

Tested on (Requirements)

- **OS base:** CentOS (x86_64) $\geq$ 6.6 (Rocks 6.2)
- **Dependencies:**
  - GNU GCC >= 5.5.0
  - MPICH2 >= 3.2.1

Installation

1. Load required modules.

```
module purge
module load gcc/5.5.0
module load mpich2/3.2.1_gcc-5.5.0
```

2. Configure the following environment variables that specify the compilers to use:

```
$ export F77=mpif90
$ export CFLAGS='-O2 -fPIC'
$ export CXXFLAGS='-O2 -fPIC'
$ export FFLAGS='-fPIC'
$ export FCFLAGS='-fPIC'
$ export FLDFLAGS='-fPIC'
$ export F90LDFLAGS='-fPIC'
$ export LDFLAGS='-fPIC'
```

3. Download the desired version of the software (Source code - tar.gz)

```
cd /home/mgomezzul/apps/pnetcdf/src/
wget http://cucis.ece.northwestern.edu/projects/PnetCDF/Release/parallel-
↪netcdf-1.6.1.tar.gz
tar -xvf parallel-netcdf-1.6.1.tar.gz
```

4. After unpacking Parallel NetCDF, continue with the following steps for configuration and compilation:

```
cd parallel-netcdf-1.6.1

./configure --prefix=/share/apps/pnetcdf/1.6.1/gcc-5.5.0_mpich2-3.2.1 --
↪build=x86_64-redhat-linux --enable-fortran --enable-largefile --with-
↪mpi=/share/apps/mpich2/3.2.1/gcc-5.5.0 2&1 | tee pnetcdf-conf.log
```

(continues on next page)

(continued from previous page)

```
make -j 10 2>&1 | tee pnetcdf-make.log
make check 2>&1 | tee pnetcdf-make-check.log
sudo mkdir -p /share/apps/pnetcdf/1.6.1/gcc-5.5.0_mpich2-3.2.1
sudo chown -R mgomezzul.apolo /share/apps/pnetcdf/1.6.1/gcc-5.5.0_mpich2-
↪3.2.1
make install 2>&1 | tee pnetcdf-make-install.log
sudo chown -R root.root /share/apps/pnetcdf/1.6.1/gcc-5.5.0_mpich2-3.2.1
```

5. Generate the module with the following environment variable:

```
setenv          PNET                    $topdir
sudo moduleGenerator
```

Module

```
#%Module1.0#####################################################################
↪##
##
## module load pnetcdf/1.6.1_gcc-5.5.0_mpich2-3.2.1
##
## /share/apps/modules/pnetcdf/1.6.1_gcc-5.5.0_mpich2-3.2.1
## Written by Mateo Gómez-Zuluaga
##

proc ModulesHelp {} {
    global version modroot
    puts stderr "Sets the environment for using pnetcdf-1.6.1\
                \nin the shared directory \
                \n/share/apps/pnetcdf/1.6.1/gcc-5.5.0_mpich2-3.2.1\
                \nbuilded with gcc-5.5.0 and mpich2-3.2.1."
}

module-whatis "(Name________) pnetcdf"
module-whatis "(Version_____) 1.6.1"
module-whatis "(Compilers___) gcc-5.5.0_mpich2-3.2.1"
module-whatis "(System______) x86_64-redhat-linux"
module-whatis "(Libraries___) "

# for Tcl script use only
set         topdir        /share/apps/pnetcdf/1.6.1/gcc-5.5.0_mpich2-3.2.1
set         version       1.6.1
set         sys           x86_64-redhat-linux

conflict pnetcdf
module load mpich2/3.2.1_gcc-5.5.0

setenv          PNET                    $topdir

prepend-path    PATH                    $topdir/bin

prepend-path    LD_LIBRARY_PATH         $topdir/lib
prepend-path    LIBRARY_PATH            $topdir/lib
prepend-path    LD_RUN_PATH             $topdir/lib
```

(continues on next page)

(continued from previous page)

```
prepend-path    C_INCLUDE_PATH          $topdir/include
prepend-path    CXX_INCLUDE_PATH        $topdir/include
prepend-path    CPLUS_INCLUDE_PATH      $topdir/include
```

Use

```
module load pnetcdf/1.6.1_gcc-5.5.0_mpich2-3.2.1
```

Resources

- http://cucis.ece.northwestern.edu/projects/PnetCDF/download.html

Author

- Andrés Felipe Zapata Palacio

Resources

3.8.2 NetCDF

Description

NetCDF (network Common Data Form) is a set of software libraries and machine-independent data formats that support the creation, access, and sharing of array-oriented scientific data. Distributions are provided for Java and C/C++/Fortran. See the netCDF web site and the FAQ answer to How do I get the netCDF software package? for more information.

C Version

NetCDF 4.3.3

Table of Contents

Basic information

- **Official Website:** https://www.unidata.ucar.edu/software/netcdf/
- **License:** Copyright University Corporation for Atmospheric Research/Unidata.
- **Installed on:** Apolo II and Cronos
- **Installation date:** 13/02/2018

Tested on (Requirements)

- **OS base:** CentOS (x86_64) $\geq$ 6.6 (Rocks 6.2)
- **Dependencies:**
  - mpich2 v3.2.1
  - HDF5 v1.8.15-patch-1
  - PNetCDF/1.6.1

Installation

1. Load required modules.

```
module load gcc/5.5.0
module load mpich2/3.2.1_gcc-5.5.0
module load hdf5/1.8.15-patch1_gcc-5.5.0_mpich2-3.2.1
module load pnetcdf/1.6.1_gcc-5.5.0_mpich2-3.2.1
```

2. Configure the following environment variables that specify the compilers to use:

```
export CC=mpicc
export CXX=mpicxx
export FC=mpif90
export F77=mpif90
export F90=mpif90
export CPPFLAGS="-I$HDF5/include -I$PNET/include"
export CFLAGS="-I$HDF5/include -I$PNET/include"
export CXXFLAGS="-I$HDF5/include -I$PNET/include"
export FCFLAGS="-I$HDF5/include -I$PNET/include"
export FFLAGS="-I$HDF5/include -I$PNET/include"
export LDFLAGS="-L$HDF5/lib -L$PNET/lib"
HDF5=/
PNET=/
```

3. Download the desired version of the software (Source code - tar.gz)

```
cd /home/mgomezzul/apps/netcdf/src/
wget ftp://ftp.unidata.ucar.edu/pub/netcdf/netcdf-4.3.3.tar.gz
tar -xvf netcdf-4.3.3.tar.gz
```

4. After unpacking NetCDF, continue with the following steps for configuration and compilation:

```
cd netcdf-4.3.3

./configure --prefix=/share/apps/netcdf/4.3.3/gcc-5.5.0_mpich2-3.2.1 --
↪build=x86_64-redhat-linux --enable-large-file-tests --enable-parallel-
↪tests --enable-largefile

make -j 10 2>&1 | tee netcdf-make.log
make check 2>&1 | tee netcdf-make-check.log
sudo mkdir -p /share/apps/netcdf/4.3.3/gcc-5.5.0_mpich2-3.2.1
sudo chown -R mgomezzul.apolo /share/apps/netcdf/4.3.3/gcc-5.5.0_mpich2-3.
↪2.1
make install 2>&1 | tee netcdf-make-install.log
sudo chown -R root.root /share/apps/netcdf/4.3.3/gcc-5.5.0_mpich2-3.2.1
```

5. Generate the module with the following environment variable:

```
setenv          PNET                    $topdir
sudo moduleGenerator
```

Module

```
##%Module1.0#####################################################################
↪##
##
## module load netcdf/4.3.3_gcc-5.5.0_mpich2-3.2.1
##
## /share/apps/modules/netcdf/4.3.3_gcc-5.5.0_mpich2-3.2.1
## Written by Mateo Gómez-Zuluaga
##

proc ModulesHelp {} {
    global version modroot
    puts stderr "Sets the environment for using netcdf-4.3.3\
                \nin the shared directory /share/apps/netcdf/4.3.3/gcc-5.5.0_
↪mpich2-3.2.1\
                \nbuilded with gcc-5.5.0, mpich2-3.2.1, hdf5-1.8.15-patch1,␣
↪pnetcdf/1.6.1."
}

module-whatis "(Name________) netcdf"
module-whatis "(Version_____) 4.3.3"
module-whatis "(Compilers___) gcc-5.5.0, mpich2-3.2.1"
module-whatis "(System______) x86_64-redhat-linux"
module-whatis "(Libraries___) zlib-1.2.11, szip-2.1.1, hdf5-1.8.15-patch1,␣
↪pnetcdf-1.6.1"

# for Tcl script use only
set         topdir        /share/apps/netcdf/4.3.3/gcc-5.5.0_mpich2-3.2.1
set         version       4.3.3
set         sys           x86_64-redhat-linux

conflict netcdf
module load mpich2/3.2.1_gcc-5.5.0
module load hdf5/1.8.15-patch1_gcc-5.5.0_mpich2-3.2.1
module load pnetcdf/1.6.1_gcc-5.5.0_mpich2-3.2.1
```

(continues on next page)

(continued from previous page)

```
setenv          NETCDF                  $topdir

prepend-path    PATH                    $topdir/bin

prepend-path    LD_LIBRARY_PATH         $topdir/lib
prepend-path    LIBRARY_PATH            $topdir/lib
prepend-path    LD_RUN_PATH             $topdir/lib

prepend-path    C_INCLUDE_PATH          $topdir/include
prepend-path    CXX_INCLUDE_PATH        $topdir/include
prepend-path    CPLUS_INCLUDE_PATH      $topdir/include

prepend-path    PKG_CONFIG_PATH         $topdir/lib/pkgconfig

prepend-path    MANPATH                 $topdir/share/man
```

Use

```
module load netcdf/4.3.3_gcc-5.5.0_mpich2-3.2.1
```

Resources

- ftp://ftp.unidata.ucar.edu/pub/netcdf/
- https://www.unidata.ucar.edu/software/netcdf/docs/copyright.html

Author

- Andrés Felipe Zapata Palacio

NetCDF 4.5.0

Table of Contents

Basic information

- **Official Website:** https://www.unidata.ucar.edu/downloads/netcdf/index.jsp
- **License:** MIT open source
- **Installed on:** Apolo II and Cronos
- **Installation date:** 14/11/2017

Tested on (Requirements)

- **OS base:** CentOS (x86_64) $\geq$ 6.6 (Rocks 6.2)
- **Dependencies:**
  - Intel Parallel Studio XE Cluster Edition >= 17.0.1
  - hdf5 >= 1.8.19
  - zlib >= 1.2.11
  - szip >= 2.1

Installation

1. Download the desired version of the software (Source code - tar.gz)

```
cd /home/mgomezzul/apps/netCDF/intel
wget ftp://ftp.unidata.ucar.edu/pub/netcdf/netcdf-4.5.0.tar.gz
tar -xvf netcdf-4.5.0.tar.gz
```

2. After unpacking NetCDF, continue with the following steps for configuration and compilation:

```
cd netcdf-c-4.5.0
module unload slurm
module load hdf5/1.8.19_intel-2017_update-1
./configure --prefix=/share/apps/netcdf/4.5.0/intel-17.0.1 --build=x86_64-
↪redhat-linux 2>&1 | tee netcdf-conf.log
make 2>&1 | tee netcdf-make.log
sudo mkdir -p /share/apps/netcdf/4.5.0/intel-17.0.1
sudo chown -R mgomezzul.apolo /share/apps/netcdf/4.5.0/intel-17.0.1
make install 2>&1 | tee netcdf-make-install.log
sudo chown -R root.root /share/apps/netcdf/4.5.0/intel-17.0.1
```

Module

```
#%Module1.0#####################################################################
↪##
##
## module load netcdf/4.5.0_intel-17.0.1
##
## /share/apps/modules/netcdf/4.5.0_intel-17.0.1
## Written by Mateo Gómez-Zuluaga
##
```

(continues on next page)

(continued from previous page)

```
proc ModulesHelp {} {
    global version modroot
    puts stderr "Sets the environment for using netcdf 4.5.0\
        \nin the shared directory \
        \n/share/apps/netcdf/4.5.0/intel-17.0.1 builded with\
        \nIntel Paralle Studio XE 2017 Update 1 Cluster Edition,\
        \nzlib-1.2.11, szip-2.1, hdf5-1.8.19 and libtools-2.4.6."
}

module-whatis "(Name________) netcdf"
module-whatis "(Version_____) 4.5.0"
module-whatis "(Compilers___) intel-17.0.1"
module-whatis "(System______) x86_64-redhat-linux"
module-whatis "(Libraries___) libtools-2.4.6,szip-2.1,zlib-1.2.11"

# for Tcl script use only
set         topdir        /share/apps/netcdf/4.5.0/intel-17.0.1
set         version       4.5.0
set         sys           x86_64-redhat-linux

conflict netcdf
module load intel/2017_update-1
module load szip/2.1_intel-2017_update-1
module load zlib/1.2.11_intel-2017_update-1
module load libtool/2.4.6_intel-17.0.1
module load hdf5/1.8.19_intel-2017_update-1

prepend-path    PATH                    $topdir/bin

prepend-path    LD_LIBRARY_PATH         $topdir/lib
prepend-path    LIBRARY_PATH            $topdir/lib
prepend-path    LD_RUN_PATH             $topdir/lib

prepend-path    C_INCLUDE_PATH          $topdir/include
prepend-path    CXX_INCLUDE_PATH        $topdir/include
prepend-path    CPLUS_INCLUDE_PATH      $topdir/include

prepend-path    PKG_CONFIG_PATH         $topdir/lib/pkgconfig

prepend-path    MANPATH                 $topdir/share/man
```

Use

TO-DO

Resources

- https://www.unidata.ucar.edu/downloads/netcdf/index.jsp
- http://www.unidata.ucar.edu/software/netcdf/docs/getting_and_building_netcdf.html

Author

- Mateo Gómez Zuluaga

NetCDF 4.7.4

Table of Contents

Basic information

- **Official Website:** https://www.unidata.ucar.edu/downloads/netcdf/index.jsp
- **License:** MIT open source
- **Installed on:** Apolo II
- **Installation date:** June 2020

Tested on (Requirements)

- **OS base:** CentOS (x86_64) $\geq$ 6.6 (Rocks 6.2)
- **Dependencies:**
  - Intel Parallel Studio XE Cluster Edition >= 17.0.1
  - hdf5 >= 1.8.19
  - zlib >= 1.2.11
  - szip >= 2.1

Installation

1. Download the desired version of the software (Source code - tar.gz)

```
cd $HOME/apps/netCDF/intel
git clone https://github.com/Unidata/netcdf-c.git
```

2. After unpacking NetCDF, continue with the following steps for configuration and compilation:

```
$ cd netcdf-c
$ module load hdf5/1.12_intel_19.0.4
$ export LDFLAGS="-L/share/apps/hdf5/1.12/intel-19.0.4/lib"
$ export CPPFLAGS="-I/share/apps/hdf5/1.12/intel-19.0.4/include"
$ ./configure --prefix=/share/apps/netcdf/4.7.4/intel-19.0.4 --build=x86_
↪64-redhat-linux --enable-largefile
$ make 2>&1 | tee netcdf-make.log
$ sudo mkdir -p /share/apps/netcdf/4.5.0/intel-17.0.1
$ sudo make install 2>&1 | tee netcdf-make-install.log
```

Module

```
##%Module1.0#####################################################################
↪##
##
## module load netcdf/4.7.4_intel-19.0.4
##
## /share/apps/modules/netcdf/4.7.4_intel-19.0.4
## Written by Tomas Navarro & Santiago Alzate
##

proc ModulesHelp {} {
     global version modroot
     puts stderr "Sets the environment for using netcdf 4.7.4\
                  \nin the shared directory \
                  \n/share/apps/netcdf/4.7.4/intel-19.0.4 builded with\
                  \nIntel Paralle Studio XE 2019 Update 4 Cluster Edition,\
                  \nzlib-1.2.11, szip-2.1.1, hdf5-1.12"
}

module-whatis "(Name________) netcdf"
module-whatis "(Version_____) 4.7.4"
module-whatis "(Compilers___) intel-19.0.4"
module-whatis "(System______) x86_64-redhat-linux"
module-whatis "(Libraries___) szip-2.1,zlib-1.2.11"

# for Tcl script use only
set         topdir        /share/apps/netcdf/4.7.4/intel-19.0.4
set         version       4.7.4
set         sys           x86_64-redhat-linux

conflict netcdf
module load intel/19.0.4
module load szip/2.1.1_intel_19.0.4
module load zlib/1.2.11_intel_19.0.4
module load hdf5/1.12_intel-19.0.4

#setenv                 NETCDF                  $topdir

prepend-path    PATH                    $topdir/bin

prepend-path    LD_LIBRARY_PATH         $topdir/lib
prepend-path    LIBRARY_PATH            $topdir/lib
prepend-path    LD_RUN_PATH             $topdir/lib
```

(continues on next page)

(continued from previous page)

```
prepend-path    C_INCLUDE_PATH          $topdir/include
prepend-path    CXX_INCLUDE_PATH        $topdir/include
prepend-path    CPLUS_INCLUDE_PATH      $topdir/include

prepend-path    PKG_CONFIG_PATH         $topdir/lib/pkgconfig

prepend-path    MANPATH                 $topdir/share/man
```

Mode Of Use

```
module load netcdf/4.7.4_intel-19.0.4
```

Resources

- https://www.unidata.ucar.edu/downloads/netcdf/index.jsp

Author

- Tomas David Navarro
- Santiago Alzate Cardona

Fortran Version

NetCDF 4.4.4

Table of Contents

Basic information

- **Official Website:** https://www.unidata.ucar.edu/downloads/netcdf/index.jsp
- **License:** MIT open source

- **Installed on:** Apolo II and Cronos
- **Installation date:** 14/11/2017

Tested on (Requirements)

- **OS base:** CentOS (x86_64) $\geq$ 6.6 (Rocks 6.2)
- **Dependencies:**
  - Intel Parallel Studio XE Cluster Edition >= 17.0.1
  - netCDF-C >= 4.5.0
  - hdf5 >= 1.8.19
  - zlib >= 1.2.11
  - szip >= 2.1

Installation

1. Download the desired version of the software (Source code - tar.gz)

```
cd /home/mgomezzul/apps/netCDF-Fortran/intel/
wget ftp://ftp.unidata.ucar.edu/pub/netcdf/netcdf-fortran-4.4.4.tar.gz
tar -xvf netcdf-fortran-4.4.4.tar.gz
```

2. After unpacking NetCDF, continue with the following steps for configuration and compilation:

```
cd netcdf-fortran-4.4.4
module unload slurm
module load netcdf/4.5.0_intel-17.0.1
./configure --prefix=/share/apps/netcdf-fortran/4.4.4/intel-2017_update-1␣
↪--build=x86_64-redhat-linux 2>&1 | tee netcdf-fortran-conf.log
make 2>&1 | tee netcdf-fortran-make.log
sudo mkdir -p /share/apps/netcdf-fortran/4.4.4/intel-2017_update-1
sudo chown -R mgomezzul.apolo /share/apps/netcdf-fortran/4.4.4/intel-2017_
↪update-1
make install 2>&1 | tee netcdf-make-install.log
sudo chown -R root.root /share/apps/netcdf-fortran/4.4.4/intel-2017_
↪update-1
```

Module

```
#%Module1.0#####################################################################
↪###
##
## module netcdf-fortran/4.4.4_intel-2017_update-1
##
## /share/apps/netcdf-fortran/4.4.4/intel-2017_update-1     Written by Mateo␣
↪Gomez-Zuluaga
##

proc ModulesHelp { } {
```

(continues on next page)

(continued from previous page)

```
    puts stderr "\tnetcdf-fortran/4.4.4 - sets the Environment for ZLIB in \
    \n\tthe share directory /share/apps/netcdf-fortran/4.4.4/intel-2017_
↪update-1\n"
}

module-whatis "\n\n\tSets the environment for using NetCDF-Fortran 4.4.4 \
           \n\tbuilded with Intel Parallel Studio XE 2017\n"

# for Tcl script use only
set       topdir     /share/apps/netcdf-fortran/4.4.4/intel-2017_update-1
set       version    4.4.4
set       sys        x86_64-redhat-linux

module load netcdf/4.5.0_intel-17.0.1

prepend-path PATH               $topdir/bin

prepend-path LD_LIBRARY_PATH    $topdir/lib
prepend-path LIBRARY_PATH       $topdir/lib
prepend-path LD_RUN_PATH        $topdir/lib

prepend-path C_INCLUDE_PATH     $topdir/include
prepend-path CXX_INCLUDE_PATH   $topdir/include
prepend-path CPLUS_INCLUDE_PATH $topdir/include
prepend-path INCLUDE            $topdir/include

prepend-path PKG_CONFIG_PATH    $topdir/lib/pkgconfig

prepend-path MANPATH            $topdir/share/man
```

Use

TO-DO

Resources

- https://www.unidata.ucar.edu/downloads/netcdf/index.jsp
- http://www.unidata.ucar.edu/software/netcdf/docs/getting_and_building_netcdf.html

Author

- Mateo Gómez Zuluaga

NetCDF 4.4.2

Table of Contents

Basic information

- **Official Website:** https://www.unidata.ucar.edu/software/netcdf/
- **License:** Copyright University Corporation for Atmospheric Research/Unidata
- **Installed on:** Apolo II and Cronos
- **Installation date:** 13/02/2018

Tested on (Requirements)

- **OS base:** CentOS (x86_64) $\geq$ 6.6 (Rocks 6.2)
- **Dependencies:**
  - mpich2 v3.2.1
  - HDF5 v1.8.15-patch-1
  - PNetCDF/1.6.1
  - NetCDF-C/4.3.3

Installation

Note: netCDF-C must be installed before the nerCDF-Fortran installation proceedure.

1. Configure the following environment variables that specify the compilers to use:

```
export CC=mpicc
export CXX=mpicxx
export FC=mpif90
export F77=mpif90
export F90=mpif90
NETCDF=
MPICH2=
export CPPFLAGS="-I$HDF5/include -I$NETCDF/include -I/export/apps/mpich2/
↪3.2.1/gcc-5.5.0/include"
export CFLAGS="-I$HDF5/include -I$NETCDF/include -I/export/apps/mpich2/3.
↪2.1/gcc-5.5.0/include"
export CXXFLAGS="-I$HDF5/include -I$NETCDF/include -I/export/apps/mpich2/
↪3.2.1/gcc-5.5.0/include"
```

(continues on next page)

(continued from previous page)

```
export FCFLAGS="-I$HDF5/include -I$NETCDF/include -I/export/apps/mpich2/3.
↪2.1/gcc-5.5.0/include"
export FFLAGS="-I$HDF5/include -I$NETCDF/include -I/export/apps/mpich2/3.
↪2.1/gcc-5.5.0/include"
export LDFLAGS="-L$HDF5/lib -L$NETCDF/lib -L/export/apps/mpich2/3.2.1/gcc-
↪5.5.0/lib"
HDF5=
NCDIR=
```

2. Download the desired version of the software (Source code - tar.gz)

```
cd /home/mgomezzul/apps/netcdf/src/
wget ftp://ftp.unidata.ucar.edu/pub/netcdf/netcdf-fortran-4.4.2.tar.gz
tar -xvf netcdf-fortran-4.4.2.tar.gz
```

3. After unpacking NetCDF, continue with the following steps for configuration and compilation:

```
cd netcdf-4.4.2

./configure --prefix=/share/apps/netcdf/4.3.3/gcc-5.5.0_mpich2-3.2.1 --
↪build=x86_64-redhat-linux --enable-large-file-tests --enable-parallel-
↪tests --enable-largefile

make -j 10 2>&1 | tee netcdff-make.log
make check 2>&1 | tee netcdff-make-check.log
sudo mkdir -p /share/apps/netcdf/4.3.3/gcc-5.5.0_mpich2-3.2.1
sudo chown -R mgomezzul.apolo /share/apps/netcdf/4.3.3/gcc-5.5.0_mpich2-3.
↪2.1
make install 2>&1 | tee netcdff-make-install.log
sudo chown -R root.root /share/apps/netcdf/4.3.3/gcc-5.5.0_mpich2-3.2.1
```

4. Generate the module with the following environment variable:

```
setenv          PNET                    $topdir
sudo moduleGenerator
```

Module

```
#%Module1.0#####################################################################
↪##
##
## module load netcdf/4.4.2_gcc-5.5.0_mpich2-3.2.1
##
## /share/apps/modules/netcdf/4.3.3_gcc-5.5.0_mpich2-3.2.1
## Written by Mateo Gómez-Zuluaga
##

proc ModulesHelp {} {
    global version modroot
    puts stderr "Sets the environment for using netcdf-4.3.3\
                \nin the shared directory /share/apps/netcdf/4.3.3/gcc-5.5.0_
↪mpich2-3.2.1\
                \nbuilded with gcc-5.5.0, mpich2-3.2.1, hdf5-1.8.15-patch1,␣
↪pnetcdf/1.6.1."
```

(continues on next page)

(continued from previous page)

```
}

module-whatis "(Name_________) netcdf"
module-whatis "(Version______) 4.4.2"
module-whatis "(Compilers____) gcc-5.5.0, mpich2-3.2.1"
module-whatis "(System_______) x86_64-redhat-linux"
module-whatis "(Libraries____) zlib-1.2.11, szip-2.1.1, hdf5-1.8.15-patch1,␣
↪pnetcdf-1.6.1"

# for Tcl script use only
set         topdir        /share/apps/netcdf/4.3.3/gcc-5.5.0_mpich2-3.2.1
set         version       4.4.2
set         sys           x86_64-redhat-linux

conflict netcdf
module load mpich2/3.2.1_gcc-5.5.0
module load hdf5/1.8.15-patch1_gcc-5.5.0_mpich2-3.2.1
module load pnetcdf/1.6.1_gcc-5.5.0_mpich2-3.2.1

setenv          NETCDF                  $topdir

prepend-path    PATH                    $topdir/bin

prepend-path    LD_LIBRARY_PATH         $topdir/lib
prepend-path    LIBRARY_PATH            $topdir/lib
prepend-path    LD_RUN_PATH             $topdir/lib

prepend-path    C_INCLUDE_PATH          $topdir/include
prepend-path    CXX_INCLUDE_PATH        $topdir/include
prepend-path    CPLUS_INCLUDE_PATH      $topdir/include

prepend-path    PKG_CONFIG_PATH         $topdir/lib/pkgconfig

prepend-path    MANPATH                 $topdir/share/man
```

Use

```
module load netcdf/4.4.2_gcc-5.5.0_mpich2-3.2.1
```

Resources

- ftp://ftp.unidata.ucar.edu/pub/netcdf/
- https://www.unidata.ucar.edu/software/netcdf/docs/copyright.html

Author

- Andrés Felipe Zapata Palacio

NetCDF-Fortran 4.5.3

Table of Contents

Basic information

- **Official Website:** https://www.unidata.ucar.edu/downloads/netcdf/index.jsp
- **License:** MIT open source
- **Installed on:** Apolo II
- **Installation date:** June 2020

Tested on (Requirements)

- **OS base:** CentOS (x86_64) $\geq$ 6.6 (Rocks 6.2)
- **Dependencies:**
  - Intel Parallel Studio XE Cluster Edition >= 17.0.1
  - hdf5 >= 1.8.19
  - zlib >= 1.2.11
  - szip >= 2.1

Installation

1. Download the desired version of the software (Source code - tar.gz)

```
cd $HOME/apps/netCDF/intel
git clone https://github.com/Unidata/netcdf-fortran.git
```

2. After unpacking NetCDF, continue with the following steps for configuration and compilation:

```
$ cd netcdf-fortran
$ module load hdf5/1.12_intel_19.0.4
$ export LDFLAGS="-L/share/apps/hdf5/1.12/intel-19.0.4/lib -L/share/apps/
↪netcdf/4.7.4/intel-19.0.4/lib -L/share/apps/mpich2/3.3.2/intel-19.0.4/
↪lib"
```

(continues on next page)

(continued from previous page)

```
$ export CPPFLAGS="-I/share/apps/hdf5/1.12/intel-19.0.4/include -L/share/
↪apps/netcdf/4.7.4/intel-19.0.4/include -L/share/apps/mpich2/3.3.2/intel-
↪19.0.4/include"
$ ./configure --prefix=/share/apps/netcdf-fortran/4.5.3/intel-19.0.4 --
↪build=x86_64-redhat-linux
$ make 2>&1 | tee netcdf-make.log
$ sudo mkdir -p /share/apps/netcdf/4.5.0/intel-17.0.1
$ sudo make install 2>&1 | tee netcdf-make-install.log
```

Module

```
#%Module1.0#####################################################################
↪##################
##
## module netcdf-fortran/4.5.3_intel-19.0.4
##
## /share/apps/netcdf-fortran/4.5.3/intel-19.0.4     Written by Tomas␣
↪Navarro & Santiago Alzate
##

proc ModulesHelp { } {
    puts stderr "\tnetcdf-fortran/4.5.3 - sets the Environment for Netcdf-
↪fortran in \
        \n\tthe share directory /share/apps/netcdf-fortran/4.5.3/intel-19.0.
↪4\n"
}

module-whatis "\n\n\tSets the environment for using NetCDF-Fortran 4.4.4 \
               \n\tbuilded with Intel Parallel Studio XE 2017\n"

# for Tcl script use only
set       topdir     /share/apps/netcdf-fortran/4.5.3/intel-19.0.4
set       version    4.5.3
set       sys        x86_64-redhat-linux


setenv          NETCDF                  $topdir

module load netcdf/4.7.4_intel-19.0.4

prepend-path PATH               $topdir/bin
prepend-path LD_LIBRARY_PATH    $topdir/lib
prepend-path LIBRARY_PATH       $topdir/lib
prepend-path LD_RUN_PATH        $topdir/lib

prepend-path C_INCLUDE_PATH     $topdir/include
prepend-path CXX_INCLUDE_PATH   $topdir/include
prepend-path CPLUS_INCLUDE_PATH $topdir/include
prepend-path INCLUDE            $topdir/include

prepend-path PKG_CONFIG_PATH    $topdir/lib/pkgconfig

prepend-path MANPATH            $topdir/share/man
```

Mode Of Use

```
netcdf-fortran/4.5.3_intel-19.0.4
```

Resources

- https://www.unidata.ucar.edu/downloads/netcdf/index.jsp

Author

- Tomas David Navarro
- Santiago Alzate Cardona

NetCDF-Fortran 4.5.3 disable netcdf 4

Table of Contents

Basic information

- **Official Website:** https://www.unidata.ucar.edu/downloads/netcdf/index.jsp
- **License:** MIT open source
- **Installed on:** Apolo II
- **Installation date:** 03/03/2022

Tested on (Requirements)

- **OS base:** Rocky Linux 8.5 (x86_64)
- **Dependencies:**
  - gcc >= 9.3.0
  - curl >= 7.77.0
  - zlib >= 1.2.11

- netcdf-c >= 4.8.0

Installation

1. Download the desired version of the software (Source code - tar.gz)

```
cd /home/blopezp
wget https://downloads.unidata.ucar.edu/netcdf-fortran/4.5.3/netcdf-
↪fortran-4.5.3.tar.gz
```

2. After unpacking NetCDF, continue with the following steps for configuration and compilation only WRF-CMAQ:

```
$ cd netcdf-fortran
$ module load gcc/9.3.0 netcdf/4.8.0_gcc-9.3.0_disable-netcdf-4 mpich/3.4.
↪2_gcc-9.3.0
$ CPPFLAGS="-I/share/apps/netcdf/4.8.0/gcc-9.3.0/include" LDFLAGS="-L/
↪share/apps/netcdf/4.8.0/gcc-9.3.0/lib" ./configure --prefix=/share/apps/
↪netcdf-fortran/4.5.3_disable-netcdf-4/gcc-9.3.0
$ make 2>&1 | tee netcdf-make.log
$ sudo mkdir -p /share/apps/netcdf-fortran/4.5.3_disable-netcdf-4
$ sudo make install 2>&1 | tee netcdf-make-install.log
```

Module

```
##%Module1.0#####################################################################
↪##
##
## module load netcdf-fortran/4.5.3_gcc-9.3.0_disable-netcdf-4
##
## /share/apps/netcdf-fortran/4.5.3_disable-netcdf-4/gcc-9.3.0
## Written by Bryan Lopez Parra
##

proc ModulesHelp {} {
     global version modroot
     puts stderr "Sets the environment for using netcdf 4.5.3\
                  \nin the shared directory \
                  \n/share/apps/netcdf-fortran/4.5.3_disable-netcdf-4/gcc-9.
↪3.0 builded with\
                  \nIntel gcc 9.3.0\"
}

module-whatis "(Name________) netcdf-fortran"
module-whatis "(Version_____) 4.5.3"
module-whatis "(Compilers___) gcc-9.3.0"
module-whatis "(System______) x86_64-redhat-linux"
module-whatis "(Libraries___) zlib-1.2.11"

# for Tcl script use only
set         topdir        /share/apps/netcdf-fortran/4.5.3_disable-netcdf-4/
↪gcc-9.3.0
set         version       4.5.3
set         sys           x86_64-redhat-linux
```

(continues on next page)

(continued from previous page)

```
conflict netcdf-fortran

module load netcdf/4.8.0_gcc-9.3.0_disable-netcdf-4

prepend-path    PATH                    $topdir/bin

setenv          NETCDF                  $topdir
prepend-path    LD_LIBRARY_PATH         $topdir/lib
prepend-path    LIBRARY_PATH            $topdir/lib
prepend-path    LD_RUN_PATH             $topdir/lib

prepend-path    C_INCLUDE_PATH          $topdir/include
prepend-path    CXX_INCLUDE_PATH        $topdir/include
prepend-path    CPLUS_INCLUDE_PATH      $topdir/include

prepend-path    PKG_CONFIG_PATH         $topdir/lib/pkgconfig

prepend-path    MANPATH                 $topdir/share/man
```

Mode Of Use

```
module load netcdf-fortran/4.5.3_gcc-9.3.0_disable-netcdf-4
```

Resources

- https://www.unidata.ucar.edu/downloads/netcdf/index.jsp

Author

- Bryan López Parra <blopezp@eafit.edu.co>

NetCDF 4.8.0 disable netcdf 4

Table of Contents

Basic information

- **Official Website:** https://www.unidata.ucar.edu/software/netcdf/
- **License:** Copyright University Corporation for Atmospheric Research/Unidata.
- **Installed on:** Apolo II
- **Installation date:** 03/03/2022

Tested on (Requirements)

- **OS base:** Rocky Linux 8.5 (x86_64)
- **Dependencies:**
  - gcc v9.3.0
  - curl v.7.77.0
  - Mpich v.3.4.2
  - Zlib v.1.2.11

Installation

1. Load required modules.

```
module load gcc/9.3.0
module load curl/7.77.0_gcc-9.3.0
module load mpich/3.4.2_gcc-9.3.0
```

2. Configure the following environment variables that specify the compilers to use:

```
export CC=mpicc
export CXX=mpicxx
export FC=mpif90
export F77=mpif90
export F90=mpif90
```

3. Download the desired version of the software (Source code - tar.gz)

```
cd /home/blopezp
wget https://ftp.unidata.ucar.edu/pub/netcdf/netcdf-c-4.8.0.tar.gz
tar -xvf netcdf-4.3.3.tar.gz
```

4. After unpacking NetCDF, continue with the following steps for configuration and compilation only WRF-CMAQ:

```
cd netcdf-c-4.8.0.tar.gz

CPPFLAGS="-I/share/apps/curl/7.77.0/gcc-9.3.0/include" LDFLAGS="-L/share/
↪apps/curl/7.77.0/gcc-9.3.0/lib" ./configure --prefix=/share/apps/netcdf/
↪4.8.0/gcc-9.3.0 --disable-netcdf-4

make -j 10 2>&1 | tee netcdf-make.log
make check 2>&1 | tee netcdf-make-check.log
```

(continues on next page)

(continued from previous page)

```
sudo mkdir -p /share/apps/netcdf/4.8.0
sudo make -j 10 install 2>&1 | tee netcdf-make-install.log
```

Module

```
##%Module1.0#####################################################################
↪##
##
## module load netcdf/4.8.0_gcc-9.3.0
##
## /share/apps/modules/netcdf/4.8.1_Intel_oneAPI_2022-update-1
## Written by Bryan Lopez Parra
##

proc ModulesHelp {} {
     global version modroot
     puts stderr "Sets the environment for using netcdf 4.8.1\
                  \nin the shared directory \
                  \n/share/apps/netcdf/4.8.0/gcc-9.3.0 builded with\
                  \nGCC 9.3.0,\
                  \nzlib-1.2.11"
}

module-whatis "(Name________) netcdf"
module-whatis "(Version_____) 4.8.0"
module-whatis "(Compilers___) gcc-9.3.0"
module-whatis "(System______) x86_64-redhat-linux"
module-whatis "(Libraries___) zlib-1.2.11"

# for Tcl script use only
set         topdir        /share/apps/netcdf/4.8.0/gcc-9.3.0
set         version       4.8.0
set         sys           x86_64-redhat-linux

conflict netcdf

module load curl/7.77.0_gcc-9.3.0



#setenv                 NETCDF                  $topdir

prepend-path    PATH                    $topdir/bin

setenv          NCDIR                   $topdir

prepend-path    LD_LIBRARY_PATH         $topdir/lib
prepend-path    LIBRARY_PATH            $topdir/lib
prepend-path    LD_RUN_PATH             $topdir/lib

prepend-path    C_INCLUDE_PATH          $topdir/include
prepend-path    CXX_INCLUDE_PATH        $topdir/include
prepend-path    CPLUS_INCLUDE_PATH      $topdir/include
```

(continues on next page)

(continued from previous page)

```
prepend-path    PKG_CONFIG_PATH         $topdir/lib/pkgconfig

prepend-path    MANPATH                 $topdir/share/man
```

Use

```
module load zlib/1.2.11_gcc-9.3.0
```

Resources

- https://ftp.unidata.ucar.edu/pub/netcdf/
- https://www.unidata.ucar.edu/software/netcdf/docs/copyright.html

Author

- Bryan López Parra <blopezp@eafit.edu.co>

3.8.3 GMP

Description

GMP is a free library for arbitrary precision arithmetic, operating on signed integers, rational numbers, and floating-point numbers. There is no practical limit to the precision except the ones implied by the available memory in the machine GMP runs on. GMP has a rich set of functions, and the functions have a regular interface.

The main target applications for GMP are cryptography applications and research, Internet security applications, algebra systems, computational algebra research, etc.

GMP is carefully designed to be as fast as possible, both for small operands and for huge operands. The speed is achieved by using fullwords as the basic arithmetic type, by using fast algorithms, with highly optimised assembly code for the most common inner loops for a lot of CPUs, and by a general emphasis on speed.[1]

GMP 6.1.2

Table of Contents

[1] https://gmplib.org/

Basic information

- **Official Website:** https://gmplib.org/
- **License:** Free Software Foundation's GNU General Public License
- **Installed on:** Apolo II and Cronos
- **Installation date:** 01/02/2018

Installation

1. Download the desired version of the software (Source code - tar.gz)

```
cd /home/mgomezzul/apps/gmp/src/
wget https://gmplib.org/download/gmp/gmp-6.1.2.tar.bz2
tar -xvf gmp-6.1.2.tar.bz2
```

2. After unpacking GMP, continue with the following steps for configuration and compilation:

```
cd gmp-6.1.2
./configure --prefix=/share/apps/gmp/6.1.2/gcc-4.4.7-18 --build=x86_64-
↪redhat-linux-gnu --enable-cxx --enable-assert --with-gnu-ld
make -j 10 2>&1 | tee gmp-make.log
sudo mkdir -p /share/apps/gmp/6.1.2
sudo chown -R mgomezzul.apolo /share/apps/gmp/6.1.2
make install 2>&1 | tee gmp-make-install.log
sudo chown -R root.root /share/apps/gmp/6.1.2
```

Module

```
##%Module1.0#####################################################################
↪##
##
## module load gmp/6.1.2_gcc-4.4.7-18
##
## /share/apps/modules/gmp/6.1.2_gcc-4.4.7-18
## Written by Mateo Gomez Zuluaga
##
proc ModulesHelp {} {
    global version modroot
    puts stderr "\n\n\tSets the environment for using gmp (GNU Multiple \
                \n\tPrecision Arithmetic Library) builded gcc 4.4.7-18␣
↪version\n"
}
module-whatis "(Name________) gmp"
module-whatis "(Version_____) 6.1.2"
module-whatis "(Compilers___) gcc-4.4.7-18"
module-whatis "(System______) x86_64-redhat-linux"
module-whatis "(Libraries___) "
# for Tcl script use only
set         topdir        /share/apps/gmp/6.1.2/gcc-4.4.7-18
set         version       6.1.2
set         sys           x86_64-redhat-linux
conflict gmp
```

(continues on next page)

(continued from previous page)

```
prepend-path    LD_LIBRARY_PATH         $topdir/lib
prepend-path    LIBRARY_PATH            $topdir/lib
prepend-path    LD_RUN_PATH             $topdir/lib
prepend-path    GMP_LIBRARIES           $topdir/lib
prepend-path    GMPXX_LIBRARIES         $topdir/lib
prepend-path    C_INCLUDE_PATH          $topdir/include
prepend-path    CXX_INCLUDE_PATH        $topdir/include
prepend-path    CPLUS_INCLUDE_PATH      $topdir/include
prepend-path    GMP_INCLUDE_DIR         $topdir/include
prepend-path    GMPXX_INCLUDE_DIR       $topdir/include
prepend-path    INFOPATH                $topdir/share/info
```

Use

```
module load /share/apps/modules/gmp/6.1.2_gcc-4.4.7-18
```

Resources

- https://gmplib.org/#DOWNLOAD

Author

- Tomas Navarro (Translated the document)

GMP 6.2.1

Table of Contents

Basic information

- **Official Website:** https://gmplib.org/
- **License:** Free Software Foundation's GNU General Public License
- **Installed on:** Apolo II
- **Installation date:** 03/03/2022

Installation

1. Download the desired version of the software (Source code - tar.gz)

```
cd /home/blopezp
wget https://gmplib.org/download/gmp-6.2.1/gmp-6.2.1.tar.xz
tar -xvf gmp-6.2.1.tar.xz
```

2. After unpacking GMP, continue with the following steps for configuration and compilation:

```
cd gmp-6.2.1
./configure --prefix=/share/apps/gmp/6.2.1/gcc-8.5.0  --build=x86_64-
↪redhat-linux-gnu --enable-cxx --enable-assert --with-gnu-ld
make -j 10 2>&1 | tee gmp-make.log
sudo mkdir -p /share/apps/gmp/6.2.1
sudo make install 2>&1 | tee gmp-make-install.log
```

Module

```
#%Module1.0#####################################################################
↪##
##
## module load gmp/6.2.1_gcc-8.5.0
##
## /share/apps/modules/gmp/6.1.2_gcc-8.5.0
## Written by Bryan Lopez Parra
##
proc ModulesHelp {} {
    global version modroot
    puts stderr "\n\n\tSets the environment for using gmp (GNU Multiple \
                \n\tPrecision Arithmetic Library) builded gcc 8.5.0 version\n
↪"
}
module-whatis "(Name________) gmp"
module-whatis "(Version_____) 6.2.1"
module-whatis "(Compilers___) gcc-8.5.0"
module-whatis "(System______) x86_64-redhat-linux"
module-whatis "(Libraries___) "
# for Tcl script use only
set         topdir        /share/apps/gmp/6.2.1/gcc-8.5.0
set         version       6.2.1
set         sys           x86_64-redhat-linux
conflict gmp

prepend-path    LD_LIBRARY_PATH         $topdir/lib
prepend-path    LIBRARY_PATH            $topdir/lib
prepend-path    LD_RUN_PATH             $topdir/lib
prepend-path    GMP_LIBRARIES           $topdir/lib
prepend-path    GMPXX_LIBRARIES         $topdir/lib
prepend-path    C_INCLUDE_PATH          $topdir/include
prepend-path    CXX_INCLUDE_PATH        $topdir/include
prepend-path    CPLUS_INCLUDE_PATH      $topdir/include
prepend-path    GMP_INCLUDE_DIR         $topdir/include
prepend-path    GMPXX_INCLUDE_DIR       $topdir/include
prepend-path    INFOPATH                $topdir/share/info
```

Use

```
module load gmp/6.2.1_gcc-8.5.0
```

Resources

- https://gmplib.org/#DOWNLOAD

Authors

- Bryan López Parra <blopezp@eafit.edu.co>

Resources

3.8.4 Eigen

Eigen[1] is a C++ template library for linear algebra: matrices, vectors, numerical solvers, and related algorithms.

Eigen 3.3.7

Table of Contents

Basic information

- **Official Website:** http://eigen.tuxfamily.org/index.php?title=Main_Page#License
- **Installed on:** *Apolo II*

Tested on (Requirements)

- **OS base:** CentOS (x86_64) $\geq$ 6.6 (Rocks 6.2)

[1] http://eigen.tuxfamily.org/index.php?title=Main_Page#Credits

Dependencies

Eigen[1] doesn't have any dependencies other than the C++ standard library. Eigen uses the CMake build system, but only to build the documentation and unit-tests, and to automate installation. If you just want to use Eigen, you can use the header files right away. There is no binary library to link to, and no configured header file. Eigen is a pure template library defined in the headers.

Installation

1. Download the latest version of Eigen and unzip the file

```
$ wget https://gitlab.com/libeigen/eigen/-/archive/3.3.7/eigen-3.3.7.tar.
↪gz
$ tar -xzvf eigen-3.3.7.tar.gz
```

2. Inside the folder, on the top create a `build` directory where the installation binaries will be put by cmake.

```
$ cd eigen-3.3.7
$ mkdir build
$ cd build
```

3. Load the necessary modules for the building.

```
$ module load cmake/3.7.1
$ module load intel/19.0.4
$ module load boost/1.62.0_gcc-5.4.0_openmpi-1.8.8-x86_64
```

4. Execute the cmake command with the desired directives.

```
$  cmake ..  -DCMAKE_C_COMPILER=icc -DCMAKE_CXX_COMPILER=icpc -DCMAKE_
↪INSTALL_PREFIX=/share/apps/eigen/3.3.7/intel-19.0.4
```

5. Execute the make commands sequence.

```
$ make -j <N>
$ make check
$ make -j <N> install
```

Warning: Some tests may fail, but the installation can continue depending on the number of failed tests.

Module

```
#%Module1.0#####################################################################
↪###
##
## modulefile /share/apps/eigen/3.3.7/intel-19.0.4/
##

proc ModulesHelp { } {
    global version modroot
```

(continues on next page)

[1] http://eigen.tuxfamily.org/index.php?title=Main_Page#Credits

(continued from previous page)

```
        puts stderr "\t Eigen 3.3.7"
}

module-whatis "\n\n\tSets the environment for using Eigen 3.3.7 \n"


set     topdir          /share/apps/eigen/3.3.7/intel-19.0.4/
set     version         3.3.7
set     sys             x86_64-redhat-linux

module load intel/19.0.4
module load boost/1.62.0_gcc-5.4.0_openmpi-1.8.8-x86_64

setenv          EIGEN_HOME              $topdir

prepend-path    C_INCLUDE_PATH          $topdir/include/eigen3
prepend-path    CXX_INCLUDE_PATH        $topdir/include/eigen3
prepend-path    CPLUS_INCLUDE_PATH      $topdir/include/eigen3

prepend-path    MANPATH                 $topdir/share
```

References

Authors

- Santiago Hidalgo Ocampo <shidalgoo1@eafit.edu.co>

3.8.5 Pybind

Pybind[1] pybind11 is a lightweight header-only library that exposes C++ types in Python and vice versa, mainly to create Python bindings of existing C++ code. Its goals and syntax are similar to the excellent Boost.Python library by David Abrahams: to minimize boilerplate code in traditional extension modules by inferring type information using compile-time introspection.

Pybind11

Table of Contents

[1] https://github.com/pybind/pybind11

Basic information

- **Official Website:** https://github.com/pybind/pybind11
- **Installed on:** *Apolo II*

Tested on (Requirements)

- **OS base:** CentOS (x86_64) $\geq$ 6.6 (Rocks 6.2)

Installation

1. Download the latest version of pybind11

```
$ git clone https://github.com/pybind/pybind11.git
```

2. Inside the folder, on the top create a `build` directory where the installation binaries will be put by cmake.

```
$ cd pybind11
$ mkdir build
$ cd build
```

3. Load the necessary modules for the building.

```
$ module load python/3.6.5_miniconda-4.5.1
$ module load cmake/3.7.1
$ module load gcc/5.4.0
$ module load boost/1.62.0_gcc-5.4.0_openmpi-1.8.8-x86_64
```

4. Execute the cmake command with the desired directives.

```
$ cmake .. -DCMAKE_INSTALL_PREFIX=/share/apps/pybind11/gcc-5.4.0/ -DCMAKE_
↪C_COMPILER=gcc -DCMAKE_CXX_COMPILER=g++
```

5. Execute the make commands sequence.

```
$ make -j <N>
$ make check
$ make -j <N> install
```

Module

```
##%Module1.0#####################################################################
↪###
##
## modulefile /share/apps/modules/pybind11/11_gcc-5.4.0
## Written by Juan Diego Ocampo and Santiago Hidalgo Ocampo
##

proc ModulesHelp { } {
    global version modroot
        puts stderr "\t Pybind11"
```

(continues on next page)

(continued from previous page)

```
}

module-whatis "\n\n\tSets the environment for using Pybind11 \n"

module-whatis "(Name________) Pybind11"
module-whatis "(Version_____) 11"
module-whatis "(Compilers___) gcc-5.4.0"
module-whatis "(System______) x86_64-redhat-linux"

set     topdir          /share/apps/pybind11/gcc-5.4.0
set     version         11
set     sys             x86_64-redhat-linux

module load gcc/5.4.0
module load boost/1.62.0_gcc-5.4.0_openmpi-1.8.8-x86_64

setenv          PYBIND11_HOME           $topdir

prepend-path    C_INCLUDE_PATH          $topdir/include
prepend-path    CXX_INCLUDE_PATH        $topdir/include
prepend-path    CPLUS_INCLUDE_PATH      $topdir/include
```

Troubleshooting

See also:

If you have this problem : CMake Error at tests/CMakeLists.txt:217 (message): Running the tests requires pytest. You must run the following command with administrator privileges taking into account the version of python you are using:

```
$ pip install pytest
```

Authors

- Santiago Hidalgo Ocampo <shidalgoo1@eafit.edu.co>

3.8.6 FFTW3

FFTW is a C subroutine library for computing the discrete Fourier transform (DFT) in one or more dimensions, of arbitrary input size, and of both real and complex data (as well as of even/odd data, i.e. the discrete cosine/sine transforms or DCT/DST). We believe that FFTW, which is free software, should become the FFT library of choice for most applications. The latest official release of FFTW is version 3.3.6, available from our download page. Version 3.3 introduced support for the AVX x86 extensions, a distributed-memory implementation on top of MPI, and a Fortran 2003 API. Version 3.3.1 introduced support for the ARM Neon extensions. See the release notes for more information.

The FFTW package was developed at MIT by Matteo Frigo and Steven G. Johnson.

Our benchmarks, performed on on a variety of platforms, show that FFTW's performance is typically superior to that of other publicly available FFT software, and is even competitive with vendor-tuned codes. In contrast to vendor-tuned codes, however, FFTW's performance is portable: the same program will perform well on most architectures without modification. Hence the name, "FFTW," which stands for the somewhat whimsical title of "Fastest Fourier Transform in the West."s.

FFTW 3.3.6

Table of Contents

Basic information

- **Official Website:** http://www.fftw.org/
- **License:** GNU GENERAL PUBLIC LICENSE Version 2
- **Installed on:** Apolo II and Cronos
- **Installation date:** 21/03/2017

Tested on (Requirements)

- **OS base:** CentOS (x86_64) $\geq$ 6.6 (Rocks 6.2)
- **Dependencies:**
  - GCC >= 5.4.0
  - OpenMPI >= 1.8.8

Installation

1. Download the desired version of the software (Source code - tar.gz)

```
cd /home/mgomezzul/apps/fftw/3/gcc/5.4.0
wget ftp://ftp.fftw.org/pub/fftw/fftw-3.3.6-pl2.tar.gz
tar -zxvf fftw-3.3.6-pl2.tar.gz
```

2. After unpacking fftw3, continue with the following steps for configuration and compilation:

```
cd fftw-3.3.6-pl2
module load openmpi/1.8.8_gcc-5.4.0
module unload slurm
./configure --prefix=/share/apps/fftw/3.3.6/gcc-5.4.0_openmpi-1.8.8 --
↪build=x86_64-redhat-linux --enable-shared --enable-static --enable-
↪threads --enable-openmp --enable-mpi --enable-sse2 --enable-avx --
↪enable-avx2 --enable-avx512 --enable-avx-128-fma --with-gnu-ld 2>&1 |
↪tee fftw-conf.log
```

(continues on next page)

(continued from previous page)

```
echo Ignore warning of OpenGL not found
make 2>&1 | tee fftw-make.log
sudo mkdir -p /share/apps/fftw/3.3.6/gcc-5.4.0_openmpi-1.8.8
sudo chown -R mgomezzul.apolo /share/apps/fftw/3.3.6/gcc-5.4.0_openmpi-1.
↪8.8
make install 2>&1 | tee fftw-make-install.log
sudo chown -R root.root /share/apps/fftw/3.3.6/gcc-5.4.0_openmpi-1.8.8
```

Module

```
#%Module1.0#####################################################################
↪###
##
## module fftw/3.3.6_gcc-5.4.0_openmpi-1.8.8
##
## /share/apps/modules/fftw/3.3.6_gcc-5.4.0_openmpi-1.8.8 Written by Mateo␣
↪Gomez-Zuluaga
##

proc ModulesHelp { } {
    global version
    puts stderr "\tSets the environment for FFTW-$version in the next \
            \n\tthe shared directory /share/apps/fftw/3.3.6/gcc_5.4.0_
↪openmpi-1.8.8\n"
}

module-whatis "\n\n\tSets the environment for using FFTW3 \
            \n\t(library for computing the discrete Fourier transform)  \
            \n\tbuilded with gcc-5.4.0 and openmpi-1.8.8"

# for Tcl script use only
set     topdir          /share/apps/fftw/3.3.6/gcc-5.4.0_openmpi-1.8.8
set     version         3.3.6
set     sys             x86_64-redhat-linux

module load gcc/5.4.0
module load openmpi/1.8.8_gcc-5.4.0

prepend-path    PATH                    $topdir/bin

prepend-path    C_INCLUDE_PATH          $topdir/include
prepend-path    CXX_INCLUDE_PATH        $topdir/include
prepend-path    CPLUS_INCLUDE_PATH      $topdir/include

prepend-path    LD_LIBRARY_PATH         $topdir/lib
prepend-path    LIBRARY_PATH            $topdir/lib
prepend-path    LD_RUN_PATH             $topdir/lib

prepend-path    INFOPATH                $topdir/share/info
prepend-path    MANPATH                 $topdir/share/man

prepend-path    PKG_CONFIG_PATH         $topdir/lib/pkgconfig
```

Use

TO-DO

Resources

- http://www.fftw.org/fftw3_doc/

Author

- Mateo Gómez Zuluaga

FFTW 3.3.7

Table of Contents

Basic information

- **Official Website:** http://www.fftw.org/
- **License:** GNU GENERAL PUBLIC LICENSE Version 2
- **Installed on:** Apolo II and Cronos
- **Installation date:** 10/11/2017

Tested on (Requirements)

- **OS base:** CentOS (x86_64) $\geq$ 6.6 (Rocks 6.2)
- **Dependencies:**
  - Intel Compiler >= 2017_update-1
  - Gnu GCC >= 5.5.0

- Mpich2 >= 3.2.1

Installation

Apolo

1. Download the desired version of the software (Source code - tar.gz)

```
$ cd /home/mgomezzul/apps/fftw/3/intel
$ wget http://www.fftw.org/fftw-3.3.7.tar.gz
$ tar xf fftw-3.3.7.tar.gz
```

2. After unpacking fftw3, continue with the following steps for configuration and compilation:

```
$ module unload impi/2017_update-1
$ module load intel/2017_update-1
$ module unload slurm
$ cd fftw-3.3.7
$ ./configure --prefix=/share/apps/fftw/3.3.7/intel-17.0.1 --build=x86_64-
↪redhat-linux --enable-shared -enable-static -enable-sse2 --enable-avx --
↪enable-avx2 --enable-openmp --enable-threads # Without Floating-point␣
↪Precision
$ make clean
$ ./configure --prefix=/share/apps/fftw/3.3.7/intel-17.0.1 --build=x86_64-
↪redhat-linux --enable-shared -enable-static -enable-sse2 --enable-avx --
↪enable-avx2 --enable-openmp --enable-threads --enable-float # With␣
↪Floating-point Precision
$ make -j 16
$ make check
$ sudo mkdir -p /share/apps/fftw/3.3.7/intel-17.0.1
$ sudo chown -R mgomezzul.apolo /share/apps/fftw/3.3.7/intel-17.0.1
$ make install
```

Cronos

1. Download the desired version of the software (Source code - tar.gz)

```
$ cd /home/mgomezzul/apps/fftw/3/intel
$ wget http://www.fftw.org/fftw-3.3.7.tar.gz
$ tar xf fftw-3.3.7.tar.gz
```

2. After unpacking fftw3, continue with the following steps for configuration and compilation:

```
$ module purge
$ module load mpich2/3.2.1_gcc-5.5.0
$ /home/mgomezzul/apps/fftw/src/gcc-5.5.0_mpich2-3.2.1
$ wget http://www.fftw.org/fftw-3.3.7.tar.gz
$ tar xf fftw-3.3.7.tar.gz
$ cd fftw-3.3.7
$ sudo mkdir -p /share/apps/fftw/3.3.7/gcc-5.5.0_mpich2-3.2.1
$ sudo chown -R mgomezzul.apolo /share/apps/fftw/3.3.7/gcc-5.5.0_mpich2-3.
↪2.1
```

Version: parallel, threads and double precision floating point

```
$ ./configure --prefix=/share/apps/fftw/3.3.7/gcc-5.5.0_mpich2-3.2.
↪1 --build=x86_64-redhat-linux --enable-shared --enable-static --
↪enable-threads --enable-openmp --enable-mpi --enable-sse2 --
↪enable-avx 2>&1 | tee fftw3-conf.log
$ make -j 16 2>&1 | tee fftw3-make.log
$ make -j 16 check 2>&1 | tee fftw3-make-check.log
$ make install 2>&1 | tee fftw3-make-install.log
```

Version: parallel, "threads" and simple floating point precision

Note: The idea is to produce the single and double precision version and install them in the same place as the names of the resulting files are different, making it easier for the user to use them.

```
$ make clean
$ ./configure --prefix=/share/apps/fftw/3.3.7/gcc-5.5.0_mpich2-3.2.
↪1 --build=x86_64-redhat-linux --enable-shared --enable-static --
↪enable-threads --enable-openmp --enable-mpi --enable-sse2 --
↪enable-avx  --enable-single 2>&1 | tee fftw3-conf.log
$ make -j 16 2>&1 | tee fftw3-make.log
$ make -j 16 check 2>&1 | tee fftw3-make-check.log
$ make install 2>&1 | tee fftw3-make-install.log
$ sudo chown -R root.root /share/apps/fftw/3.3.7/gcc-5.5.0_mpich2-
↪3.2.1
```

Module

```
#%Module1.0#####################################################################
↪##
##
## module load fftw/3.3.7_intel-17.0.1
##
## /share/apps/modules/fftw/3.3.7_intel-17.0.1
## Written by Mateo G
##

proc ModulesHelp {} {
    global version modroot
    puts stderr "Sets the environment for using fftw 3.3.7\
                \nin the shared directory \
                \n/share/apps/fftw/3.3.7/intel-17.0.1\
                \nbuilded with gcc-5.4.0"
}

module-whatis "(Name________) fftw"
module-whatis "(Version_____) 3.3.7"
module-whatis "(Compilers___) intel-17.0.1"
module-whatis "(System______) x86_64-redhat-linux"
module-whatis "(Libraries___) "

# for Tcl script use only
set         topdir        /share/apps/fftw/3.3.7/intel-17.0.1
set         version       3.3.7
```

(continues on next page)

(continued from previous page)

```
set         sys           x86_64-redhat-linux

conflict fftw
module load intel/2017_update-1


prepend-path    PATH                    $topdir/bin

prepend-path    LD_LIBRARY_PATH         $topdir/lib
prepend-path    LIBRARY_PATH            $topdir/lib
prepend-path    LD_RUN_PATH             $topdir/lib

prepend-path    C_INCLUDE_PATH          $topdir/include
prepend-path    CXX_INCLLUDE_PATH       $topdir/include
prepend-path    CPLUS_INCLUDE_PATH      $topdir/include

prepend-path    PKG_CONFIG_PATH         $topdir/lib/pkgconfig

prepend-path    MANPATH                 $topdir/share/man
```

Use

TO-DO

Resources

- http://www.fftw.org/fftw3_doc/

Author

- Mateo Gómez Zuluaga
- Juan Pablo Alcaraz Flórez

FFTW 3.3.10

Table of Contents

Basic information

- **Official Website:** http://www.fftw.org/
- **License:** GNU GENERAL PUBLIC LICENSE Version 2
- **Installed on:** Apolo II
- **Installation date:** 28/03/2022

Tested on (Requirements)

- **OS base:** Rocky Linux 8.5 (x86_64)
- **Dependencies:**
  - GCC >= 11.2.0
  - OpenMPI >= 1.8.8
  - UCX >= 1.12.1

Installation

1. Download the desired version of the software (Source code - tar.gz)

```
$ cd /home/blopezp
$ wget http://www.fftw.org/fftw-3.3.10.tar.gz
$ tar -zxvf fftw-3.3.10.tar.gz
```

2. After unpacking fftw3, continue with the following steps for configuration and compilation:

```
$ cd fftw-3.3.10
$ module load openmpi/4.1.2_gcc-11.2.0
$ ./configure --prefix=/share/apps/fftw/3.3.10/gcc-11.2.0 --build=x86_64-
↪redhat-linux --enable-shared -enable-static -enable-sse2 --enable-avx --
↪enable-avx2 --enable-openmp --enable-threads --enable-float
$ make -j 16 |& tee make.log
$ sudo mkdir -p /share/apps/fftw/3.3.10
$ sudo make install 2>&1 | tee fftw-make-install.log
```

Module

```
##%Module1.0#####################################################################
↪##
##
## module load fftw/3.3.10_gcc-11.2.0
##
## /share/apps/modules/fftw/3.3.7_intel-17.0.1
## Written by Bryan Lopez Parra
##

proc ModulesHelp {} {
    global version modroot
    puts stderr "Sets the environment for using fftw 3.3.7\
```

(continues on next page)

(continued from previous page)

```
                \nin the shared directory \
                \n/share/apps/fftw/3.3.7/intel-17.0.1\
                \nbuilded with gcc-5.4.0"
}

module-whatis "(Name________) fftw"
module-whatis "(Version_____) 3.3.10"
module-whatis "(Compilers___) gcc-11.2.0"
module-whatis "(System______) x86_64-redhat-linux"
module-whatis "(Libraries___) "

# for Tcl script use only
set         topdir        /share/apps/fftw/3.3.10/gcc-11.2.0
set         version       3.3.10
set         sys           x86_64-redhat-linux

conflict fftw
module load gcc/11.2.0
module load openmpi/4.1.2_gcc-11.2.0


prepend-path    PATH                    $topdir/bin

prepend-path    LD_LIBRARY_PATH         $topdir/lib
prepend-path    LIBRARY_PATH            $topdir/lib
prepend-path    LD_RUN_PATH             $topdir/lib

prepend-path    C_INCLUDE_PATH          $topdir/include
prepend-path    CXX_INCLLUDE_PATH       $topdir/include
prepend-path    CPLUS_INCLUDE_PATH      $topdir/include

prepend-path    PKG_CONFIG_PATH         $topdir/lib/pkgconfig

prepend-path    MANPATH                 $topdir/share/man
```

Use

$ module load fftw/3.3.10_gcc-11.2.0

Resources

- http://www.fftw.org/fftw3_doc/

Author

- Bryan López Parra <blopezp@eafit.edu.co>

3.8.7 Beagle

BEAGLE is a high-performance library that can perform the core calculations at the heart of most Bayesian and Maximum Likelihood phylogenetics packages. It can make use of highly-parallel processors such as those in graphics cards (GPUs) found in many PCs.

The project involves an open API and fast implementations of a library for evaluating phylogenetic likelihoods (continuous time Markov processes) of biomolecular sequence evolution.

The aim is to provide high performance evaluation 'services' to a wide range of phylogenetic software, both Bayesian samplers and Maximum Likelihood optimizers. This allows these packages to make use of implementations that make use of optimized hardware such as graphics processing units.

Beagle 2.1.2

Table of Contents

Basic information

- **Official Website:** https://github.com/beagle-dev/beagle-lib
- **License:** GNU GENERAL PUBLIC LICENSE Version 3
- **Installed on:** Apolo II and Cronos
- **Installation date:** 08/02/2017

Tested on (Requirements)

- **OS base:** CentOS (x86_64) $\geq$ 6.6 (Rocks 6.2)
- **Dependencies:**
  - Intel Parallel Studio XE Cluster Edition (2017-U1)
  - Cuda >= 7.0 (Opcional, soporte aceleradora)
  - Controlador NVIDIA en el Master (Opcional, soporte aceleradora)

Installation

CUDA Support

1. Download the desired version of the software (Source code - tar.gz)

```
cd /home/mgomezzul/apps/bpp/intel
wget https://github.com/beagle-dev/beagle-lib/archive/beagle_release_2_1_
↪2.tar.gz
tar -zxvf beagle_release_2_1_2.tar.gz
```

2. After unpacking beagle, continue with the following steps for configuration and compilation:

```
cd beagle-lib
module load intel/2017_update-1
module load cuda/7.0
module load java/jdk-1.8.0_112
./configure --prefix=/share/apps/beagle-lib/2.1.2/cuda/7.0/intel/2017_
↪update-1 --build=x86_64-redhat-linux --enable-static --with-cuda=/share/
↪apps/cuda/7.0 2>&1 | tee beagle-conf.log
# Ignorar el comentario de que OpenCL no puede ser detectado
make 2>&1 | tee beagle-make.log
sudo mkdir -p /share/apps/beagle-lib/2.1.2/cuda/7.0/intel/2017_update-1
sudo chown -R mgomezzul.apolo /share/apps/beagle-lib/2.1.2/cuda/7.0/intel/
↪2017_update-1
make install 2>&1 | tee beagle-make-install.log
sudo chown -R root.root /share/apps/beagle-lib/2.1.2/cuda/7.0/intel/2017_
↪update-1
```

CPU Support

1. Download the desired version of the software (Source code - tar.gz)

```
cd /home/mgomezzul/apps/bpp/intel
wget https://github.com/beagle-dev/beagle-lib/archive/beagle_release_2_1_
↪2.tar.gz
tar -zxvf beagle_release_2_1_2.tar.gz
```

2. After unpacking beagle, continue with the following steps for configuration and compilation:

```
cd beagle-lib
module load intel/2017_update-1
module load java/jdk-1.8.0_112
./configure --prefix=/share/apps/beagle-lib/2.1.2/intel/2017_update-1 --
↪build=x86_64-redhat-linux --enable-static 2>&1 | tee beagle-conf.log
# Ignorar el comentario de que OpenCL no puede ser detectado
make 2>&1 | tee beagle-make.log
sudo mkdir -p /share/apps/beagle-lib/2.1.2/intel/2017_update-1
sudo chown -R mgomezzul.apolo /share/apps/beagle-lib/2.1.2/intel/2017_
↪update-1
make install 2>&1 | tee beagle-make-install.log
sudo chown -R root.root /share/apps/beagle-lib/2.1.2/intel/2017_update-1
```

Module

CUDA

```
#%Module1.0#####################################################################
##
## module beagle-lib/2.1.2_cuda-7.0_intel-2017_update-1
##
## /share/apps/modules/beagle-lib/2.1.2_cuda-7.0_intel-2017_update-1     Written by Mateo Gomez-Zuluaga
##

proc ModulesHelp { } {
    puts stderr "\tzlib/1.2.11 - sets the Environment for Beagle-lib in \
    \n\tthe share directory /share/apps/beagle-lib/2.1.2/cuda/7.0/intel/2017_update-1\n"
}

module-whatis "\n\n\tSets the environment for using Beagle-lib 2.1.2 \
            \n\tbuilded with Intel Parallel Studio XE 2017 and CUDA 7.0\n"

# for Tcl script use only
set       topdir     /share/apps/beagle-lib/2.1.2/cuda/7.0/intel/2017_update-1
set       version    2.1.2
set       sys        x86_64-redhat-linux

module load intel/2017_update-1
module load java/jdk-1.8.0_112
module load cuda/7.0

prepend-path LD_LIBRARY_PATH    $topdir/lib
prepend-path LIBRARY_PATH       $topdir/lib
prepend-path LD_RUN_PATH        $topdir/lib

prepend-path C_INCLUDE_PATH     $topdir/include
prepend-path CXX_INCLUDE_PATH   $topdir/include
prepend-path CPLUS_INCLUDE_PATH $topdir/include

prepend-path PKG_CONFIG_PATH    $topdir/lib/pkgconfig
```

CPU

```
#%Module1.0#####################################################################
##
## module beagle-lib/2.1.2_intel-2017_update-1
##
## /share/apps/beagle-lib/2.1.2/intel/2017_update-1     Written by Mateo Gomez-Zuluaga
##

proc ModulesHelp { } {
    puts stderr "\tzlib/1.2.11 - sets the Environment for Beagle-lib in \
    \n\tthe share directory /share/apps/beagle-lib/2.1.2/intel/2017_update-1\n"
```

(continues on next page)

(continued from previous page)

```
}

module-whatis "\n\n\tSets the environment for using Beagle-lib 2.1.2 \
            \n\tbuilded with Intel Parallel Studio XE 2017\n"

# for Tcl script use only
set       topdir     /share/apps/beagle-lib/2.1.2/intel/2017_update-1
set       version    2.1.2
set       sys        x86_64-redhat-linux

module load intel/2017_update-1
module load java/jdk-1.8.0_112

prepend-path LD_LIBRARY_PATH    $topdir/lib
prepend-path LIBRARY_PATH       $topdir/lib
prepend-path LD_RUN_PATH        $topdir/lib

prepend-path C_INCLUDE_PATH     $topdir/include
prepend-path CXX_INCLUDE_PATH   $topdir/include
prepend-path CPLUS_INCLUDE_PATH $topdir/include

prepend-path PKG_CONFIG_PATH    $topdir/lib/pkgconfig
```

Use

CUDA

```
module load beagle-lib/2.1.2_cuda-7.0_intel-2017_update-1
```

CPU

```
module load beagle-lib/2.1.2_intel-2017_update-1
```

Resources

- https://github.com/beagle-dev/beagle-lib

Author

- Mateo Gómez Zuluaga

3.8.8 Boost

Boost provides free peer-reviewed portable C++ source libraries.

We emphasize libraries that work well with the C++ Standard Library. Boost libraries are intended to be widely useful, and usable across a broad spectrum of applications. The Boost license encourages both commercial and non-commercial use.

We aim to establish "existing practice" and provide reference implementations so that Boost libraries are suitable for eventual standardization. Ten Boost libraries are included in the C++ Standards Committee's Library Technical Report (TR1) and in the new C++11 Standard. C++11 also includes several more Boost libraries in addition to those from TR1. More Boost libraries are proposed for standardization in C++17.

Since 2006 an intimate week long annual conference related to Boost called C++ Now has been held in Aspen, Colorado each May. Boost has been a participant in the annual Google Summer of Code since 2007.

Boost 1.63.0

Table of Contents

Basic information

- **Official Website:** http://www.boost.org/
- **License:** Boost Software License - Version 1.0 - August 17th, 2003
- **Installed on:** Apolo II and Cronos
- **Installation date:** 22/02/2017

Tested on (Requirements)

- **OS base:** CentOS (x86_64) $\geq$ 6.6 (Rocks 6.2)
- **Dependencies:**
  - GNU GCC >= 5.4.0 (64 Bits)
  - OpenMPI >= 1.8.8 (64 Bits)
  - Python >= 2.7.12
  - ICU >= 58.2

Installation

1. Download the desired version of the software (Source code - tar.gz)

```
cd /home/mgomezzul/apps/boost/gcc/5.4.0
wget https://downloads.sourceforge.net/project/boost/boost/1.63.0/boost_1_
↪63_0.tar.gz?r=https%3A%2F%2Fsourceforge.net%2Fprojects%2Fboost%2Ffiles
↪%2Fboost%2F1.63.0%2F&ts=1487777375&use_mirror=superb-dca2
tar -zxvf boost_1_63_0.tar.gz
```

2. After unpacking Boost, continue with the following steps for configuration and compilation:

```
cd boost_1_63_0/tools/build/
module load openmpi/1.8.8-x86_64_gcc-5.4.0_cuda-8.0 python/2.7.12_intel-
↪2017_update-1 icu/58.2_gcc-5.4.0
./bootstrap.sh
sudo mkdir -p /share/apps/boost/1.63.0/gcc/5.4.0_openmpi-1.8.8-x86_64_
↪cuda-8.0
sudo chown -R mgomezzul.apolo /share/apps/boost/1.63.0/gcc/5.4.0_openmpi-
↪1.8.8-x86_64_cuda-8.0
emacs ~/user-config.jam
...
using mpi ;
...
./bjam install --prefix=/share/apps/boost/1.63.0/gcc/5.4.0_openmpi-1.8.8-
↪x86_64_cuda-8.0
export PATH=/share/apps/boost/1.63.0/gcc/5.4.0_openmpi-1.8.8-x86_64_cuda-
↪8.0/bin:$PATH
cd ../..
bjam --build-dir=/tmp/build-boost toolset=gcc stage address-model=64 --
↪prefix=/share/apps/boost/1.63.0/gcc/5.4.0_openmpi-1.8.8-x86_64_cuda-8.0␣
↪install
```

Module

```
#%Module1.0#####################################################################
↪###
##
## modules boost/1.63.0_gcc-5.4.0_openmpi-1.8.8-x86_64_cuda-8.0
##
## /share/apps/modules/boost/1.63.0_gcc-5.4.0_openmpi-1.8.8-x86_64_cuda-8.0 ␣
↪Written by Mateo Gomez-Zuluaga
##

proc ModulesHelp { } {
    puts stderr "\tboost/1.63.0_gcc-5.4.0_openmpi-1.8.8-x86_64_cuda-8.0 -␣
↪sets the environment for BOOST C++ \
               \n\tLIB in the shared directory /share/apps/boost/1.63.0/
↪intel_impi/2017_update-1\n"
}

module-whatis "\n\n\tSets the environment for using BOOST 1.63.0 \
        \n\tbuilded with gcc-5.4.0, OpenMPI 1.8.8 x86_64, Python 2.7.12␣
↪Intel\n"

# for Tcl script use only
set     topdir     /share/apps/boost/1.63.0/gcc/5.4.0_openmpi-1.8.8-x86_64_
↪cuda-8.0
set     version    1.63.0
```

(continues on next page)

(continued from previous page)

```
set     sys        x86_64-redhat-linux

conflict boost

module load openmpi/1.8.8-x86_64_gcc-5.4.0_cuda-8.0
module load python/2.7.12_intel-2017_update-1
module load icu/58.2_gcc-5.4.0

prepend-path    PATH                    $topdir/bin

prepend-path    C_INCLUDE_PATH          $topdir/include
prepend-path    CXX_INCLUDE_PATH        $topdir/include
prepend-path    CPLUS_INCLUDE_PATH      $topdir/include

prepend-path    LD_LIBRARY_PATH         $topdir/lib
prepend-path    LIBRARY_PATH            $topdir/lib
prepend-path    LD_RUN_PATH             $topdir/lib

setenv          BOOST_HOME              $topdir
setenv          BOOST_INCLUDE_DIR       $topdir/include
setenv          BOOST_LIBRARY           $topdir/lib
```

Use

```
module load boost/1.63.0_gcc-5.4.0_openmpi-1.8.8-x86_64_cuda-8.0
```

Resources

- http://www.boost.org/doc/libs/1_46_1/more/getting_started/unix-variants.html
- https://software.intel.com/en-us/articles/building-boost-with-intel-c-compiler-150
- http://www.boost.org/doc/libs/1_61_0/doc/html/mpi/getting_started.html#mpi.bjam
- http://www.boost.org/build/doc/html/bbv2/overview/invocation.html
- http://kratos-wiki.cimne.upc.edu/index.php/How_to_compile_the_Boost_if_you_want_to_use_MPI

Author

- Mateo Gómez Zuluaga

Boost 1.64.0

Table of Contents

Basic information

- **Official Website:** http://www.boost.org/
- **License:** Boost Software License - Version 1.0 - August 17th, 2003
- **Installed on:** Apolo II and Cronos
- **Installation date:** 18/07/2017

Tested on (Requirements)

- **OS base:** CentOS (x86_64) $\geq$ 6.6 (Rocks 6.2)
- **Dependencies:**
  - GNU GCC >= 5.4.0 (64 Bits)
  - OpenMPI >= 1.8.8 (64 Bits)
  - Python >= 2.7.12
  - ICU >= 58.2

Installation

1. Download the desired version of the software (Source code - tar.gz)

```
cd /home/mgomezzul/apps/boost/gcc/5.4.0
wget https://downloads.sourceforge.net/project/boost/boost/1.64.0/boost_1_
↪64_0.tar.gz?r=https%3A%2F%2Fsourceforge.net%2Fprojects%2Fboost%2Ffiles
↪%2Fboost%2F1.64.0%2F&ts=1487777375&use_mirror=superb-dca2
tar -zxvf boost_1_64_0.tar.gz
```

2. After unpacking Boost, continue with the following steps for configuration and compilation:

```
cd boost_1_64_0/tools/build/
module load openmpi/1.8.8-x86_64_gcc-5.4.0_cuda-8.0 python/2.7.12_intel-
↪2017_update-1 icu/58.2_gcc-5.4.0
./bootstrap.sh
sudo mkdir -p /share/apps/boost/1.64.0/gcc/5.4.0_openmpi-1.8.8-x86_64_
↪cuda-8.0
sudo chown -R mgomezzul.apolo /share/apps/boost/1.64.0/gcc-5.4.0_openmpi-
↪1.8.8_cuda-8.0
emacs ~/user-config.jam
...
using mpi ;
```

(continues on next page)

(continued from previous page)

```
...
./bjam install --prefix=/share/apps/boost/1.64.0/gcc-5.4.0_openmpi-1.8.8_
↪cuda-8.0
export PATH=/share/apps/boost/1.64.0/gcc-5.4.0_openmpi-1.8.8_cuda-8.0/bin:
↪$PATH
cd ../..
bjam --build-dir=/tmp/build-boost toolset=gcc stage address-model=64 --
↪prefix=/share/apps/boost/1.64.0/gcc-5.4.0_openmpi-1.8.8_cuda-8.0 install
```

Module

```
#%Module1.0#####################################################################
↪###
##
## modules boost/1.64.0_gcc-5.4.0_openmpi-1.8.8-x86_64_cuda-8.0
##
## /share/apps/modules/boost/1.64.0_gcc-5.4.0_openmpi-1.8.8-x86_64_cuda-8.0 ␣
↪Written by Mateo Gomez-Zuluaga
##

proc ModulesHelp { } {
    puts stderr "\tboost/1.64.0_gcc-5.4.0_openmpi-1.8.8-x86_64_cuda-8.0 -␣
↪sets the environment for BOOST C++ \
               \n\tLIB in the shared directory /share/apps/boost/1.64.0/
↪intel_impi/2017_update-1\n"
}

module-whatis "\n\n\tSets the environment for using BOOST 1.64.0 \
        \n\tbuilded with gcc-5.4.0, OpenMPI 1.8.8 x86_64, Python 2.7.12␣
↪Intel\n"

# for Tcl script use only
set     topdir     /share/apps/boost/1.63.0/gcc/5.4.0_openmpi-1.8.8-x86_64_
↪cuda-8.0
set     version    1.64.0
set     sys        x86_64-redhat-linux

conflict boost

module load openmpi/1.8.8-x86_64_gcc-5.4.0_cuda-8.0
module load python/2.7.12_intel-2017_update-1
module load icu/58.2_gcc-5.4.0

prepend-path    PATH                    $topdir/bin

prepend-path    C_INCLUDE_PATH          $topdir/include
prepend-path    CXX_INCLUDE_PATH        $topdir/include
prepend-path    CPLUS_INCLUDE_PATH      $topdir/include

prepend-path    LD_LIBRARY_PATH         $topdir/lib
prepend-path    LIBRARY_PATH            $topdir/lib
prepend-path    LD_RUN_PATH             $topdir/lib

setenv          BOOST_HOME              $topdir
```

(continues on next page)

(continued from previous page)

```
setenv          BOOST_INCLUDE_DIR       $topdir/include
setenv          BOOST_LIBRARY           $topdir/lib
```

Use

```
module load boost/1.63.0_gcc-5.4.0_openmpi-1.8.8-x86_64_cuda-8.0
```

Resources

- http://www.boost.org/doc/libs/1_46_1/more/getting_started/unix-variants.html
- https://software.intel.com/en-us/articles/building-boost-with-intel-c-compiler-150
- http://www.boost.org/doc/libs/1_61_0/doc/html/mpi/getting_started.html#mpi.bjam
- http://www.boost.org/build/doc/html/bbv2/overview/invocation.html
- http://kratos-wiki.cimne.upc.edu/index.php/How_to_compile_the_Boost_if_you_want_to_use_MPI

Author

- Mateo Gómez Zuluaga

Boost 1.65.1

Table of Contents

Basic information

- **Official Website:** http://www.boost.org/
- **License:** Boost Software License - Version 1.0 - August 17th, 2003
- **Installed on:** Apolo II and Cronos
- **Installation date:** 10/11/2017

Tested on (Requirements)

- **OS base:** CentOS (x86_64) $\geq$ 6.6 (Rocks 6.2)
- **Dependencies:**
  - Intel Compile >= intel-17.0.1
  - Python >= 2.7.12
  - ICU >= 58.2

Installation

1. Download the desired version of the software (Source code - tar.gz)

```
$ cd /home/mgomezzul/apps/boost/gcc/5.4.0
$ https://dl.bintray.com/boostorg/release/1.65.1/source/boost_1_65_1.tar.
↪gz
$ tar xf boost_1_65_1.tar.gz
```

2. After unpacking Boost, continue with the following steps for configuration and compilation:

```
$ cd boost_1_65_1
$ module load python/2.7.12_intel-2017_update-1
$ module load icu/58.2_intel-2017_update-1
$ sed -e '/using python/ s@;@: /usr/include/python${PYTHON_VERSION/3*/$
↪{PYTHON_VERSION}m} ;@' -i bootstrap.sh # Correción "Bug"
$ cd tools/build
$ sed -e '/using python/ s@;@: /usr/include/python${PYTHON_VERSION/3*/$
↪{PYTHON_VERSION}m} ;@' -i bootstrap.sh # Correción "Bug"
$ ./bootstrap.sh
$ sudo mkdir -p /share/apps/boost/1.65.1/intel-17.0.1
$ sudo chown -R mgomezzul.apolo /share/apps/boost/1.65.1/intel-17.0.1
$ ./b2 install --prefix=/share/apps/boost/1.65.1/intel-17.0.1
$ export PATH=/share/apps/boost/1.65.1/intel-17.0.1/bin:$PATH
$ cd ../..
$ mkdir /tmp/boost
$ b2 --build-dir=/tmp/boost address-model=64 toolset=intel stage --
↪prefix=/share/apps/boost/1.65.1/intel-17.0.1 install
```

Module

```
##%Module1.0#####################################################################
↪##
##
## module load boost/1.65.1_intel-17.0.1
##
## /share/apps/modules/boost/1.65.1_intel-17.0.1
## Written by Mateo Gómez-Zuluaga
##

proc ModulesHelp {} {
    global version modroot
    puts stderr "Sets the environment for using boost 1.65.1\
```

(continues on next page)

(continued from previous page)

```
                \nin the shared directory /share/apps/boost/1.65.1/intel-17.
↪0.1\
                \nbuilded with Intel Parallel Studio XE 2017 Update-1."
}

module-whatis "(Name_________) boost"
module-whatis "(Version______) 1.65.1"
module-whatis "(Compilers____) intel-17.0.1"
module-whatis "(System_______) x86_64-redhat-linux"
module-whatis "(Libraries____) icu"

# for Tcl script use only
set         topdir        /share/apps/boost/1.65.1/intel-17.0.1
set         version       1.65.1
set         sys           x86_64-redhat-linux

conflict boost
module load intel/2017_update-1
module load icu/58.2_intel-2017_update-1
module load python/2.7.12_intel-2017_update-1


prepend-path    PATH                    $topdir/bin

prepend-path    LD_LIBRARY_PATH         $topdir/lib
prepend-path    LIBRARY_PATH            $topdir/lib
prepend-path    LD_RUN_PATH             $topdir/lib

prepend-path    C_INCLUDE_PATH          $topdir/include
prepend-path    CXX_INCLLUDE_PATH       $topdir/include
prepend-path    CPLUS_INCLUDE_PATH      $topdir/include
```

Resources

- https://software.intel.com/en-us/articles/building-boost-with-intel-c-compiler-150
- http://www.linuxfromscratch.org/blfs/view/cvs/general/boost.html
- http://www.boost.org/doc/libs/1_65_1/more/getting_started/unix-variants.html

Author

- Mateo Gómez Zuluaga
- Juan Pablo Alcaraz Flórez

Boost 1.66.0

Table of Contents

Basic information

- **Official Website:** http://www.boost.org/
- **License:** Boost Software License - Version 1.0 - August 17th, 2003
- **Installed on:** Apolo II and Cronos
- **Installation date:** 19/02/2018

Tested on (Requirements)

- **OS base:** CentOS (x86_64) $\geq$ 6.6 (Rocks 6.2)
- **Dependencies:**
  - GNU Gcc >= 5.5.0
  - OpenMPI >= 1.10.7
  - Python >= 2.7.14
  - ICU = 58.2

Installation

1. Download the desired version of the software (Source code - tar.gz)

```
module load python/2.7.14_intel-18_u1 gcc/5.5.0 openmpi/1.10.7_gcc-5.5.0␣
↪icu/58.2_gcc-5.5.0
cd /home/mgomezzul/apps/boost/src/gcc-5.5.0/boost_1_66_0
wget https://dl.bintray.com/boostorg/release/1.66.0/source/boost_1_66_0.
↪tar.gz
tar xf boost_1_66_0.tar.gz
```

2. After unpacking Boost, continue with the following steps for configuration and compilation:

```
mkdir -p /share/apps/boost/1.66.0/gcc-5.5.0_openmpi-1.10.7
sudo chown -R mgomezzul.apolo /share/apps/boost/1.66.0/gcc-5.5.0_openmpi-
↪1.10.7
cd boost_1_66_0/tools/build
./bootstrap.sh
/b2 install --prefix=/share/apps/boost/1.66.0/gcc-5.5.0_openmpi-1.10.7
export PATH=/share/apps/boost/1.66.0/gcc-5.5.0_openmpi-1.10.7/bin:$PATH
cd ../..
```

(continues on next page)

(continued from previous page)

```
b2 address-model=64 link=static,shared threading=multi toolset=gcc␣
↪variant=release -j 16 -sICU_PATH=/share/apps/icu/58.2/gcc-5.5.0 -sICU_
↪ROOT=/share/apps/icu/58.2/gcc-5.5.0 --prefix=/share/apps/boost/1.66.0/
↪gcc-5.5.0_openmpi-1.10.7 stage install
sudo chown -R root.root /share/apps/boost/1.66.0/gcc-5.5.0_openmpi-1.10.7
```

Module

```
##%Module1.0#####################################################################
↪##
##
## module load boost/1.66.0_gcc-5.5.0_openmpi-1.10.7
##
## /share/apps/modules/boost/1.66.0_gcc-5.5.0_openmpi-1.10.7
## Written by Mateo Gómez Zuluaga
##

proc ModulesHelp {} {
    global version modroot
    puts stderr "Sets the environment for using Boost 1.66.0\
        \nin the shared directory \
        \n/share/apps/boost/1.66.0/gcc-5.5.0_openmpi-1.10.7\
        \nbuilded with gcc-5.5.0, OpenMPI-1.10.7, ICU-58.2\
        \nand python-2.7.14."
}

module-whatis "(Name________) boost"
module-whatis "(Version_____) 1.66.0"
module-whatis "(Compilers___) gcc-5.5.0_openmpi-1.10.7"
module-whatis "(System______) x86_64-redhat-linux"
module-whatis "(Libraries___) icu-58.2"

# for Tcl script use only
set         topdir        /share/apps/boost/1.66.0/gcc-5.5.0_openmpi-1.10.7
set         version       1.66.0
set         sys           x86_64-redhat-linux

conflict boost
module load python/2.7.14_intel-18_u1
module load icu/58.2_gcc-5.5.0
module load openmpi/1.10.7_gcc-5.5.0

prepend-path    PATH                    $topdir/bin

prepend-path    LD_LIBRARY_PATH         $topdir/lib
prepend-path    LIBRARY_PATH            $topdir/lib
prepend-path    LD_RUN_PATH             $topdir/lib

prepend-path    C_INCLUDE_PATH          $topdir/include
prepend-path    CXX_INCLUDE_PATH        $topdir/include
prepend-path    CPLUS_INCLUDE_PATH      $topdir/include
```

Resources

- https://software.intel.com/en-us/articles/building-boost-with-intel-c-compiler-150
- http://www.linuxfromscratch.org/blfs/view/cvs/general/boost.html
- http://www.boost.org/doc/libs/1_66_0/more/getting_started/unix-variants.html

Author

- Mateo Gómez Zuluaga
- Juan Pablo Alcaraz Flórez

Boost 1.67.0

Table of Contents

Basic information

- **Official Website:** http://www.boost.org/
- **License:** Boost Software License - Version 1.0 - August 17th, 2003
- **Installed on:** Apolo II and Cronos
- **Installation date:** 19/02/2018

Tested on (Requirements)

- **OS base:** CentOS (x86_64) $\geq$ 6.6 (Rocks 6.2)
- **Dependencies:**
  - Intel Parallel Studio XE Cluster Edition (Intel MPI) >= 2017 Update 1
  - Python2 (Intel Version) >= 2.7.14
  - ICU >= 58.2

- GNU GCC >= 5.4.0
- OpenMPI >= 1.10.7
- Python2 >= 2.7.14
- ICU >= 58.2

Installation

Intel

1. Download the desired version of the software (Source code - tar.gz)

```
module load intel/2017_update-1 impi/2017_update-1 mkl/2017_update-1␣
↪python/2.7.14_intel-18_u1 gcc/5.5.0 icu/58.2_gcc-5.5.0
cd /home/mgomezzul/apps/boost/src/intel-17.0.1/boost_1_67_0
wget https://dl.bintray.com/boostorg/release/1.67.0/source/boost_1_76_0.
↪tar.gz
tar xf boost_1_67_0.tar.gz
```

2. After unpacking Boost, continue with the following steps for configuration and compilation:

```
mkdir -p /share/apps/boost/1.67.0/intel-17.0.1
sudo chown -R mgomezzul.apolo /share/apps/boost/1.67.0/intel-17.0.1
cd boost_1_67_0/tools/build
sed -e '/using python/ s@;@: /usr/include/python${PYTHON_VERSION/3*/$
↪{PYTHON_VERSION}m} ;@' -i bootstrap.sh
./bootstrap.sh --with-toolset=intel-linux
./b2 install --prefix=/share/apps/boost/1.67.0/intel-17.0.1 toolset=intel-
↪linux
export PATH=/share/apps/boost/1.67.0/intel-17.0.1/bin:$PATH
cd ../..
sed -e '/using python/ s@;@: /usr/include/python${PYTHON_VERSION/3*/$
↪{PYTHON_VERSION}m} ;@' -i bootstrap.sh
mpiicc -show # A partir de esta salida armar el contenido del siguiente␣
↪archivo.
emacs user-config.jam
...
using mpi : mpiicc :
    <library-path>/share/apps/intel/ps_xe/2017_update-1/compilers_and_
↪libraries_2017.1.132/linux/mpi/intel64/lib
    <library-path>/share/apps/intel/ps_xe/2017_update-1/compilers_and_
↪libraries_2017.1.132/linux/mpi/intel64/lib/release_mt
    <include>/share/apps/intel/ps_xe/2017_update-1/compilers_and_
↪libraries_2017.1.132/linux/mpi/intel64/include
    <find-shared-library>mpifort
    <find-shared-library>mpi
    <find-shared-library>mpigi
    <find-shared-library>dl
    <find-shared-library>rt
    <find-shared-library>pthread ;
...
b2 --prefix=/share/apps/boost/1.67.0/intel-17.0.1 toolset=intel-linux␣
↪stage install
sudo chown -R root.root /share/apps/boost/1.67.0/intel-17.0.1
```

Free Software

1. Download the desired version of the software (Source code - tar.gz)

```
module load openmpi/1.10.7_gcc-5.4.0 icu/58.2_gcc-5.5.0
cd /home/mgomezzul/apps/boost/src/gcc-5.4.0
wget https://dl.bintray.com/boostorg/release/1.67.0/source/boost_1_76_0.
↪tar.gz
tar xf boost_1_67_0.tar.gz
```

2. After unpacking Boost, continue with the following steps for configuration and compilation:

```
mkdir -p /share/apps/boost/1.67.0/gcc-5.4.0
sudo chown -R mgomezzul.apolo /share/apps/boost/1.67.0/gcc-5.4.0
cd boost_1_67_0/tools/build
sed -e '/using python/ s@;@: /usr/include/python${PYTHON_VERSION/3*/$
↪{PYTHON_VERSION}m} ;@' -i bootstrap.sh
./bootstrap.sh --with-toolset=gcc
./b2 install --prefix=/share/apps/boost/1.67.0/gcc-5.4.0 toolset=gcc
export PATH=/share/apps/boost/1.67.0/gcc-5.4.0/bin:$PATH
cd ../..
sed -e '/using python/ s@;@: /usr/include/python${PYTHON_VERSION/3*/$
↪{PYTHON_VERSION}m} ;@' -i bootstrap.sh
emacs user-config.jam
...
using mpi ;
...
b2 --prefix=/share/apps/boost/1.67.0/gcc-5.4.0 toolset=gcc stage install
sudo chown -R root.root /share/apps/boost/1.67.0/gcc-5.4.0
```

Module

```
#%Module1.0#####################################################################
↪##
##
## module load boost/1.67.0_intel-17.0.1
##
## /share/apps/modules/boost/1.67.1_intel-17.0.1
## Written by Mateo Gómez-Zuluaga
##

proc ModulesHelp {} {
    global version modroot
    puts stderr "Sets the environment for using boost 1.67.0\
        \nin the shared directory /share/apps/boost/1.67.0/intel-17.0.1\
        \nbuilded with Intel Parallel Studio XE 2017 Update-1."
}

module-whatis "(Name________) boost"
module-whatis "(Version_____) 1.67.0"
module-whatis "(Compilers___) intel-17.0.1"
module-whatis "(System______) x86_64-redhat-linux"
module-whatis "(Libraries___) icu"

# for Tcl script use only
```

(continues on next page)

(continued from previous page)

```
set         topdir        /share/apps/boost/1.67.0/intel-17.0.1
set         version       1.67.0
set         sys           x86_64-redhat-linux

conflict boost
module load intel/2017_update-1
module load icu/58.2_intel-2017_update-1
module load python/2.7.12_intel-2017_update-1
module load impi/2017_update-1
module load mkl/2017_update-1

prepend-path    PATH                    $topdir/bin

prepend-path    LD_LIBRARY_PATH         $topdir/lib
prepend-path    LIBRARY_PATH            $topdir/lib
prepend-path    LD_RUN_PATH             $topdir/lib

prepend-path    C_INCLUDE_PATH          $topdir/include
prepend-path    CXX_INCLLUDE_PATH       $topdir/include
prepend-path    CPLUS_INCLUDE_PATH      $topdir/include
```

Resources

- https://software.intel.com/en-us/articles/building-boost-with-intel-c-compiler-150
- http://www.linuxfromscratch.org/blfs/view/cvs/general/boost.html
- http://www.boost.org/doc/libs/1_66_0/more/getting_started/unix-variants.html
- https://www.boost.org/doc/libs/1_67_0/more/getting_started/unix-variants.html
- https://www.boost.org/doc/libs/1_67_0/doc/html/mpi/getting_started.html
- https://www.boost.org/doc/libs/1_66_0/doc/html/bbv2/overview.html
- http://www.linuxfromscratch.org/blfs/view/8.2/general/boost.html

Author

- Mateo Gómez Zuluaga
- Juan Pablo Alcaraz Flórez

3.8.9 OpenBLAS

In scientific computing, OpenBLAS is an open source implementation of the BLAS (Basic Linear Algebra Subprograms) API with many hand-crafted optimizations for specific processor types. It is developed at the Lab of Parallel Software and Computational Science, ISCAS.

- OpenBLAS adds optimized implementations of linear algebra kernels for several processor architectures, including Intel Sandy Bridge[2] and Loongson.[3] It claims to achieve performance comparable to the Intel MKL.
- OpenBLAS is a fork of GotoBLAS2, which was created by Kazushige Goto at the Texas Advanced Computing Center.

OpenBLAS 0.2.19

Table of Contents

Basic information

- **Official Website:** https://www.openblas.net/
- **License:** 3-clause BSD license
- **Installed on:** Apolo II and Cronos
- **Installation date:** 21/02/2017

Tested on (Requirements)

- **OS base:** CentOS (x86_64) $\geq$ 6.6 (Rocks 6.2)
- **Dependencies:**
  - GNU GCC >= 5.4.0

Installation

1. Download the desired version of the software (Source code - tar.gz)

```
cd /home/mgomezzul/apps/openblas/src
wget http://github.com/xianyi/OpenBLAS/archive/v0.2.19.tar.gz
tar -zxvf v0.2.19.tar.gz
```

2. After unpacking OpenBLAS, continue with the following steps for configuration and compilation:

```
cd OpenBLAS-0.2.19/
echo add LAPACK support
wget http://www.netlib.org/lapack/lapack-3.7.0.tgz
sudo mkdir -p /share/apps/openblas/0.2.19/gcc/5.4.0
sudo chown -R mgomezzul.apolo /share/apps/openblas/0.2.19/gcc/5.4.0
module load gcc/5.4.0
make
make PREFIX=/share/apps/openblas/0.2.19/gcc/5.4.0 install
```

Module

```
#%Module1.0#####################################################################
###
##
## modules openblas/0.2.19_gcc-5.4.0
##
## /share/apps/modules/openblas/0.2.19_gcc-5.4.0  Written by Mateo Gomez-Zuluaga
##

proc ModulesHelp { } {
    puts stderr "\topenblas-0.2.19_gcc-5.4.0 - sets the Environment for OpenBLAS 0.2.19 in \
                \n\tthe shared directory /share/apps/openblas/0.2.19/gcc-5.4.0 \n"
}

module-whatis "\n\n\tSets the environment for using OpenBLAS 0.2.19 \
            \n\tbuilded with \
            \n\tand gcc 5.4.0 version\n"

# for Tcl script use only
set   topdir      /share/apps/openblas/0.2.19/gcc/5.4.0
set   version     0.2.19
set   sys         x86_64-redhat-linux


module load gcc/5.4.0

prepend-path LD_LIBRARY_PATH    $topdir/lib
prepend-path LIBRARY_PATH       $topdir/lib
prepend-path LD_RUN_PATH        $topdir/lib

prepend-path C_INCLUDE_PATH     $topdir/include
prepend-path CXX_INCLUDE_PATH   $topdir/include
prepend-path CPLUS_INCLUDE_PATH $topdir/include
```

Use

```
module load openblas/0.2.19_gcc-5.4.0
```

Resources

- https://www.openblas.net/

Author

- Mateo Gómez Zuluaga

3.8.10 ICU

ICU is a mature, widely used set of C/C++ and Java libraries providing Unicode and Globalization support for software applications. ICU is widely portable and gives applications the same results on all platforms and between C/C++ and Java software. ICU is released under a nonrestrictive open source license that is suitable for use with both commercial software and with other open source or free software.

Here are a few highlights of the services provided by ICU:

Code Page Conversion: Convert text data to or from Unicode and nearly any other character set or encoding. ICU's conversion tables are based on charset data collected by IBM over the course of many decades, and is the most complete available anywhere.

Collation: Compare strings according to the conventions and standards of a particular language, region or country. ICU's collation is based on the Unicode Collation Algorithm plus locale-specific comparison rules from the Common Locale Data Repository, a comprehensive source for this type of data.

Formatting: Format numbers, dates, times and currency amounts according the conventions of a chosen locale. This includes translating month and day names into the selected language, choosing appropriate abbreviations, ordering fields correctly, etc. This data also comes from the Common Locale Data Repository.

Time Calculations: Multiple types of calendars are provided beyond the traditional Gregorian calendar. A thorough set of timezone calculation APIs are provided.

Unicode Support: ICU closely tracks the Unicode standard, providing easy access to all of the many Unicode character properties, Unicode Normalization, Case Folding and other fundamental operations as specified by the Unicode Standard.

Regular Expression: ICU's regular expressions fully support Unicode while providing very competitive performance.

Bidi: support for handling text containing a mixture of left to right (English) and right to left (Arabic or Hebrew) data.

Text Boundaries: Locate the positions of words, sentences, paragraphs within a range of text, or identify locations that would be suitable for line wrapping when displaying the text.

ICU 58.2

Table of Contents

Basic information

- **Official Website:** http://site.icu-project.org/
- **License:** Unicode license
- **Installed on:** Apolo II and Cronos
- **Installation date:** 21/02/2017

Tested on (Requirements)

- **OS base:** CentOS (x86_64) $\geq$ 6.6 (Rocks 6.2)
- **Dependencies:**
  - GNU GCC >= 5.4.0

Installation

1. Download the desired version of the software (Source code - tar.gz)

```
cd /home/mgomezzul/apps/icu/gcc/5.4.0
wget http://download.icu-project.org/files/icu4c/58.2/icu4c-58_2-src.tgz
tar -zxvf icu4c-58_2-src.tgz
```

2. After unpacking ICU, continue with the following steps for configuration and compilation:

```
cd icu/source
module load gcc/5.4.0
./configure --prefix=/share/apps/icu/58.2/gcc/5.4.0 --build=x86_64-redhat-
↪linux --enable-static --with-library-bits=64 2>&1 | tee icu-conf.log
make 2>&1 | tee icu-make.log
sudo mkdir -p /share/apps/icu/58.2/gcc/5.4.0
sudo chown -R mgomezzul.apolo /share/apps/beagle-lib/2.1.2/intel/2017_
↪update-1
make install 2>&1 | tee icu-make-install.log
sudo chown -R root.root /share/apps/icu/58.2/gcc/5.4.0
```

Module

```
##%Module1.0#####################################################################
↪###
##
## module icu/58.2_gcc-5.4.0
##
## /share/apps/modules/icu/58.2_gcc-5.4.0 Written by Mateo Gomez-Zuluaga
##

proc ModulesHelp { } {
    puts stderr "\ticu/58.2_gcc-5.4.0 - sets the Environment for ICU in \
    \n\tthe share directory /share/apps/icu/58.2/gcc/5.4.0\n"
}
```

(continues on next page)

(continued from previous page)

```
module-whatis "\n\n\tSets the environment for using ICU-58.2 \
            \n\tbuilded with GNU GCC 5.4.0\n"

# for Tcl script use only
set       topdir     /share/apps/icu/58.2/gcc/5.4.0
set       version    58.2
set       sys        x86_64-redhat-linux

module load gcc/5.4.0

prepend-path PATH               $topdir/bin
prepend-path PATH               $topdir/sbin

prepend-path LD_LIBRARY_PATH    $topdir/lib
prepend-path LIBRARY_PATH       $topdir/lib
prepend-path LD_RUN_PATH        $topdir/lib

prepend-path C_INCLUDE_PATH     $topdir/include
prepend-path CXX_INCLUDE_PATH   $topdir/include
prepend-path CPLUS_INCLUDE_PATH $topdir/include

prepend-path PKG_CONFIG_PATH    $topdir/lib/pkgconfig

prepend-path MANPATH            $topdir/share/man
```

Use

```
module load icu/58.2_gcc-5.4.0
```

Resources

- http://site.icu-project.org/

Author

- Mateo Gómez Zuluaga

3.8.11 HTSeq

HTSeq[1] is a Python package that provides infrastructure to process data from high-throughput sequencing assays.

HTSeq 0.10.0

Table of Contents

[1] https://htseq.readthedocs.io/en/release_0.10.0/overview.html

Basic information

- **Instalation date:** 12/07/2018
- **Official Website:** https://htseq.readthedocs.io/en/release_0.10.0/
- **Supercomputer:** Cronos
- **License:** GNU General Public License

Prerequisites

- Python 2.7, Python 3.4 o superior
- numpy
- Pysam
- matplotlib (Just in case you want to generate graphics)

Installation

1. Charge Conda's environment:

```
$ conda create --name <envName> python=3.6
$ source activate <envName>
```

2. Install numpy in the environment

```
$ conda install numpy
```

3. Install Pysam:

```
$ conda config --add channels r
$ conda config --add channels bioconda
$ conda install pysam
```

4. Download the official HTSeq repository (Source code - tar.gz)[1]

```
$ cd /home/mgomezzul/apps/
$ git clone https://github.com/simon-anders/htseq.git
```

5. Install HTSeq with the corresponding conda environment active:

[1] https://github.com/simon-anders/htseq

```
$ cd htseq/
$ python setup.py build
$ python setup.py install
```

In this way HTSeq is installed in the Conda environment, and added to the Conda Python path.

Mode of use

```
$ source activate <envName>
```

This will load the Conda environment and the HTSeq routes will be added to the Path.

References

Author

Andrés Felipe Zapata Palacio

3.8.12 Jasper

The JasPer Project[1] is an open-source initiative to provide a free software-based reference implementation of the codec specified in the JPEG-2000 Part-1 standard (i.e., ISO/IEC 15444-1).

Jasper 1.900.1

Table of Contents

Basic information

- **Instalation date:** 13/02/2018
- **Official Website:** https://www.ece.uvic.ca/~frodo/jasper/
- **Supercomputer:** Cronos

[1] https://www.ece.uvic.ca/~frodo/jasper/

- **License:** JasPer License Version 2.0

Prerequisites

- Python 2.7, Python 3.4 or higher
- numpy
- Pysam
- matplotlib (Just in case you want to generate graphics)

Installation

1. Load the necessary modules for compilation

```
$ module purge
$ module load gcc/5.5.0
```

2. Download the desired version of the software (Source code - tar.gz)[1]

```
$ conda install numpy
```

3. After decompressing JASPER, continue with the following steps for configuration and compilation:

```
$ cd jasper-1.900.1

$./configure --prefix=/share/apps/jasper/1.900.1/gcc-5.5.0 --build=x86_64-
↪redhat-linux --enable-shared

$ make -j 10 2>&1 | tee jasper-make.log
$ make check 2>&1 | tee jasper-make-check.log
$ sudo mkdir -p /share/apps/jasper/1.900.1/gcc-5.5.0
$ sudo chown -R mgomezzul.apolo /share/apps/jasper/1.900.1/gcc-5.5.0
$ make install 2>&1 | tee jasper-make-install.log
$ sudo chown -R root.root /share/apps/jasper/1.900.1/gcc-5.5.0
```

Module

```
#%Module1.0#####################################################################
↪##
##
## module load jasper/1.900.1_gcc-5.5.0
##
## /share/apps/modules/jasper/1.900.1_gcc-5.5.0
## Written by Mateo Gómez-Zuluaga
##

proc ModulesHelp {} {
    global version modroot
    puts stderr "Sets the environment for using JasPer-1.900.1\
                \nin the shared directory /share/apps/jasper/1.900.1/gcc-5.5.
↪0\
```

(continues on next page)

[1] https://www.ece.uvic.ca/~frodo/jasper/#download

(continued from previous page)

```
                \nbuilded with gcc-5.5.0"
}

module-whatis "(Name________) jasper"
module-whatis "(Version_____) 1.900.1"
module-whatis "(Compilers___) gcc-5.5.0"
module-whatis "(System______) x86_64-redhat-linux"
module-whatis "(Libraries___) "

# for Tcl script use only
set         topdir        /share/apps/jasper/1.900.1/gcc-5.5.0
set         version       1.900.1
set         sys           x86_64-redhat-linux

conflict jasper
module load gcc/5.5.0


prepend-path    PATH                    $topdir/bin

prepend-path    LD_LIBRARY_PATH         $topdir/lib
prepend-path    LIBRARY_PATH            $topdir/lib
prepend-path    LD_RUN_PATH             $topdir/lib

prepend-path    C_INCLUDE_PATH          $topdir/include
prepend-path    CXX_INCLUDE_PATH        $topdir/include
prepend-path    CPLUS_INCLUDE_PATH      $topdir/include
```

Mode of use

```
$ module load jasper/1.900.1_gcc-5.5.0
```

References

Author

Andrés Felipe Zapata Palacio

Jasper 1.900.1

Table of Contents

Basic information

- **Instalation date:** June 2020
- **Official Website:** https://www.ece.uvic.ca/~frodo/jasper/
- **Supercomputer:** Apolo II
- **License:** JasPer License Version 2.0

Prerequisites

- Python 2.7, Python 3.4 or higher
- numpy
- Pysam
- matplotlib (Just in case you want to generate graphics)

Installation

1. Load the necessary modules for compilation

```
$ module load intel/19.0.4
```

2. Download the desired version of the software (Source code - tar.gz)[1]

```
$ wget https://www.ece.uvic.ca/~frodo/jasper/software/jasper-1.900.1.zip
$ unzip jasper-1.900.1.zip
```

3. After decompressing JASPER, continue with the following steps for configuration and compilation:

```
$ cd jasper-1.900.1
$ ./configure --prefix=/share/apps/jasper/1.900.1/intel-19.0.4 --
↪build=x86_64-redhat-linux
$ make -j 10 2>&1 | tee jasper-make.log
$ make check 2>&1 | tee jasper-make-check.log
$ sudo mkdir -p /share/apps/jasper/1.900.1/intel-19.0.4
$ make install 2>&1 | tee jasper-make-install.log
```

Module

[1] https://www.ece.uvic.ca/~frodo/jasper/#download

```
##%Module1.0#####################################################################
↪##
##
## module load jasper/1.900.1_intel-19.0.4
##
## /share/apps/modules/jasper/1.900.1_intel-2017_update-1
## Written by Tomas Navarro & Santiago Alzate
##

proc ModulesHelp {} {
     global version modroot
     puts stderr "Sets the environment for using jasper 1in the shared␣
↪directory /share/apps/jasper/1.900.1/intel-19.0.4/\
                  \nbuilded with Intel Parallel Studio XE .900.1\
                  \n2017\n"
}

module-whatis "(Name________) jasper"
module-whatis "(Version_____) 1.900.1"
module-whatis "(Compilers___) intel-2019_update-4"
module-whatis "(System______) x86_64-redhat-linux"
module-whatis "(Libraries___) "

# for Tcl script use only
set         topdir        /share/apps/jasper/1.900.1/intel-19.0.4
set         version       1.900.1
set         sys           x86_64-redhat-linux

setenv          JASPERLIB               $topdir/lib
setenv          JASPERINC               $topdir/include

prepend-path    PATH                    $topdir/bin

prepend-path    LD_LIBRARY_PATH         $topdir/lib
prepend-path    LIBRARY_PATH            $topdir/lib
prepend-path    LD_RUN_PATH             $topdir/lib

prepend-path    C_INCLUDE_PATH          $topdir/include
prepend-path    CXX_INCLLUDE_PATH       $topdir/include
prepend-path    CPLUS_INCLUDE_PATH      $topdir/include
```

Mode of use

```
$ module load jasper/1.900.1_intel-19.0.4
```

References

Author

- Tomas David Navarro
- Santiago Alzate Cardona

Jasper 3.0.3

Table of Contents

Basic information

- **Instalation date:** 15/03/2022
- **Official Website:** https://www.ece.uvic.ca/~frodo/jasper/
- **Supercomputer:** Apolo II
- **License:** JasPer License Version 3.0

Prerequisites

- GCC >= 9.3.0
- cmake >= 3.20.2

Installation

1. Load the necessary modules for compilation

```
$ module load gcc/9.3.0
```

2. Download the desired version of the software (Source code - tar.gz)[1]

```
$ cd /home/blopezp
$ wget https://github.com/jasper-software/jasper/archive/refs/tags/
↪version-3.0.3.tar.gz
```

3. After decompressing JASPER, continue with the following steps for configuration and compilation:

```
$ cd jasper-version-3.0.3
$ mkdir build2
$cd build2

$ make .. -DCMAKE_INSTALL_PREFIX=/share/apps/jasper/3.0.3/gcc-9.3.0
```

(continues on next page)

[1] https://www.ece.uvic.ca/~frodo/jasper/#download

(continued from previous page)

```
$ make -j 10 2>&1 | tee jasper-make.log
$ make check 2>&1 | tee jasper-make-check.log
$ sudo mkdir -p /share/apps/jasper/3.0.3
$ sudo make install 2>&1 | tee jasper-make-install.log
```

Module

```
##%Module1.0#####################################################################
##
## module load jasper/3.0.3_gcc-9.3.0
##
## /share/apps/jasper/3.0.3/gcc-9.3.0
## Written by Bryan Lopez Parra
##

proc ModulesHelp {} {
     global version modroot
     puts stderr "Sets the environment for using jasper 1in the shared␣
↪directory /share/apps/jasper/3.0.3/gcc-9.3.0/\
                  \nbuilded with gcc 9.3.0\
                  \n2019\n"
}

module-whatis "(Name________) jasper"
module-whatis "(Version_____) 3.0.3"
module-whatis "(Compilers___) gcc-9.3.0"
module-whatis "(System______) x86_64-redhat-linux"
module-whatis "(Libraries___) "

# for Tcl script use only
set         topdir        /share/apps/jasper/3.0.3
set         version       3.0.3
set         sys           x86_64-redhat-linux

module load gcc/9.3.0

setenv          JASPERLIB               $topdir/lib
setenv          JASPERINC               $topdir/include

prepend-path    PATH                    $topdir/bin

prepend-path    LD_LIBRARY_PATH         $topdir/lib
prepend-path    LIBRARY_PATH            $topdir/lib
prepend-path    LD_RUN_PATH             $topdir/lib

prepend-path    C_INCLUDE_PATH          $topdir/include
prepend-path    CXX_INCLLUDE_PATH       $topdir/include
prepend-path    CPLUS_INCLUDE_PATH      $topdir/include
```

Mode of use

```
$ module load jasper/3.0.3_gcc-9.3.0
```

References

Author

- Bryan López Parra <blopezp@eafit.edu.co>

3.8.13 Qt framework

Qt is a cross-platform object-oriented framework widely used to develop programs (software) that use a graphical user interface, as well as different types of tools for the command line and consoles for servers that do not need a graphical user interface.

Qt is developed as a free and open source software through Qt Project, where the community participates, as well as developers from Nokia, Digia and other companies.3 Previously, it was developed by Nokia's Qt software division, which entered effective after Nokia's acquisition of the Norwegian company Trolltech, the original producer of Qt, on June 17, 2008.4 Qt is distributed under the terms of GNU Lesser General Public License and others. On the other hand, Digia has been in charge of Qt commercial licenses since March 2011.

Qt is used in KDE, desktop environment for systems such as GNU / Linux or FreeBSD, among others.

Qt 4.8.7

Table of Contents

Basic information

- **Official Website:** https://www.qt.io/
- **License:** LGPL
- **Installed on:** Apolo II and Cronos
- **Installation date:** 27/02/2017

Tested on (Requirements)

- **OS base:** CentOS (x86_64) $\geq$ 6.6 (Rocks 6.2)
- **Dependencies:**
  - GNU GCC >= 5.5.0

Installation

1. Download the desired version of the software (Source code - tar.gz)

```
cd /home/mgomezzul/apps/qt/intel/
wget https://download.qt.io/official_releases/qt/4.8/4.8.7/qt-everywhere-
↪opensource-src-4.8.7.tar.gz
tar -zxvf qt-everywhere-opensource-src-4.8.7.tar.gz
```

2. After unpacking Qt, continue with the following steps for configuration and compilation:

```
module purge
module load gcc/5.5.0
cd qt-everywhere-opensource-src-4.8.7
configure -prefix /share/apps/qt/4.8.7/gcc-5.5.0 -release -system-zlib -
↪no-gui -no-webkit -opensource -nomake examples -nomake demos -nomake␣
↪tests -optimized-qmake -confirm-license -platform linux-g++ -no-
↪qt3support
make 2>&1 | tee make-qt.log
make check 2>&1 | tee make-check-qt.log
```

3. Installation of compiled binaries

```
sudo mkdir -p /share/apps/qt/4.8.7/gcc-5.5.0
sudo chmod -R mgomezzul.apolo /share/apps/qt/4.8.7/gcc-5.5.0
make install 2>&1 | tee make-install-qt.log
```

Module

```
##%Module1.0#####################################################################
↪##
##
## module load qt/4.8.7_gcc-5.5.0
##
## /share/apps/modules/qt/4.8.7_gcc-5.5.0
## Written by Mateo G;ómez-Zuluaga
##

proc ModulesHelp {} {
    global version modroot
    puts stderr "Sets the environment for using QT 4.8.7\
        \nin the shared directory /share/apps/qt/4.8.7/gcc-5.5.0\
        \nbuilded with gcc-5.4.0."
}

module-whatis "(Name________) qt"
```

(continues on next page)

(continued from previous page)

```
module-whatis "(Version_____) 4.8.7"
module-whatis "(Compilers___) gcc-5.5.0"
module-whatis "(System______) x86_64-redhat-linux"
module-whatis "(Libraries___) "

# for Tcl script use only
set         topdir        /share/apps/qt/4.8.7/gcc-5.5.0
set         version       4.8.7
set         sys           x86_64-redhat-linux

conflict qt
module load gcc/5.5.0

setenv          QTDIR                   $topdir
setenv          QTINC                   $topdir/include
setenv          QTLIB                   $topdir/lib

prepend-path    PATH                    $topdir/bin

prepend-path    LD_LIBRARY_PATH         $topdir/lib
prepend-path    LIBRARY_PATH            $topdir/lib
prepend-path    LD_RUN_PATH             $topdir/lib

prepend-path    C_INCLUDE_PATH          $topdir/include
prepend-path    CXX_INCLUDE_PATH        $topdir/include
prepend-path    CPLUS_INCLUDE_PATH      $topdir/include

prepend-path    PKG_CONFIG_PATH         $topdir/lib/pkgconfig
```

Use

```
module load qt/4.8.7_gcc-5.5.0
```

Resources

- https://raw.githubusercontent.com/OpenFOAM/ThirdParty-2.4.x/master/makeQt
- https://openfoamwiki.net/index.php/Installation/Linux/OpenFOAM-4.0/CentOS_SL_RHEL
- http://www.linuxfromscratch.org/blfs/view/7.8/x/qt4.html
- https://software.intel.com/en-us/articles/build-qt-libraries-with-intel-compiler-linux

Author

- Mateo Gómez Zuluaga

3.8.14 Hypre

Livermore's HYPRE library of linear solvers makes possible larger, more detailed simulations by solving problems faster than traditional methods at large scales. It offers a comprehensive suite of scalable solvers for large-scale

scientific simulation, featuring parallel multigrid methods for both structured and unstructured grid problems. The HYPRE library is highly portable and supports a number of languages.

Work on HYPRE began in the late 1990s. It has since been used by research institutions and private companies to simulate groundwater flow, magnetic fusion energy plasmas in tokomaks and stellarators, blood flow through the heart, fluid flow in steam generators for nuclear power plants, and pumping activity in oil reservoirs, to name just a few areas. In 2007, HYPRE won an R&D 100 award from R&D Magazine as one of the year's most significant technological breakthroughs.

The HYPRE team was one of the first to develop algebraic multigrid algorithms and software for extreme-scale parallel supercomputers. The team maintains an active role in the multigrid research community and is recognized for its leadership in both algorithm and software development.

Hypre 2.10.1

Table of Contents

Basic information

- **Official Website:** http://computation.llnl.gov/projects/hypre-scalable-linear-solvers-multigrid-methods
- **License:** GNU Lesser General Public License 2.1
- **Installed on:** Apolo II and Cronos
- **Installation date:** 03/03/2017

Tested on (Requirements)

- **OS base:** CentOS (x86_64) $\geq$ 6.6 (Rocks 6.2)
- **Dependencies:**
  - Intel Parallel Studio XE Cluster Edition 2017 (Update 1) (Intel Compiler, Intel MPI and MKL)

Installation

1. Download the desired version of the software (Source code - tar.gz)

```
cd /home/mgomezzul/apps/hypre/src/intel
wget http://computation.llnl.gov/projects/hypre-scalable-linear-solvers-
↪multigrid-methods/download/hypre-2.10.1.tar.gz
tar -zxvf hypre-2.10.1.tar.gz
```

2. After unpacking Hypre, continue with the following steps for configuration and compilation:

```
module load intel/2017_update-1
module load impi/2017_update-1
module load mkl/2017_update-1
cd hypre-2.10.1/src
./configure --prefix=/share/apps/hypre/2.10.1/intel_impi/2017_update-1 --
↪enable-fortran --with-MPI --with-blas-libs=mkl_intel_ilp64 mkl_
↪sequential mkl_core pthread m dl --with-blas-lib-dirs=/share/apps/intel/
↪ps_xe/2017_update-1/mkl/lib/intel64_lin --with-lapack-libs=mkl_intel_
↪ilp64 mkl_sequential mkl_core pthread m dl --with-lapack-lib-dirs=/
↪share/apps/intel/ps_xe/2017_update-1/mkl/lib/intel64_lin
make -2>&1 | tee make-hypre.log
make check 2>&1 | tee make-check-hypre.log
```

3. Installation of compiled binaries

```
sudo mkdir -p /share/apps/hypre/2.10.1/intel_impi/2017_update-1
sudo chown -R mgomezzul.apolo /share/apps/hypre/2.10.1/intel_impi/2017_
↪update-1
make install 2>&1 | tee make-install-hypre.log
```

Module

```
##%Module1.0#####################################################################
↪###
##
## module hypre/2.10.1_intel_impi-2017_update-1
##
## /share/apps/modules/hypre/2.10.1_intel_impi-2017_update-1    Written by␣
↪Mateo Gomez-Zuluaga
##

proc ModulesHelp { } {
    puts stderr "\thypre/2.10.1_intel-2017_update-1 - sets the Environment␣
↪for Hypre in \
    \n\tthe share directory /share/apps/hypre/2.10.1/intel_impi/2017_update-
↪1\n"
}

module-whatis "\n\n\tSets the environment for using Hypre-2.10.1 \
            \n\tbuilded with Intel Parallel Studio XE 2017 Update 1\n"

conflict hypre

module load intel/2017_update-1
module load impi/2017_update-1
module load mkl/2017_update-1

# for Tcl script use only
```

(continues on next page)

(continued from previous page)

```
set            topdir     /share/apps/hypre/2.10.1/intel_impi/2017_update-1
set            version    2.10.1
set            sys        x86_64-redhat-linux

prepend-path LD_LIBRARY_PATH            $topdir/lib
prepend-path LIBRARY_PATH               $topdir/lib
prepend-path LD_RUN_PATH                $topdir/lib

prepend-path C_INCLUDE_PATH             $topdir/include
prepend-path CXX_INCLUDE_PATH           $topdir/include
prepend-path CPLUS_INCLUDE_PATH         $topdir/include
```

Use

```
module load hypre/2.10.1_intel_impi_2017_update-1
```

Resources

- https://wiki.rc.ufl.edu/doc/Hypre
- https://software.intel.com/en-us/articles/intel-mkl-link-line-advisor

Author

- Mateo Gómez Zuluaga

3.8.15 GSL

The GNU Scientific Library (GSL) is a numerical library for C and C++ programmers. It is free software under the GNU General Public License.

The library provides a wide range of mathematical routines such as random number generators, special functions and least-squares fitting. There are over 1000 functions in total with an extensive test suite.

The complete range of subject areas covered by the library includes,

> Complex Numbers Roots of Polynomials Special Functions Vectors and Matrices Permutations Sorting BLAS Support Linear Algebra Eigensystems Fast Fourier Transforms Quadrature Random Numbers Quasi-Random Sequences Random Distributions Statistics Histograms N-Tuples Monte Carlo Integration Simulated Annealing Differential Equations Interpolation Numerical Differentiation Chebyshev Approximation Series Acceleration Discrete Hankel Transforms Root-Finding Minimization Least-Squares Fitting Physical Constants IEEE Floating-Point Discrete Wavelet Transforms Basis splines Running Statistics Sparse Matrices and Linear Algebra

GSL 2.4

Table of Contents

Basic information

- **Official Website:** https://www.gnu.org/software/gsl/
- **License:** GNU General Public License
- **Installed on:** Apolo II and Cronos
- **Installation date:** 19/09/2017

Tested on (Requirements)

- **OS base:** CentOS (x86_64) $\geq$ 6.6 (Rocks 6.2)
- **Dependencies:**
  - GNU GCC >= 5.4.0

Installation

1. Download the desired version of the software (Source code - tar.gz)

```
cd /home/mgomezzul/apps/gsl/src
wget http://mirror.cedia.org.ec/gnu/gsl/gsl-2.4.tar.gz
tar -zxvf gsl-2.4.tar.gz
cd gsl-2.4
```

2. After unpacking GSL, continue with the following steps for configuration and compilation:

```
module load gcc/5.4.0
./configure --prefix=/share/apps/gsl/2.4/gcc-5.4.0 --build=x86_64-redhat-
↪linux --enable-shared --enable-static 2>&1 | tee gsl-config.log
make 2>&1 | tee make-gsl.log
```

3. Installation of compiled binaries

```
sudo mkdir -p /share/apps/gsl/2.4/gcc-5.4.0
sudo chown -R mgomezzul.apolo /share/apps/gsl/2.4/gcc-5.4.0
make install 2>&1 | tee make-install-gsl.log
make installcheck 2>&1 | tee make-check-gsl.log
sudo chown -R root.root /share/apps/gsl/2.4/gcc-5.4.0
```

Module

```
#%Module1.0#####################################################################
##
## module load gsl/2.4_gcc-5.4.0
##
## /share/apps/modules/gsl/2.4_gcc-5.4.0
##
## Written by Mateo Gómez Zuluaga
##

proc ModulesHelp {} {
    global version modroot
    puts stderr "Sets the environment for using GSL 2.4\
        \nin the shared directory /share/apps/gsl/2.4/gcc-5.4.0\
        \nbuilded with gcc-5.4.0."
}

module-whatis "(Name________) gsl"
module-whatis "(Version_____) 2.4"
module-whatis "(Compilers___) gcc-5.4.0"
module-whatis "(System______) x86_64-redhat-linux"
module-whatis "(Libraries___) "

# for Tcl script use only
set         topdir        /share/apps/gsl/2.4/gcc-5.4.0
set         version       2.4
set         sys           x86_64-redhat-linux

conflict gsl
module load gcc/5.4.0


prepend-path    PATH                    $topdir/bin

prepend-path    LD_LIBRARY_PATH         $topdir/lib
prepend-path    LIBRARY_PATH            $topdir/lib
prepend-path    LD_RUN_PATH             $topdir/lib

prepend-path    C_INCLUDE_PATH          $topdir/include
prepend-path    CXX_INCLLUDE_PATH       $topdir/include
prepend-path    CPLUS_INCLUDE_PATH      $topdir/include

prepend-path    PKG_CONFIG_PATH         $topdir/lib/pkgconfig

prepend-path    MANPATH                 $topdir/share/man
```

Use

```
module load gsl/2.4_gcc-5.4.0
gcc -I/share/apps/gsl/2.4/gcc-5.4.0/include -L/share/apps/gsl/2.4/gcc-5.4.0/
↪lib main.c -o main -lgsl -lgslcblas -lm
```

Resources

- https://www.gnu.org/software/gsl/manual/html_node/Linking-programs-with-the-library.html

Author

- Mateo Gómez Zuluaga

3.8.16 Hdf5

HDF5 is a data model, library, and file format for storing and managing data. It supports an unlimited variety of datatypes, and is designed for flexible and efficient I/O and for high volume and complex data. HDF5 is portable and is extensible, allowing applications to evolve in their use of HDF5. The HDF5 Technology suite includes tools and applications for managing, manipulating, viewing, and analyzing data in the HDF5 format.

Hdf5 1.8.15 Patch 1

Table of Contents

Basic information

- **Official Website:** https://support.hdfgroup.org/ftp/HDF5/releases/hdf5-1.8/hdf5-1.8.15-patch1/
- **License:** GNU GENERAL PUBLIC LICENSE Version 2
- **Installed on:** Apolo II and Cronos
- **Installation date:** 13/02/2018

Tested on (Requirements)

- **OS base:** CentOS (x86_64) $\geq$ 6.6 (Rocks 6.2)
- **Dependencies:**
  - mpich2 v3.2.1
  - Zlib v1.2.11

- Szip v2.1.1

Installation

1. Load required modules.

```
module purge
module load gcc/5.5.0
module load mpich2/3.2.1_gcc-5.5.0
module load szip/2.1.1_gcc-5.5.0
module load zlib/1.2.11_gcc-5.5.0
```

2. Configure the following environment variables that specify the compilers to use:

```
$ export CC=mpicc
$ export CXX=mpic++
$ export FC=mpif90
$ export F90=mpif90
```

3. Download the desired version of the software (Source code - tar.gz)

```
cd /home/mgomezzul/apps/hdf5/src/
wget https://support.hdfgroup.org/ftp/HDF5/releases/hdf5-1.8/hdf5-1.8.15-
↪patch1/src/hdf5-1.8.15-patch1.tar.gz
tar -xvf hdf5-1.8.15-patch1.tar.gz
```

4. After unpacking HDF5, continue with the following steps for configuration and compilation:

```
cd hdf5-1.8.15-patch1

./configure --prefix=/share/apps/hdf5/1.8.15-patch1/gcc-5.5.0_mpich2-3.2.
↪1 --build=x86_64-redhat-linux --enable-fortran --enable-parallel --with-
↪zlib=/share/apps/zlib/1.2.11/gcc-5.5.0 --with-szlib=/share/apps/szip/2.
↪1.1/gcc-5.5.0

make -j 10 2>&1 | tee hdf5-make.log
make check 2>&1 | tee hdf5-make-check.log
sudo mkdir -p /share/apps/hdf5/1.8.15-patch1/gcc-5.5.0_mpich2-3.2.1
sudo chown -R mgomezzul.apolo /share/apps/hdf5/1.8.15-patch1/gcc-5.5.0_
↪mpich2-3.2.1
make install 2>&1 | tee hdf5-make-install.log
sudo chown -R root.root /share/apps/hdf5/1.8.15-patch1/gcc-5.5.0_mpich2-3.
↪2.1
```

Module

```
#%Module1.0#####################################################################
↪##
##
## module load hdf5/1.8.15-patch1_gcc-5.5.0_mpich2-3.2.1
##
## /share/apps/modules/hdf5/1.8.15-patch1_gcc-5.5.0_mpich2-3.2.1
## Written by Mateo Gómez-Zuluaga
##
```

(continues on next page)

(continued from previous page)

```
proc ModulesHelp {} {
    global version modroot
    puts stderr "Sets the environment for using HDF5-1.8.15-patch1\
                \nin the shared directory /share/apps/hdf5/1.8.15-patch1/gcc-
↪5.5.0_mpich2-3.2.1\
                \nbuilded with gcc-5.5.0, mpich2-3.2.1, zlib-1.2.1, szip-2.1.
↪1."
}

module-whatis "(Name________) hdf5"
module-whatis "(Version_____) 1.8.15-patch1"
module-whatis "(Compilers___) gcc-5.5.0_mpich2-3.2.1"
module-whatis "(System______) x86_64-redhat-linux"
module-whatis "(Libraries___) szip-2.1.1, zlib-1.2.11"

# for Tcl script use only
set         topdir        /share/apps/hdf5/1.8.15-patch1/gcc-5.5.0_mpich2-3.
↪2.1
set         version       1.8.15-patch1
set         sys           x86_64-redhat-linux

conflict hdf5
module load mpich2/3.2.1_gcc-5.5.0
module load szip/2.1.1_gcc-5.5.0
module load zlib/1.2.11_gcc-5.5.0

setenv          HDF5                    $topdir

prepend-path    PATH                    $topdir/bin

prepend-path    LD_LIBRARY_PATH         $topdir/lib
prepend-path    LIBRARY_PATH            $topdir/lib
prepend-path    LD_RUN_PATH             $topdir/lib

prepend-path    C_INCLUDE_PATH          $topdir/include
prepend-path    CXX_INCLUDE_PATH        $topdir/include
prepend-path    CPLUS_INCLUDE_PATH      $topdir/include
```

Use

```
module load hdf5/1.8.15-patch1_gcc-5.5.0_mpich2-3.2.1
```

Resources

- https://support.hdfgroup.org/downloads/index.html

Author

- Andrés Felipe Zapata Palacio

Hdf5 1.8.19

Table of Contents

Basic information

- **Official Website:** https://support.hdfgroup.org/HDF5/release/obtainsrc518.html
- **License:** GNU GENERAL PUBLIC LICENSE Version 2
- **Installed on:** Apolo II and Cronos
- **Installation date:** 14/11/2017

Tested on (Requirements)

- **OS base:** CentOS (x86_64) $\geq$ 6.6 (Rocks 6.2)
- **Dependencies:**
  - Intel Parallel Studio XE Cluster Edition >= 17.0.1
  - zlib >= 1.2.11
  - szip >= 2.1

Installation

1. Download the desired version of the software (Source code - tar.gz)

```
cd /home/mgomezzul/apps/hdf5/intel
wget https://support.hdfgroup.org/ftp/HDF5/current18/src/hdf5-1.8.19.tar
tar -xvf hdf5-1.8.19.tar
```

2. After unpacking HDF5, continue with the following steps for configuration and compilation:

```
cd hdf5-1.8.19
module unload slurm
module load intel/2017_update-1 zlib/1.2.11_intel-2017_update-1 szip/2.1_
↪intel-2017_update-1
```

(continues on next page)

(continued from previous page)

```
./configure --prefix=/share/apps/hdf5/1.8.19/intel-2017_update-1 --
↪build=x86_64-redhat-linux --enable-fortran --enable-cxx --with-zlib=/
↪share/apps/zlib/1.2.11 --with-szlib=/share/apps/szip/2.1/intel-2017_
↪update-1 2>&1 | tee hdf5-conf.log
make 2>&1 | tee hdf5-make.log
sudo mkdir -p /share/apps/hdf5/1.8.19/intel-2017_update-1
sudo chown -R mgomezzul.apolo /share/apps/hdf5/1.8.19/intel-2017_update-1
make install 2>&1 | tee hdf5-make-install.log
sudo chown -R root.root /share/apps/hdf5/1.8.19/intel-2017_update-1
```

Module

```
#%Module1.0#####################################################################
↪##
##
## module load hdf5/1.8.19_intel-2017_update-1
##
## /share/apps/modules/hdf5/1.8.19_intel-2017_update-1
## Written by Mateo Gómez-Zuluaga
##

proc ModulesHelp {} {
    global version modroot
    puts stderr "Sets the environment for using hdf5 1.8.19\
        \nin the shared directory /share/apps/hdf5/1.8.19/intel-2017_update-
↪1\
        \nbuilded with intel-17.0.1."
}

module-whatis "(Name________) hdf5"
module-whatis "(Version_____) 1.8.19"
module-whatis "(Compilers___) intel-17.0.1"
module-whatis "(System______) x86_64-redhat-linux"
module-whatis "(Libraries___) zlib, szip"

# for Tcl script use only
set         topdir        /share/apps/hdf5/1.8.19/intel-2017_update-1
set         version       1.8.19
set         sys           x86_64-redhat-linux

conflict hdf5
module load intel/2017_update-1
module load szip/2.1_intel-2017_update-1
module load zlib/1.2.11_intel-2017_update-1


prepend-path    PATH                    $topdir/bin

prepend-path    LD_LIBRARY_PATH         $topdir/lib
prepend-path    LIBRARY_PATH            $topdir/lib
prepend-path    LD_RUN_PATH             $topdir/lib

prepend-path    C_INCLUDE_PATH          $topdir/include
prepend-path    CXX_INCLUDE_PATH        $topdir/include
prepend-path    CPLUS_INCLUDE_PATH      $topdir/include
```

Use

TO-DO

Resources

- https://support.hdfgroup.org/downloads/index.html

Author

- Mateo Gómez Zuluaga

Hdf5 1.12.0

Table of Contents

Basic information

- **Official Website:** https://support.hdfgroup.org/HDF5/release/obtainsrc518.html
- **License:** GNU GENERAL PUBLIC LICENSE Version 2
- **Installed on:** Apolo II
- **Installation date:** 14/11/2017

Tested on (Requirements)

- **OS base:** CentOS (x86_64) $\geq$ 6.6 (Rocks 6.2)
- **Dependencies:**
  - Intel Parallel Studio XE Cluster Edition >= 17.0.1
  - zlib >= 1.2.11
  - szip >= 2.1

Installation

1. Download the desired version of the software (Source code - tar.gz)

```
cd $HOME/apps/hdf5/intel
wget https://hdf-wordpress-1.s3.amazonaws.com/wp-content/uploads/manual/
↪HDF5/HDF5_1_12_0/source/hdf5-1.12.0.tar.gz
tar -xvf hdf5-1.8.19.tar
```

2. After unpacking HDF5, continue with the following steps for configuration and compilation:

```
cd hdf5-1.8.19
module load intel/intel-19.0.4 zlib/1.2.11_intel-19.0.4 szip/2.1.1_intel-
↪19.0.4
./configure -prefix=/share/apps/hdf5/1.12/intel-19.0.4 --with-zlib=/share/
↪apps/zlib/1.2.11/intel_19.0.4 --with-szlib=/share/apps/szip/2.1.1/intel_
↪19.0.4 --enable-fortran --enable-cxx
make 2>&1 | tee hdf5-make.log
sudo mkdir -p /share/apps/hdf5/1.12/intel-19.0.4
make install 2>&1 | tee hdf5-make-install.log
```

Troubleshooting

If you have an erro during the configure that says *Can not find ifort* or another compiler then do the followin:

Please export these variables

```
export CC=icc
export F9X=ifort
export CXX=icpc
```

If you continue to have the error, burn the variables in the ./configure command

```
./configure CC=icc F9X=ifort CXX=icpc ...
```

If you continue to have an error, burn the path of the icc and ifort commands to the configure command

```
./configure CC=/path/to/icc F9X=/path/to/ifort CXX=/path/to/icpc ...
```

Module

```
#%Module1.0#####################################################################
↪##
##
## module load hdf5/1.12_intel-19.0.4
##
## /share/apps/modules/hdf5/1.12_intel-19.0.4
## Written by Tomas Navarro & Santiago Alzate
##

proc ModulesHelp {} {
     global version modroot
     puts stderr "Sets the environment for using hdf5 1.12\
                  \nin the shared directory /share/apps/hdf5/1.12/intel-19.0.
↪4\
```

(continues on next page)

(continued from previous page)

```
                \nbuilded with intel-17.0.1."
}

module-whatis "(Name________) hdf5"
module-whatis "(Version_____) 1.12"
module-whatis "(Compilers___) intel-19.0.4"
module-whatis "(System______) x86_64-redhat-linux"
module-whatis "(Libraries___) zlib, szip"

# for Tcl script use only
set         topdir        /share/apps/hdf5/1.12/intel-19.0.4
set         version       1.8.19
set         sys           x86_64-redhat-linux

conflict hdf5
module load intel
module load szip/2.1.1_intel_19.0.4
module load zlib/1.2.11_intel_19.0.4


prepend-path    PATH                    $topdir/bin

prepend-path    LD_LIBRARY_PATH         $topdir/lib
prepend-path    LIBRARY_PATH            $topdir/lib
prepend-path    LD_RUN_PATH             $topdir/lib

prepend-path    C_INCLUDE_PATH          $topdir/include
prepend-path    CXX_INCLUDE_PATH        $topdir/include
prepend-path    CPLUS_INCLUDE_PATH      $topdir/include
```

Mode of use

```
$ module load hdf5/1.12_intel_19.0.4
```

Resources

- https://support.hdfgroup.org/downloads/index.html

Author

- Tomas David Navarro
- Santiago Alzate Cardona

3.8.17 MPFR

The MPFR library is a C library for multiple-precision floating-point computations with correct rounding. MPFR has continuously been supported by the INRIA and the current main authors come from the Caramba and AriC project-teams at Loria (Nancy, France) and LIP (Lyon, France) respectively; see more on the credit page. MPFR is based on the GMP multiple-precision library.

The main goal of MPFR is to provide a library for multiple-precision floating-point computation which is both efficient and has a well-defined semantics. It copies the good ideas from the ANSI/IEEE-754 standard for double-precision floating-point arithmetic (53-bit significand).[1]

MPFR 3.1.6

Table of Contents

Basic information

- **Official Website:** http://www.mpfr.org/mpfr-3.1.6/
- **License:** Free Software Foundation's GNU General Public License
- **Installed on:** Apolo II and Cronos
- **Installation date:** 01/02/2018

Tested on (Requirements)

- **OS base:** CentOS (x86_64) $\geq$ 6.6 (Rocks 6.2)
- **Dependencies:**
  - GMP 6.1.2

Installation

1. Download the desired version of the software (Source code - tar.gz)

```
cd /home/mgomezzul/apps/mpfr/src/
wget http://www.mpfr.org/mpfr-3.1.6/mpfr-3.1.6.tar.gz
tar -xvf mpfr-3.1.6.tar.gz
```

2. After unpacking MPFR, continue with the following steps for configuration and compilation:

[1] http://www.mpfr.org/

```
cd mpfr-3.1.6

./configure --prefix=/share/apps/mpfr/3.1.6/gcc-4.4.7-18 --build=x86_64-
↪redhat-linux-gnu --enable-thread-safe --with-gmp=/share/apps/gmp/6.1.2/
↪gcc-4.4.7-18 --enable-assert --with-gnu-ld

make -j 10 2>&1 | tee mpfr-make.log
sudo mkdir -p /share/apps/mpfr/3.1.6/gcc-4.4.7-18
sudo chown -R mgomezzul.apolo /share/apps/mpfr/3.1.6/gcc-4.4.7-18
make install 2>&1 | tee mpfr-make-install.log
sudo chown -R root.root /share/apps/mpfr/3.1.6/gcc-4.4.7-18
```

Module

```
#%Module1.0#####################################################################
↪##
##
## module load mpfr/3.1.6_gcc-4.4.7-18
##
## /share/apps/modules/mpfr/3.1.6_gcc-4.4.7-18
## Written by Mateo Gomez Zuluaga
##

proc ModulesHelp {} {
    global version modroot
    puts stderr "Sets the environment for using MPFR 3.1.6\
                \nin the shared directory /share/apps/mpfr/3.1.6/gcc-4.4.7-
↪18\
                \nbuilded with gcc-4.4.7-18"
}

module-whatis "(Name________) mpfr"
module-whatis "(Version_____) 3.1.6"
module-whatis "(Compilers___) gcc-4.4.7-18"
module-whatis "(System______) x86_64-redhat-linux"
module-whatis "(Libraries___) gmp-6.1.2"

# for Tcl script use only
set         topdir        /share/apps/mpfr/3.1.6/gcc-4.4.7-18
set         version       3.1.6
set         sys           x86_64-redhat-linux

conflict mpfr
module load gmp/6.1.2_gcc-4.4.7-18

prepend-path    LD_LIBRARY_PATH         $topdir/lib
prepend-path    LIBRARY_PATH            $topdir/lib
prepend-path    LD_RUN_PATH             $topdir/lib

prepend-path    C_INCLUDE_PATH          $topdir/include
prepend-path    CXX_INCLUDE_PATH        $topdir/include
prepend-path    CPLUS_INCLUDE_PATH      $topdir/include

prepend-path    INFOPATH                $topdir/share/info
```

Use

```
module load mpfr/3.1.6_gcc-4.4.7-18
```

Resources

- http://www.mpfr.org/mpfr-3.1.6/

Author

- Tomas Navarro (Translated document)

MPFR 4.1.0

Table of Contents

Basic information

- **Official Website:** https://www.mpfr.org/mpfr-current/#download
- **License:** Free Software Foundation's GNU General Public License
- **Installed on:** Apolo II
- **Installation date:** 03/03/2022

Tested on (Requirements)

- **OS base:** Rocky Linux 8.5 (x86_64)
- **Dependencies:**
  - GMP 6.2.1

Installation

1. Download the desired version of the software (Source code - tar.gz)

```
$ cd /home/blopezp
$ wget https://www.mpfr.org/mpfr-current/mpfr-4.1.0.tar.gz
$ tar -xvf mpfr-4.1.0.tar.gz
```

2. After unpacking MPFR, continue with the following steps for configuration and compilation:

```
$ cd mpfr-4.1.0

$ ./configure --prefix=/share/apps/mpfr/4.1.0/gcc-8.5.0 --build=x86_64-
↪redhat-linux-gnu --enable-thread-safe --with-gmp=/share/apps/gmp/6.2.1/
↪gcc-8.5.0 --enable-assert --with-gnu-ld

$ make -j 10 2>&1 | tee mpfr-make.log
$ sudo mkdir -p /share/apps/mpfr/4.1.0
$ make install 2>&1 | tee mpfr-make-install.log
```

Module

```
##%Module1.0#####################################################################
↪##
##
## module load mpfr/4.1.0_gcc-8.5.0
##
##/share/apps/mpfr/4.1.0/gcc-8.5.0
## Written by Bryan Lopez Parra
##

proc ModulesHelp {} {
    global version modroot
    puts stderr "Sets the environment for using MPFR 4.1.0\
                \nin the shared directory /share/apps/mpfr/4.1.0/gcc-8.5.0\
                \nbuilded with gcc-8.5.0"
}

module-whatis "(Name_________) mpfr"
module-whatis "(Version______) 8.5.0"
module-whatis "(Compilers____) gcc-8.5.0"
module-whatis "(System_______) x86_64-redhat-linux"
module-whatis "(Libraries____) gmp-6.2.1"

# for Tcl script use only
set         topdir        /share/apps/mpfr/4.1.0/gcc-8.5.0
set         version       3.1.6
set         sys           x86_64-redhat-linux

conflict mpfr
module load gmp/6.2.1_gcc-8.5.0

prepend-path    LD_LIBRARY_PATH         $topdir/lib
prepend-path    LIBRARY_PATH            $topdir/lib
prepend-path    LD_RUN_PATH             $topdir/lib
```

(continues on next page)

(continued from previous page)

```
prepend-path    C_INCLUDE_PATH          $topdir/include
prepend-path    CXX_INCLUDE_PATH        $topdir/include
prepend-path    CPLUS_INCLUDE_PATH      $topdir/include

prepend-path    INFOPATH                $topdir/share/info
```

Use

```
module load mpfr/4.1.0_gcc-8.5.0
```

Resources

- https://www.mpfr.org/mpfr-current

Autor

- Bryan López Parra <blopezp@eafit.edu.co>

Resources

3.8.18 Szip

Szip[1] is an implementation of the extended-Rice lossless compression algorithm. The Consultative Committee on Space Data Systems (CCSDS) has adopted the extended-Rice algorithm for international standards for space applications[1,6,7]. Szip is reported to provide fast and effective compression, specifically for the EOS data generated by the NASA Earth Observatory System (EOS)[1]. It was originally developed at University of New Mexico (UNM) and integrated with HDF4 by UNM researchers and developers.

Szip 2.1.1

Table of Contents

[1] https://support.hdfgroup.org/doc_resource/SZIP/

Basic information

- **Instalation date:** 01/02/2018
- **Official Website:** https://www.zlib.net/
- **Supercomputer:** Cronos
- **License:** Non commercial Use **[SEARCH]**

Installation

1. Load the necessary modules for compilation

```
module purge
module load gcc/5.5.0
```

2. Download the desired version of the software (Source code - tar.gz)[1]

```
$ cd /home/mgomezzul/apps/szip/src/
$ wget https://support.hdfgroup.org/ftp/lib-external/szip/2.1.1/src/szip-
↪2.1.1.tar.gz
$ tar -xvf szip-2.1.1.tar.gz
```

3. After unzipping SZIP, continue with the following steps for configuration and compilation:

```
$ cd szip-2.1.1

$ ./configure --prefix=/share/apps/szip/2.1.1/gcc-5.5.0 --build=x86_64-
↪redhat-linux

$ make -j 10 2>&1 | tee szip-make.log
$ make check 2>&1 | tee szip-make-check.log
$ sudo mkdir -p /share/apps/szip/2.1.1/gcc-5.5.0
$ sudo chown -R mgomezzul.apolo /share/apps/szip/2.1.1/gcc-5.5.0
$ make install 2>&1 | tee szip-make-install.log
$ sudo chown -R root.root /share/apps/szip/2.1.1/gcc-5.5.0
```

Module

```
##%Module1.0#####################################################################
↪##
##
## module load szip/2.1.1_gcc-5.5.0
##
## /share/apps/modules/szip/2.1.1_gcc-5.5.0
## Written by Mateo Gómez-Zuluaga
##

proc ModulesHelp {} {
    global version modroot
    puts stderr "Sets the environment for using Szip-2.1.1\
                 \nin the shared directory /share/apps/szip/2.1.1/gcc-5.5.0\
```

(continues on next page)

[1] https://support.hdfgroup.org/ftp/lib-external/szip/2.1.1/src/szip-2.1.1.tar.gz

(continued from previous page)

```
                \nbuilded with gcc-5.5.0."
}

module-whatis "(Name________) szip"
module-whatis "(Version_____) 2.1.1"
module-whatis "(Compilers___) gcc-5.5.0"
module-whatis "(System______) x86_64-redhat-linux"
module-whatis "(Libraries___) "

# for Tcl script use only
set         topdir        /share/apps/szip/2.1.1/gcc-5.5.0
set         version       2.1.1
set         sys           x86_64-redhat-linux

conflict szip
module load gcc/5.5.0

prepend-path    LD_LIBRARY_PATH         $topdir/lib
prepend-path    LIBRARY_PATH            $topdir/lib
prepend-path    LD_RUN_PATH             $topdir/lib

prepend-path    C_INCLUDE_PATH          $topdir/include
prepend-path    CXX_INCLUDE_PATH        $topdir/include
prepend-path    CPLUS_INCLUDE_PATH      $topdir/include
```

Mode of use

```
$ module load /share/apps/modules/szip/2.1.1_gcc-5.5.0
```

References

Szip 2.1.1 - Intel

Table of Contents

Basic information

- **Instalation date:** June 2020

- **Official Website:** https://www.zlib.net/
- **Supercomputer:** Apolo II
- **License:** Non commercial Use **[SEARCH]**

Installation

1. Load the necessary modules for compilation

```
module purge
module load intel/19.0.4
```

2. Download the desired version of the software (Source code - tar.gz)[1]

```
$ cd /home/salzatec1/apps/szip
$ wget https://support.hdfgroup.org/ftp/lib-external/szip/2.1.1/src/szip-
↪2.1.1.tar.gz
$ tar -xvf szip-2.1.1.tar.gz
```

3. After unzipping SZIP, continue with the following steps for configuration and compilation:

```
$ cd szip-2.1.1

$ ./configure --prefix=/share/apps/szip/2.1.1/intel-19.0.4 --build=x86_64-
↪redhat-linux

$ make -j 10 2>&1 | tee szip-make.log
$ make check 2>&1 | tee szip-make-check.log
$ sudo mkdir -p /share/apps/szip/2.1.1/intel-19.0.4
$ sudo make install 2>&1 | tee szip-make-install.log
```

Module

```
##%Module1.0#####################################################################
↪##
##
## module load szip/2.1.1_intel_19.0.4
##
## /share/apps/modules/szip/2.1.1_intel_19.0.4
## Written by Tomas David Navarro & Santiago Alzate Cardona
##

proc ModulesHelp {} {
     global version modroot
     puts stderr "Sets the environment for using szip 2.1.1\
                  \nin the shared directory \
                  \n/share/apps/szip/2.1.1/intel_19.0.4/\
                  \nbuilded with intel-19.0.4"
}

module-whatis "(Name________) szip"
module-whatis "(Version_____) 2.1.1"
```

(continues on next page)

[1] https://support.hdfgroup.org/ftp/lib-external/szip/2.1.1/src/szip-2.1.1.tar.gz

(continued from previous page)

```
module-whatis "(Compilers___) intel-19.0.4"
module-whatis "(System______) x86_64-redhat-linux"
module-whatis "(Libraries___) intel"

# for Tcl script use only
set         topdir        /share/apps/szip/2.1.1/intel_19.0.4
set         version       2.1.1
set         sys           x86_64-redhat-linux

conflict szip
module load intel/19.0.4

prepend-path    LD_LIBRARY_PATH         $topdir/lib
prepend-path    LIBRARY_PATH            $topdir/lib
prepend-path    LD_RUN_PATH             $topdir/lib

prepend-path    C_INCLUDE_PATH          $topdir/include
prepend-path    CXX_INCLUDE_PATH        $topdir/include
prepend-path    CPLUS_INCLUDE_PATH      $topdir/include
```

Mode of use

```
$ module load szip/2.1.1_intel_19.0.4
```

References

Author

- Tomas David Navarro
- Santiago Alzate Cardona

3.8.19 Zlib

ZLib[1] is designed to be a free, general-purpose, legally unencumbered – that is, not covered by any patents – lossless data-compression library for use on virtually any computer hardware and operating system. The zlib data format is itself portable across platforms. Unlike the LZW compression method used in Unix compress(1) and in the GIF image format, the compression method currently used in zlib essentially never expands the data.

Zlib 1.2.11

Table of Contents

[1] https://www.zlib.net/

Basic information

- **Instalation date:** 01/02/2018
- **Official Website:** https://www.zlib.net/
- **Supercomputer:** Cronos
- **License:** Zlib License

Installation

1. Load the necessary modules for compilation

```
$ module purge
$ module load gcc/5.5.0
```

2. Download the desired version of the software (Source code - tar.gz)[1]

```
$ cd /home/mgomezzul/apps/szip/src/
$ wget https://support.hdfgroup.org/ftp/lib-external/szip/2.1.1/src/szip-
↪2.1.1.tar.gz
$ tar -xvf szip-2.1.1.tar.gz
```

3. After unzipping **Zlib**, continue with the following steps for configuration and compilation:

```
$ cd zlib-1.2.11

$ ./configure --prefix=/share/apps/zlib/1.2.11/gcc-5.5.0

$ make -j 10 2>&1 | tee zlib-make.log
$ make check 2>&1 | tee zlib-make-check.log
$ sudo mkdir -p /share/apps/zlib/1.2.11/gcc-5.5.0
$ sudo chown -R mgomezzul.apolo /share/apps/zlib/1.2.11/gcc-5.5.0
$ make install 2>&1 | tee gmp-make-install.log
$ sudo chown -R root.root /share/apps/zlib/1.2.11/gcc-5.5.0
```

Module

```
#%Module1.0#####################################################################
↪##
##
## module load zlib/1.2.11_gcc-5.5.0
```

(continues on next page)

[1] https://www.zlib.net/

(continued from previous page)

```
##
## /share/apps/modules/zlib/1.2.11_gcc-5.5.0
## Written by Mateo Gomez Zuluaga
##

proc ModulesHelp {} {
    global version modroot
    puts stderr "Sets the environment for using zlib 1.2.11\
                \nin the shared directory /share/apps/zlib/1.2.11/gcc-5.5.0\
                \nbuilded with gcc-4.4.7"
}

module-whatis "(Name________) zlib"
module-whatis "(Version_____) 1.2.11"
module-whatis "(Compilers___) gcc-5.5.0"
module-whatis "(System______) x86_64-redhat-linux"
module-whatis "(Libraries___) "

# for Tcl script use only
set         topdir        /share/apps/zlib/1.2.11/gcc-5.5.0
set         version       1.2.11
set         sys           x86_64-redhat-linux

conflict zlib
module load gcc/5.5.0

prepend-path    LD_LIBRARY_PATH         $topdir/lib
prepend-path    LIBRARY_PATH            $topdir/lib
prepend-path    LD_RUN_PATH             $topdir/lib

prepend-path    C_INCLUDE_PATH          $topdir/include
prepend-path    CXX_INCLUDE_PATH        $topdir/include
prepend-path    CPLUS_INCLUDE_PATH      $topdir/include

prepend-path    PKG_CONFIG_PATH         $topdir/lib/pkgconfig

prepend-path    MANPATH                 $topdir/share/man
```

Mode of use

```
$ module load zlib/1.2.11_gcc-5.5.0
```

References

Author

Mateo Gómez Zuluaga

Zlib 1.2.11 - Intel

Table of Contents

Basic information

- **Instalation date:** June 2020
- **Official Website:** https://www.zlib.net/
- **Supercomputer:** Apolo II
- **License:** Zlib License

Installation

1. Load the necessary modules for compilation

```
$ module purge
$ module load intel/19.0.4
```

2. Download the desired version of the software (Source code - tar.gz)[1]

```
$ cd $HOME/apps/zlib
$ wget https://zlib.net/zlib-1.2.11.tar.gz
$ tar -xvf zlib-1.2.11.tar.gz
```

3. After unzipping **Zlib**, continue with the following steps for configuration and compilation:

```
$ cd zlib-1.2.11
$ ./configure --prefix=/share/apps/zlib/1.2.11/intel-19.0.4/ --archs="-
↪arch x86_64"
$ make -j 10 2>&1 | tee zlib-make.log
$ make check 2>&1 | tee zlib-make-check.log
$ sudo mkdir -p /share/apps/zlib/1.2.11/intel-19.0.4
$ sudo make install 2>&1 | tee gmp-make-install.log
```

Module

[1] https://www.zlib.net/

```
#%Module1.0#####################################################################
↪##
##
## module load zlib/1.2.11_intel_19.0.4
##
## /share/apps/modules/zlib/1.2.11_intel_19.0.4
## Written by Tomas David Navarro & Santiago Alzate Cardona
##

proc ModulesHelp {} {
     global version modroot
     puts stderr "Sets the environment for using zlib 1.2.11\
                  \nin the shared directory \
                  \n/share/apps/zlib/1.2.11/intel_19.0.4/\
                  \nbuilded with intel_19.0.4"
}

module-whatis "(Name________) zlib"
module-whatis "(Version_____) 1.2.11"
module-whatis "(Compilers___) intel-19.0.4"
module-whatis "(System______) x86_64-redhat-linux"
module-whatis "(Libraries___) intel"

# for Tcl script use only
set         topdir        /share/apps/zlib/1.2.11/intel_19.0.4
set         version       1.2.11
set         sys           x86_64-redhat-linux

conflict zlib
module load intel/19.0.4

prepend-path    LD_LIBRARY_PATH         $topdir/lib
prepend-path    LIBRARY_PATH            $topdir/lib
prepend-path    LD_RUN_PATH             $topdir/lib

prepend-path    C_INCLUDE_PATH          $topdir/include
prepend-path    CXX_INCLUDE_PATH        $topdir/include
prepend-path    CPLUS_INCLUDE_PATH      $topdir/include

prepend-path    PKG_CONFIG_PATH         $topdir/lib/pkgconfig

prepend-path    MANPATH                 $topdir/share/man
```

Mode of use

```
$ module load zlib/1.2.11_intel_19.0.4
```

References

Author

- Tomas David Navarro
- Santiago Alzate Cardona

Zlib 1.2.11-gcc

Table of Contents

Basic information

- **Instalation date:** 03/03/2022
- **Official Website:** https://www.zlib.net/
- **Supercomputer:** Apolo II
- **License:** Zlib License

Installation

1. Load the necessary modules for compilation

```
module load gcc/11.2.0
```

2. Download the desired version of the software (Source code - tar.gz)[1]

```
$ cd /home/blopezp
$ wget https://zlib.net/zlib-1.2.11.tar.gz
$ tar -zxvf zlib-1.2.11.tar.gz
```

3. After unzipping **Zlib**, continue with the following steps for configuration and compilation:

```
$ cd zlib-1.2.11

$ ./configure --prefix=/share/apps/zlib/1.2.11/gcc-11.2.0

$ make -j 10 2>&1 | tee zlib-make.log
$ make check 2>&1 | tee zlib-make-check.log
$ sudo mkdir -p /share/apps/zlib/1.2.11
$ make install 2>&1 | tee gmp-make-install.log
```

[1] https://www.zlib.net/

Module

```
##
## module load zlib/1.2.11_gcc-11.2.0
##
## /share/apps/zlib/1.2.11/gcc-11.2.0
## Written by Bryan Lopez Parra
##

proc ModulesHelp {} {
     global version modroot
     puts stderr "Sets the environment for using zlib 1.2.11\
                  \nin the shared directory \
                  \n/share/apps/zlib/1.2.11/gcc-11.2.0/\
                  \nbuilded with gcc 11.2.0"
}

module-whatis "(Name________) zlib"
module-whatis "(Version_____) 1.2.11"
module-whatis "(Compilers___) gcc-11.2.0"
module-whatis "(System______) x86_64-redhat-linux"
module-whatis "(Libraries___) gcc"

# for Tcl script use only
set         topdir        /share/apps/zlib/1.2.11/gcc-11.2.0
set         version       1.2.11
set         sys           x86_64-redhat-linux

conflict zlib
module load gcc/11.2.0

prepend-path    LD_LIBRARY_PATH         $topdir/lib
prepend-path    LIBRARY_PATH            $topdir/lib
prepend-path    LD_RUN_PATH             $topdir/lib

prepend-path    C_INCLUDE_PATH          $topdir/include
prepend-path    CXX_INCLUDE_PATH        $topdir/include
prepend-path    CPLUS_INCLUDE_PATH      $topdir/include

prepend-path    PKG_CONFIG_PATH         $topdir/lib/pkgconfig

prepend-path    MANPATH                 $topdir/share/man
```

Mode of use

```
$ module load zlib/1.2.11_gcc-11.2.0
```

References

Authors

- Bryan López Parra <blopez@eafit.edu.co>

3.8.20 ScaLAPACK

ScaLAPACK[1] is a library of high-performance linear algebra routines for parallel distributed memory machines. ScaLAPACK solves dense and banded linear systems, least squares problems, eigenvalue problems, and singular value problems.

ScaLAPACK 2.0.2

Table of Contents

Basic information

- **Instalation date:** 13/02/2018
- **Official Website:** http://www.netlib.org/scalapack/
- **Supercomputer:** Cronos
- **License:** Modified BSD License

Dependencies

- OpenBLAS 0.2.20 (Includes BLAS and openBLAS)
- BLACS (Included in ScaLAPACK)

Installation

1. Load the necessary modules for compilation

```
$ module purge
$ module load openblas/0.2.20_gcc-5.5.0
```

2. Download the desired version of the software (Source code - tar.gz)[1]

[1] http://www.netlib.org/scalapack/
[1] http://www.netlib.org/scalapack/scalapack.tgz

```
$ cd /home/mgomezzul/apps/mpc/src/
$ wget http://www.netlib.org/scalapack/scalapack.tgz
$ tar -xvf scalapack.tgz
```

3. After unzipping **ScaLAPACK**, continue with the following steps for configuration and compilation:

```
$ cd scalapack-2.0.2
$ cp SLmake.inc.example SLmake.inc
```

We edit the following variables to specify the BLAS and LAPACK paths, in our case both are included in the openBLAS directory:

```
BLASLIB       = -L/share/apps/openblas/0.2.20/gcc-5.5.0/lib -lopenblas
LAPACKLIB     =
LIBS          = $(LAPACKLIB) $(BLASLIB)
```

The file should look like this:

```
############################################################################
#
#  Program:         ScaLAPACK
#
#  Module:          SLmake.inc
#
#  Purpose:         Top-level Definitions
#
#  Creation date:   February 15, 2000
#
#  Modified:        October 13, 2011
#
#  Send bug reports, comments or suggestions to scalapack@cs.utk.edu
#
############################################################################
#
#  C preprocessor definitions:  set CDEFS to one of the following:
#
#     -DNoChange (fortran subprogram names are lower case without any suffix)
#     -DUpCase   (fortran subprogram names are upper case without any suffix)
#     -DAdd_     (fortran subprogram names are lower case with "_" appended)

CDEFS         = -DAdd_

#
#  The fortran and C compilers, loaders, and their flags
#

FC            = mpif90
CC            = mpicc
NOOPT         = -O0
FCFLAGS       = -O3
CCFLAGS       = -O3
FCLOADER      = $(FC)
CCLOADER      = $(CC)
FCLOADFLAGS   = $(FCFLAGS)
CCLOADFLAGS   = $(CCFLAGS)

#
```

(continues on next page)

(continued from previous page)

```
#  The archiver and the flag(s) to use when building archive (library)
#  Also the ranlib routine.  If your system has no ranlib, set RANLIB = echo
#

ARCH          = ar
ARCHFLAGS     = cr
RANLIB        = ranlib

#
#  The name of the ScaLAPACK library to be created
#

SCALAPACKLIB  = libscalapack.a

#
#  BLAS, LAPACK (and possibly other) libraries needed for linking test␣
↪programs
#

BLASLIB       = -L/share/apps/openblas/0.2.20/gcc-5.5.0/lib -lopenblas
LAPACKLIB     =
LIBS          = $(LAPACKLIB) $(BLASLIB)
```

4. We proceed to compile the library

```
make all 2>&1 | tee scalapack-make.log
```

5. Once the **libscalapack.a** library is generated in the current directory create the directory in which it will be located and copy it there

```
$ sudo mkdir -p /share/apps/scalapack/2.0.2/gcc-5.5.0/lib
$ sudo chown -R mgomezzul.apolo /share/apps/scalapack/2.0.2/gcc-5.5.0/lib
$ cp libscalapack.a /share/apps/scalapack/2.0.2/gcc-5.5.0/lib
$ sudo chown -R root.root /share/apps/scalapack/2.0.2/gcc-5.5.0/lib
```

Module

```
##%Module1.0#####################################################################
↪##
##
## module load scalapack/2.0.2_gcc-5.5.0
##
## /share/apps/modules/scalapack/2.0.2_gcc-5.5.0
## Written by Mateo Gómez-Zuluaga
##

proc ModulesHelp {} {
    global version modroot
    puts stderr "Sets the environment for using ScaLAPACK 2.0.2\
                \nin the shared directory /share/apps/scalapack/2.0.2/gcc-5.
↪5.0\
                \nbuilded with gcc-5.5.0, openmpi-1.10.7 and openBLAS-0.2.20.
↪"
}
```

(continues on next page)

(continued from previous page)

```
module-whatis "(Name________) scalapack"
module-whatis "(Version_____) 2.0.2"
module-whatis "(Compilers___) gcc-5.5.0_openmpi-1.10.7"
module-whatis "(System______) x86_64-redhat-linux"
module-whatis "(Libraries___) openBLAS-0.2.20"

# for Tcl script use only
set         topdir        /share/apps/scalapack/2.0.2/gcc-5.5.0
set         version       2.0.2
set         sys           x86_64-redhat-linux

conflict scalapack
module load openmpi/1.10.7_gcc-5.5.0
module load openblas/0.2.20_gcc-5.5.0

prepend-path    LD_LIBRARY_PATH         $topdir/lib
prepend-path    LIBRARY_PATH            $topdir/lib
prepend-path    LD_RUN_PATH             $topdir/lib
```

Mode of use

```
$ module load scalapack/2.0.2_gcc-5.5.0
```

References

3.8.21 Cloog

CLooG[1] is a free software and library to generate code for scanning Z-polyhedra. That is, it finds a code (e.g. in C, FORTRAN...) that reaches each integral point of one or more parameterized polyhedra. CLooG has been originally written to solve the code generation problem for optimizing compilers based on the polytope model. Nevertheless it is used now in various area e.g. to build control automata for high-level synthesis or to find the best polynomial approximation of a function. CLooG may help in any situation where scanning polyhedra matters. While the user has full control on generated code quality, CLooG is designed to avoid control overhead and to produce a very effective code.

Cloog 0.18.4

Table of Contents

[1] https://www.cloog.org/

Basic information

- **Instalation date:** 01/02/2018
- **Official Website:** https://www.cloog.org/
- **Supercomputer:** Cronos
- **License:** Free Software Foundation's GNU General Public License

Dependencies

- GMP >= 6.0.0

Installation

1.. Download the desired version of the software (Source code - tar.gz)[1]

```
$ cd /home/mgomezzul/apps/cloog/src/
$ wget https://www.bastoul.net/cloog/pages/download/cloog-0.18.4.tar.gz
$ tar -xvf cloog-0.18.4.tar.gz
```

2. After unzipping **Cloog**, continue with the following steps for configuration and compilation:

```
$cd cloog-0.18.4

$ ./configure --prefix=/share/apps/cloog/0.18.4/gcc-4.4.7-18 --build=x86_
↪64-redhat-linux-gnu -with-isl=bundled --with-gmp-prefix=/share/apps/gmp/
↪6.1.2/gcc-4.4.7-18 --with-osl=bundled --with-gnu-ld

$ make -j 10 2>&1 | tee cloog-make.log
$ sudo mkdir -p /share/apps/cloog/0.18.4/gcc-4.4.7-18
$ sudo chown -R mgomezzul.apolo /share/apps/cloog/0.18.4/gcc-4.4.7-18
$ make install 2>&1 | tee cloog-make-install.log
$ sudo chown -R root.root /share/apps/cloog/0.18.4/gcc-4.4.7-18
```

It is important to leave the flags that include isl and osl in the installation as they are dependencies of R

Module

```
#%Module1.0#####################################################################
↪##
##
## module load cloog/0.18.4_gcc-4.4.7-18
##
## /share/apps/modules/cloog/0.18.4_gcc-4.4.7-18
```

(continues on next page)

[1] https://www.cloog.org/

(continued from previous page)

```
## Written by Mateo Gomez-Zuluaga
##

proc ModulesHelp {} {
    global version modroot
    puts stderr "Sets the environment for using CLooG 0.18.4\
                \nin the shared directory /share/apps/cloog/0.18.4/gcc-4.4.7-
↪18\
                \nbuilded with gcc-4.4.7-18."
}

module-whatis "(Name________) cloog"
module-whatis "(Version_____) 0.18.4"
module-whatis "(Compilers___) gcc-4.4.7-18"
module-whatis "(System______) x86_64-redhat-linux"
module-whatis "(Libraries___) gmp-6.1.2"

# for Tcl script use only
set         topdir        /share/apps/cloog/0.18.4/gcc-4.4.7-18
set         version       0.18.4
set         sys           x86_64-redhat-linux

conflict cloog
module load gmp/6.1.2_gcc-4.4.7-18

prepend-path    PATH                    $topdir/bin

prepend-path    LD_LIBRARY_PATH         $topdir/lib
prepend-path    LIBRARY_PATH            $topdir/lib
prepend-path    LD_RUN_PATH             $topdir/lib

prepend-path    C_INCLUDE_PATH          $topdir/include
prepend-path    CXX_INCLUDE_PATH        $topdir/include
prepend-path    CPLUS_INCLUDE_PATH      $topdir/include

prepend-path    PKG_CONFIG_PATH         $topdir/lib/pkgconfig
```

Mode of use

```
$ module load /share/apps/modules/cloog/0.18.4_gcc-4.4.7-18
```

References

Cloog 0.20.0

Table of Contents

Basic information

- **Instalation date:** 03/03/2022
- **Official Website:** https://www.cloog.org/
- **Supercomputer:** Apolo II
- **License:** Free Software Foundation's GNU General Public License

Dependencies

- GMP >= 6.2.1
- Libtool >= 2.4.6

Installation

1.. Download the desired version of the software (Source code)[1]

```
$ cd /home/blopezp
$ git clone https://github.com/periscop/cloog.git
```

2. After unzipping **Cloog**, continue with the following steps for configuration and compilation:

```
$cd cloog-0.20.0


$ module load libtool/2.4.6_gcc-8.5.0
$ module load gmp/6.2.1_gcc-8.5.0

$ ./get_submodules.sh
$ ./autogen.sh
$ ./configure --prefix=/share/apps/cloog/0.20.0/gcc-8.5.0 --build=x86_64-
↪redhat-linux-gnu -with-isl=bundled --with-gmp-prefix=/share/apps/gmp/6.
↪2.1/gcc-8.5.0 --with-osl=bundled --with-gnu-ld
$ make -j 10 2>&1 | tee cloog-make.log
$ sudo mkdir -p /share/apps/cloog/0.20.0
$ sudo make -j 10 install 2>&1 | tee cloog-make-install.log
```

It is important to leave the flags that include isl and osl in the installation as they are dependencies of R

[1] https://github.com/periscop/cloog.git

Module

```
##%Module1.0#####################################################################
##
## module load cloog/0.18.4_gcc-4.4.7-18
##
## /share/apps/modules/cloog/0.18.4_gcc-4.4.7-18
## Written by Bryan Lopez Parra
##

proc ModulesHelp {} {
    global version modroot
    puts stderr "Sets the environment for using CLooG 0.18.4\
                \nin the shared directory /share/apps/cloog/0.18.4/gcc-4.4.7-18\
                \nbuilded with gcc-4.4.7-18."
}

module-whatis "(Name________) cloog"
module-whatis "(Version_____) 0.18.4"
module-whatis "(Compilers___) gcc-4.4.7-18"
module-whatis "(System______) x86_64-redhat-linux"
module-whatis "(Libraries___) gmp-6.1.2"

# for Tcl script use only
set         topdir        /share/apps/cloog/0.20.0/gcc-8.5.0
set         version       0.20.0
set         sys           x86_64-redhat-linux

conflict cloog
module load gmp/6.2.1_gcc-8.5.0
module load libtool/2.4.6_gcc-8.5.0

prepend-path    PATH                    $topdir/bin

prepend-path    LD_LIBRARY_PATH         $topdir/lib
prepend-path    LIBRARY_PATH            $topdir/lib
prepend-path    LD_RUN_PATH             $topdir/lib

prepend-path    C_INCLUDE_PATH          $topdir/include
prepend-path    CXX_INCLUDE_PATH        $topdir/include
prepend-path    CPLUS_INCLUDE_PATH      $topdir/include

prepend-path    PKG_CONFIG_PATH         $topdir/lib/pkgconfig
```

Mode of use

```
$ module load cloog/0.20.0_gcc-8.5.0
```

Author

- Bryan López Parra <blopezp@eafit.edu.co>

References

3.8.22 MPC

GNU MPC[1] is a C library for the arithmetic of complex numbers with arbitrarily high precision and correct rounding of the result. It extends the principles of the IEEE-754 standard for fixed precision real floating point numbers to complex numbers, providing well-defined semantics for every operation. At the same time, speed of operation at high precision is a major design goal.

MPC 1.0.3

Table of Contents

Basic information

- **Instalation date:** 01/02/2018
- **Official Website:** http://www.multiprecision.org/mpc/
- **Supercomputer:** Cronos
- **License:** Free Software Foundation's GNU General Public License

Prerequisites

- GMP 6.1.2
- MPFR 3.1.6

Installation

1. Download the desired version of the software (Source code - tar.gz)[1]

```
$ cd /home/mgomezzul/apps/mpc/src/
$ wget https://ftp.gnu.org/gnu/mpc/mpc-1.0.3.tar.gz
$ tar -xvf mpc-1.0.3.tar.gz
```

[1] http://www.multiprecision.org/mpc/
[1] https://www.ece.uvic.ca/~frodo/jasper/#download

2. After decompressing **MPC**, continue with the following steps for configuration and compilation:

```
$  cd mpc-1.0.3

$ ./configure --prefix=/share/apps/mpc/1.0.3/gcc-4.4.7-18 --build=x86_64-
↪redhat-linux-gnu --with-mpfr=/share/apps/mpfr/3.1.6/gcc-4.4.7-18 --with-
↪gmp=/share/apps/gmp/6.1.2/gcc-4.4.7-18 --with-gnu-ld

$ make -j 10 2>&1 | tee mpc-make.log
$ sudo mkdir -p /share/apps/mpc/1.0.3/gcc-4.4.7-18
$ sudo chown -R mgomezzul.apolo /share/apps/mpc/1.0.3/gcc-4.4.7-18
$ make install 2>&1 | tee mpc-make-install.log
$ sudo chown -R root.root /share/apps/mpc/1.0.3/gcc-4.4.7-18
```

Module

```
##%Module1.0#####################################################################
↪##
##
## module load mpc/1.0.3_gcc-4.4.7-18
##
## /share/apps/modules/mpc/1.0.3_gcc-4.4.7-18
## Written by Mateo Gomez-Zuluaga
##

proc ModulesHelp {} {
    global version modroot
    puts stderr "Sets the environment for using MPC 1.0.3\
                \nin the shared directory  /share/apps/mpc/1.0.3/gcc-4.4.7-
↪18\
                \nbuilded with gcc-4.4.7-18."
}

module-whatis "(Name________) mpc"
module-whatis "(Version_____) 1.0.3"
module-whatis "(Compilers___) gcc-4.4.7-18"
module-whatis "(System______) x86_64-redhat-linux"
module-whatis "(Libraries___) "

# for Tcl script use only
set         topdir        /share/apps/mpc/1.0.3/gcc-4.4.7-18
set         version       1.0.3
set         sys           x86_64-redhat-linux

conflict mpc
module load gmp/6.1.2_gcc-4.4.7-18
module load mpfr/3.1.6_gcc-4.4.7-18

prepend-path    LD_LIBRARY_PATH         $topdir/lib
prepend-path    LIBRARY_PATH            $topdir/lib
prepend-path    LD_RUN_PATH             $topdir/lib

prepend-path    C_INCLUDE_PATH          $topdir/include
prepend-path    CXX_INCLUDE_PATH        $topdir/include
prepend-path    CPLUS_INCLUDE_PATH      $topdir/include
```

(continues on next page)

(continued from previous page)

```
prepend-path    CPLUS_INCLUDE_PATH      $topdir/include
```

Mode of use

```
$ module load mpc/1.0.3_gcc-4.4.7-18
```

References

MPC 1.2.1

Table of Contents

Basic information

- **Instalation date:** 03/03/2022
- **Official Website:** http://www.multiprecision.org/mpc/
- **Supercomputer:** Apolo II
- **License:** Free Software Foundation's GNU General Public License

Prerequisites

- GMP 6.2.1
- MPFR 4.1.0

Installation

1. Download the desired version of the software (Source code - tar.gz)[1]

[1] http://www.multiprecision.org/mpc/

```
$ cd /home/blopezp
$ wget https://ftp.gnu.org/gnu/mpc/mpc-1.2.1.tar.gz
$ tar -zxvf mpc-1.2.1.tar.gz
```

2. After decompressing **MPC**, continue with the following steps for configuration and compilation:

```
$  cd mpc-1.2.1

$ ./configure --prefix=/share/apps/mpc/1.2.1/gcc-8.5.0 --build=x86_64-
↪redhat-linux-gnu --with-mpfr=/share/apps/mpfr/4.1.0/gcc-8.5.0 --with-
↪gmp=/share/apps/gmp/6.2.1/gcc-8.5.0 --with-gnu-ld

$ make -j 10 2>&1 | tee mpc-make.log
$ sudo mkdir -p /share/apps/mpc/1.2.1
$ make install 2>&1 | tee mpc-make-install.log
```

Module

```
##%Module1.0#####################################################################
↪##
##
## module load mpc/1.2.1_gcc.8.5.0
##
## /share/apps/mpc/1.2.1/gcc-8.5.0
## Written by Bryan Lopez Parra
##

proc ModulesHelp {} {
    global version modroot
    puts stderr "Sets the environment for using MPC 1.2.1\
                \nin the shared directory  /share/apps/mpc/1.2.1/gcc-8.5.0\
                \nbuilded with gcc-8.5.0"
}

module-whatis "(Name________) mpc"
module-whatis "(Version_____) 1.2.1"
module-whatis "(Compilers___) gcc-8.5.0"
module-whatis "(System______) x86_64-redhat-linux"
module-whatis "(Libraries___) "

# for Tcl script use only
set         topdir        /share/apps/mpc/1.2.1/gcc-8.5.0
set         version       1.2.1
set         sys           x86_64-redhat-linux

conflict mpc
module load gmp/6.2.1_gcc-8.5.0
module load mpfr/4.1.0_gcc-8.5.0

prepend-path    LD_LIBRARY_PATH         $topdir/lib
prepend-path    LIBRARY_PATH            $topdir/lib
prepend-path    LD_RUN_PATH             $topdir/lib

prepend-path    C_INCLUDE_PATH          $topdir/include
prepend-path    CXX_INCLUDE_PATH        $topdir/include
```

(continues on next page)

(continued from previous page)

```
prepend-path    CPLUS_INCLUDE_PATH      $topdir/include

prepend-path    CPLUS_INCLUDE_PATH      $topdir/include
```

Mode of use

```
$ module load mpc/1.2.1_gcc-8.5.0
```

Author

- Bryan López Parra <blopezp@eafit.edu.co>

References

3.8.23 MCT

MCT is a set of open-source software tools for creating coupled models.

MCT

Table of Contents

Basic information

- **License:** This product includes software developed by the University of Chicago, as Operator of Argonne National Laboratory. MCT license[1]
- **Apolo Version:** Indices

Requiriments

- GNU Fortran 11.2.0
- MPICH 4.0.1

[1] https://github.com/jcwarner-usgs/COAWST/blob/master/Lib/MCT/COPYRIGHT

Installation

1. clone the github repository in your home

```
$ git clone https://github.com/jcwarner-usgs/COAWST.git
```

2. move the MCT folder to the /share/apps/MCT folder, since the installation has to be done from root

```
$ cd COAWST/Lib
$ sudo cp -r MCT /share/apps/MCT/2.6.0/
```

3. Configure

```
$ ./configure --preffix=/share/apps/MCT/2.6.0/gcc-11.2.0
```

4. Enter to the Makefile.conf and modify the following lines

```
FCFLAGS = -fPIC -fopenmp -O3 -mavx2 -fallow-argument-mismatch
```

Note: The **-fallow-argument-mismatch** is necessary since there is a function that doesn't have the correct return, if you don't include the flag the compilation will fail always

```
...

REAL8 = -r8
...

ENDIAN = -convert big_endian
...

INCPATH = -I/share/apps/MCT/2.6.0/MCT/mpeu -I/share/apps/mpich/4.0.1/gcc-11.2.0/
↪include
...

MPILIBS = /share/apps/mpich/4.0.1/gcc-11.2.0/bin/mpif90
...
```

4. Make and make install

```
$ make
$ make install
```

Module

```
#%Module1.0#####################################################################
##
## module MCT/2.6.0_Intel_oneAPI-2022_update-1
##
## /share/apps/MCT/2.6.0/gcc-11.2.0     Written by Jacobo Monsalve Guzman
##

proc ModulesHelp { } {
  puts stderr "\tcurl/7.82.0 - sets the Environment for MCT in \
```

(continues on next page)

(continued from previous page)

```
  \n\tthe share directory /share/apps/MCT/2.6.0\n"
}

module-whatis "\n\n\tSets the environment for using MCT-2.6.0 \
             \n\tbuilded with gcc 11.2.0\n"

# for Tcl script use only
set       topdir     /share/apps/MCT/2.6.0/gcc-11.2.0
set       version    2.6.0
set       sys        x86_64-redhat-linux

conflict MCT

module load mpich/4.0.1_gcc-11.2.0 gcc/11.2.0

prepend-path LD_LIBRARY_PATH    $topdir/lib
prepend-path LIBRARY_PATH       $topdir/lib
prepend-path LD_RUN_PATH        $topdir/lib

prepend-path C_INCLUDE_PATH     $topdir/include
prepend-path CXX_INCLUDE_PATH   $topdir/include
prepend-path CPLUS_INCLUDE_PATH $topdir/include
```

References

3.8.24 LAPACK

Description

LAPACK is written in Fortran 90 and provides routines for solving systems of simultaneous linear equations, least-squares solutions of linear systems of equations, eigenvalue problems, and singular value problems. The associated matrix factorizations (LU, Cholesky, QR, SVD, Schur, generalized Schur) are also provided, as are related computations such as reordering of the Schur factorizations and estimating condition numbers. Dense and banded matrices are handled, but not general sparse matrices. In all areas, similar functionality is provided for real and complex matrices, in both single and double precision.

Versions

LAPACK 3.5.0

Table of Contents

Basic information

- **Official Website:** http://www.netlib.org/lapack/
- **License:** http://www.netlib.org/lapack/LICENSE.txt
- **Installed on:** Apolo II and Cronos
- **Installation date:** 14/04/2020

Tested on (Requirements)

- **OS base:** CentOS (x86_64) $\geq$ 6.6 (Rocks 6.2)
- **Dependencies:**
  - CMAKE (tested on V3.7.1)
  - GCC (tested on 5.4.0)

Installation

1. Download the 3.5.0 version of LAPACK

```
$ wget http://www.netlib.org/lapack/lapack-3.5.0.tgz
$ tar -xvf lapack-3.5.0.tgz
$ cd lapack-3.5.0
```

2. After unpacking NetCDF, continue with the following steps for configuration and compilation:

```
$ mkdir build
$ cd build
$ cmake .. -DBUILD_SHARED_LIBS=ON -DCMAKE_INSTALL_PREFIX=/share/apps/lapack/3.5.0/
↪gcc/5.4.0/
$ make -j <N>
$ make -j <N> install
```

Module

```
##%Module1.0#####################################################################
↪###
##
## modules lapack/3.5.0_gcc-5.4.0
##
## /share/apps/modules/lapack/3.5.0_gcc-5.4.0  Written by Tomás David␣
↪Navarro y Juan Diego Ocampo
##

proc ModulesHelp { } {
```

(continues on next page)

(continued from previous page)

```
    puts stderr "\tLapack/3.5.0 - sets the Environment for LAPACK 3.5.0 in \
                 \n\tThe share directory /share/apps/lapack/3.5.0/gcc/5.4.0\n
↪"
}

module-whatis "\n\n\tSets the environment for using LAPACK 3.5.0 (Linear \
               \n\tAlgebra Library) builded with gcc 5.4.0\n"

# for Tcl script use only
set   topdir     /share/apps/lapack/3.5.0/gcc/5.4.0
set   version    3.5.0
set   sys        x86_64-redhat-linux

module load gcc/5.4.0

prepend-path      LD_LIBRARY_PATH         $topdir/lib
prepend-path      LIBRARY_PATH            $topdir/lib
prepend-path      LD_RUN_PATH             $topdir/lib
setenv            LAPACK                  $topdir/lib/liblapack.a
```

Use

TO DO

Resources

- http://www.netlib.org/lapack/

Author

- Tomas David Navarro Munera

3.8.25 Libpng

The libpng[1] package contains libraries used by other programs for reading and writing PNG files. The PNG format was designed as a replacement for GIF and, to a lesser extent, TIFF, with many improvements and extensions and lack of patent problems.

Libpng-1.6.37

Table of Contents

[1] https://www.linuxfromscratch.org/blfs/view/svn/general/libpng.html

Basic information

- **Instalation date:** 03/03/2022
- **Official Website:** http://www.libpng.org/
- **Supercomputer:** Apolo II

Installation

1. Load the necessary modules for compilation

```
$ module load intel/2022_oneAPI-update1
```

2. Download the desired version of the software (Source code - tar.gz)[1]

```
$ cd /home/blopezp
$ wget https://sourceforge.net/projects/libpng/files/libpng16/1.6.37/
↪libpng-1.6.37.tar.xz
$ tar -zxvf libpng-1.6.37.tar.xz
```

3. After unzipping **Libpng**, continue with the following steps for configuration and compilation:

```
$ cd libpng-1.6.37

$ ./configure --prefix=/share/apps/libpng/1.6.37/Intel_oneAPI-2022_update-
↪1

$ make -j 10 2>&1 | tee zlib-make.log
$ make check 2>&1 | tee zlib-make-check.log
$ sudo mkdir -p /share/apps/libpng/1.6.37
$ sudo make install 2>&1 | tee gmp-make-install.log
```

Module

```
##%Module1.0#####################################################################
↪###
##
## modulefile /share/apps/libpng/1.6.37
##

proc ModulesHelp { } {
     global version modroot
          puts stderr "\t Libpng 1.6.37"
}

module-whatis "\n\n\tSets the environment for using Libpng 1.6.37 \n"
```

(continues on next page)

[1] http://www.libpng.org/

(continued from previous page)

```
set     topdir          /share/apps/libpng/1.6.37/Intel_oneAPI-2022_update-1
set     version         1.6.37
set     sys             x86_64-redhat-linux

prepend-path    PATH                    $topdir/bin

prepend-path    C_INCLUDE_PATH          $topdir/include
prepend-path    CXX_INCLUDE_PATH        $topdir/include
prepend-path    CPLUS_INCLUDE_PATH      $topdir/include

prepend-path    LD_LIBRARY_PATH         $topdir/lib
prepend-path    LIBRARY_PATH            $topdir/lib
prepend-path    LD_RUN_PATH             $topdir/lib

prepend-path    MANPATH                 $topdir/share

setenv          LIBPNG_HOME             $topdir
```

Mode of use

```
$ module load libpng/1.6.37_Intel_oneAPI-2022_update-1
```

References

Author

- Bryan López Parra <blopezp@eafit.edu.co>

3.8.26 Libtool

GNU Libtool[1] is a generic library support script that hides the complexity of using shared libraries behind a consistent, portable interface.

Libtool 2.4.6

Table of Contents

[1] https://www.gnu.org/software/libtool/

Basic information

- **Instalation date:** 03/03/2022
- **Official Website:** https://www.gnu.org/software/libtool/
- **Supercomputer:** Apolo II
- **License:** GNU License version 3.0

Installation

1. Download the desired version of the software (Source code - tar.gz)[1]

```
$ cd /home/blopezp
$ wget https://ftp.gnu.org/gnu/libtool/libtool-2.4.6.tar.gz
$ tar -zxvf libtool-2.4.6.tar.gz
```

2. After unzipping **Libtool**, continue with the following steps for configuration and compilation:

```
$ cd libtool-2.4.6

$ ./configure --prefix=/share/apps/libtool/2.4.6/gcc-8.5.0

$ make -j 10 2>&1 |& tee libtool-make.log
$ make check 2>&1 |& tee libtool-make-check.log
$ sudo mkdir -p /share/apps/libtool/2.4.6
$ make -j 10 install 2>&1 |& tee libtool-make-install.log
```

Module

```
##%Module1.0#####################################################################
↪###
##
## modules libtool/2.4.6_Intel_gcc-8.5.0
##
## /share/apps/modules/libtool/2.4.6  Written by Bryan Lopez Parra
###

proc ModulesHelp { } {
    puts stderr "\tLibtool/2.4.6 - sets the Environment for LIBTOOL 2.4.6 in␣
↪\
               \n\tThe share directory /share/apps/libtool/2.4.6/gcc-8.5.
↪0\n"
}

module-whatis "\n\n\tSets the environment for using LIBTOOL 2.4.6 \
             \n\tbuilded with Intel GCC 8.5.0\n"

# for Tcl script use only
set   topdir     /share/apps/libtool/2.4.6/gcc-8.5.0
set   version    2.4.6
set   sys        x86_64-redhat-linux
```

(continues on next page)

[1] https://www.gnu.org/software/libtool/

(continued from previous page)

```
prepend-path      PATH                    $topdir/bin
prepend-path      LD_LIBRARY_PATH         $topdir/lib
prepend-path      LIBRARY_PATH            $topdir/lib
prepend-path      LD_RUN_PATH             $topdir/lib
prepend-path      INCLUDE_PATH            $topdir/include
```

Mode of use

```
$ module load libtool/2.4.6_gcc-8.5.0
```

References

Author

- Bryan López Parra <blopezp@eafit.edu.co>

3.8.27 CUnit

Description

CUnit is a lightweight system for writing, administering, and running unit tests in C. It provides C programmers a basic testing functionality with a flexible variety of user interfaces.[1]

Versions

CUnit 2.1-3

Table of Contents

[1] http://cunit.sourceforge.net/

Basic information

- **Official Website:** http://cunit.sourceforge.net/
- **License:**
- **Installed on:** Apolo II
- **Installation date:** 10/06/2020

Tested on (Requirements)

- **OS base:** CentOS (x86_64) $\geq$ 6.6 (Rocks 6.2)
- **Dependencies:**
  - GCC (tested on 5.4.0)

Installation

1. Download the 2.1-3 version of CUnit

```
$ wget https://github.com/Linaro/libcunit/releases/download/2.1-3/CUnit-2.1-3.tar.
↪bz2
$ tar -xvf CUnit-2.1-3.tar.bz2
$ cd CUnit-2.1-3
```

2. After unpacking CUnit, continue with the following steps for configuration and compilation:

```
$ module load gcc/5.4.0
$ aclocal
$ autoconf
$ automake  # If this command fails go to Troubleshooting
$ ./configure --prefix=/share/apps/cunit/2.1-3/gcc-5.4.0/
$ make
$ make install
```

Troubleshooting

Automake will not work on Cent0S 6.2 or Rocks 6. Then, what you have to do is to use the autoreconf -i command and run automake again, this is because autoconf doesn't generate the files properly.

Module

```
#%Module1.0#####################################################################
↪##
##
## module load cunit/2.1-3_gcc-5.4.0
##
## /share/apps/modules/cunit/2.1-3_gcc-5.4.0
## Written by Laura Sanchez Córdoba
## Date: June 10, 2020
##
```

(continues on next page)

(continued from previous page)

```
proc ModulesHelp {} {
     global version modroot
     puts stderr "Sets the environment for using CUnit-2.1-3\
                  \nin the shared directory /share/apps/cunit/2.1-3/gcc-5.4.
↪0\
                  \nbuilt with gcc-5.4.0"
}

module-whatis "(Name________) cunit"
module-whatis "(Version_____) 2.1-3"
module-whatis "(Compilers___) gcc-5.4.0"
module-whatis "(System______) x86_64-redhat-linux"
module-whatis "(Libraries___) "

# for Tcl script use only
set         topdir        /share/apps/cunit/2.1-3/gcc-5.4.0
set         version       2.1-3
set         sys           x86_64-redhat-linux

conflict cunit
module load gcc/5.4.0

prepend-path    LD_LIBRARY_PATH         $topdir/lib
prepend-path    LIBRARY_PATH            $topdir/lib
prepend-path    LD_RUN_PATH             $topdir/lib

prepend-path    C_PATH                  $topdir/lib

prepend-path    C_INCLUDE_PATH          $topdir/include
prepend-path    CXX_INCLUDE_PATH        $topdir/include
prepend-path    CPLUS_INCLUDE_PATH      $topdir/include

prepend-path    PKG_CONFIG_PATH         $topdir/lib/pkgconfig

prepend-path    MANPATH                 $topdir/share/man

setenv            CUNIT_HOME              $topdir
```

Use

```
module load cunit/2.1-3_gcc-5.4.0
```

Resources

- CUnit README

Author

- Laura Sánchez Córdoba

Resources

3.8.28 udunits

Description

The UDUNITS package supports units of physical quantities. Its C library provides for arithmetic manipulation of units and for conversion of numeric values between compatible units. The package contains an extensive unit database, which is in XML format and user-extendable. The package also contains a command-line utility for investigating units and converting values.[1]

Versions

udunits 2.2.26

Table of Contents

Basic information

- **Official Website:** https://www.unidata.ucar.edu/software/udunits/
- **License:**
- **Installed on:** Apolo II
- **Installation date:** 10/06/2020

Tested on (Requirements)

- **OS base:** CentOS (x86_64) $\geq$ 6.6 (Rocks 6.2)
- **Dependencies:**
  - GCC (tested on 5.4.0)
  - CUnit (tested on 2.1-3)

[1] https://www.unidata.ucar.edu/software/udunits/

Installation

1. Download the 2.2.26 version of udunits

```
$ wget ftp://ftp.unidata.ucar.edu/pub/udunits/udunits-2.2.26.tar.gz
$ tar -xvf udunits-2.2.26.tar.gz
$ cd udunits-2.2.26
```

2. After unpacking udunits, continue with the following steps for configuration and compilation:

```
$ module load gcc/5.4.0
$ module load cunit/2.1-3_gcc-5.4.0
$ ./configure --prefix=/share/apps/cunit/2.2.26/gcc-5.4.0/
$ make
$ make check # optional, needs CUnit library
$ make install
$ make install-html install-pdf # optional
$ make clean
```

Module

```
##%Module1.0#####################################################################
↪##
##
## module load udunits/2.2.26_gcc-5.4.0
##
## /share/apps/modules/udunits/2.2.26_gcc-5.4.0
## Written by Laura Sanchez Córdoba
## Date: June 10, 2020
##

proc ModulesHelp {} {
    global version modroot
    puts stderr "Sets the environment for using udunits-2.2.26\
                \nin the shared directory /share/apps/udunits/2.2.26/gcc-5.4.
↪0\
                \nbuilt with gcc-5.4.0"
}

module-whatis "(Name_________) udunits"
module-whatis "(Version______) 2.2.26"
module-whatis "(Compilers____) gcc-5.4.0"
module-whatis "(System_______) x86_64-redhat-linux"
module-whatis "(Libraries____) cunit/2.1-3_gcc-5.4.0"

# for Tcl script use only
set         topdir        /share/apps/udunits/2.2.26/gcc-5.4.0
set         version       2.2.26
set         sys           x86_64-redhat-linux

conflict udunits
module load gcc/5.4.0
module load cunit/2.1-3_gcc-5.4.0

prepend-path    LD_LIBRARY_PATH         $topdir/lib
```

(continues on next page)

(continued from previous page)

```
prepend-path    LIBRARY_PATH            $topdir/lib
prepend-path    LD_RUN_PATH             $topdir/lib

prepend-path    C_PATH                  $topdir/lib

prepend-path    C_INCLUDE_PATH          $topdir/include
prepend-path    CXX_INCLUDE_PATH        $topdir/include
prepend-path    CPLUS_INCLUDE_PATH      $topdir/include

prepend-path      INFODIR                           $topdir/share/info
prepend-path      INFOPATH                          $topdir/share/info

setenv            UDUNITS_HOME                  $topdir
setenv            UDUNITS2_XML_PATH             $topdir/share/udunits/
↪udunits2.xml
```

Use

```
module load udunits/2.2.26_gcc-5.4.0
```

Resources

- https://www.unidata.ucar.edu/software/udunits/udunits-current/doc/udunits/udunits2.html

Author

- Laura Sánchez Córdoba

Resources

3.8.29 PCRE2

The PCRE library is a set of functions that implement regular expression pattern matching using the same syntax and semantics as Perl 5. PCRE has its own native API, as well as a set of wrapper functions that correspond to the POSIX regular expression API. The PCRE library is free, even for building proprietary software.

PCRE-2 10.39

Table of Contents

Basic information

- **Official Website:** https://pcre.org/
- **License:** https://pcre.org/licence.txt
- **Installed on:** Apolo II
- **Instalation date:** 14/03/2022

Prerequisites

- Intel oneAPI

Installation

1. Load the necessary modules for compilation

```
$ module load intel
```

2. Download the desired version of the software

```
$ wget https://github.com/PhilipHazel/pcre2/releases/download/pcre2-10.39/
↪pcre2-10.39.tar.gz
$ tar xzvf pcre2-10.39.tar.gz
```

3. Move to the decompressed directory

```
$ cd pcre2-10.39
```

4. Export the necessary variables and configure:

```
$ export CC=/share/apps/intel/oneAPI/2022_update-1/compiler/2022.0.2/
↪linux/bin/intel64/icc
$ export CFLAGS="-I/share/apps/intel/oneAPI/2022_update-1/compiler/2022.0.
↪2/linux/include -L/share/apps/intel/oneAPI/2022_update-1/compiler/2022.
↪0.2/linux/lib -O3 -xHost -ip"
$ export FC=/share/apps/intel/oneAPI/2022_update-1/compiler/2022.0.2/
↪linux/bin/intel64/ifort
$ export  FCFLAGS="-I/share/apps/intel/oneAPI/2022_update-1/compiler/2022.
↪0.2/linux/include -L/share/apps/intel/oneAPI/2022_update-1/compiler/
↪2022.0.2/linux/lib -O3 -xHost -ip"
$ export CXX=/share/apps/intel/oneAPI/2022_update-1/compiler/2022.0.2/
↪linux/bin/intel64/icc
$ export CXXFLAGS="-I/share/apps/intel/oneAPI/2022_update-1/compiler/2022.
↪0.2/linux/include -L/share/apps/intel/oneAPI/2022_update-1/compiler/
↪2022.0.2/linux/lib -O3 -xHost -ip"
$ ./configure --prefix=/share/apps/pcre2/10.39/Intel_oneAPI-2022_update-1/
```

5. Compile and install:

```
$ make -j 16
$ make -j 16 install
```

Module

```
#%Module1.0#####################################################################
↪###
##
## modules pcre2/10.39_Intel_oneAPI-2022_update-1
##
## /share/apps/modules/pcre2/10.39 Written by Jacobo Monsalve
###

proc ModulesHelp { } {
    puts stderr "\tpcre2/10.39 - sets the Environment for PCRE2 10.39 in \
                \n\tThe share directory /share/apps/pcre2/10.39/Intel_oneAPI-
↪2022_update-1\n"
}

module-whatis "\n\n\tSets the environment for using PCRE2 10.39\
            \nbuilded with Intel oneAPI 2022 update 1\n"

# for Tcl script use only
set   topdir     /share/apps/pcre2/10.39/Intel_oneAPI-2022_update-1
set   version    10.39
set   sys        x86_64-redhat-linux

module load intel/2022_oneAPI-update1
prepend-path      PATH          $topdir/bin
prepend-path      LD_LIBRARY_PATH         $topdir/lib64
prepend-path      LIBRARY_PATH            $topdir/lib64
prepend-path      LD_RUN_PATH             $topdir/lib64
prepend-path      MANPATH             $topdir/man
prepend-path     C_INCLUDE_PATH       $topdir/include
prepend-path     CXX_INCLUDE_PATH     $topdir/include
prepend-path     CPLUS_INCLUDE_PATH   $topdir/include
prepend-path    PKG_CONFIG_PATH         $topdir/lib64/pkgconfig
```

How to use

```
$ module load pcre2/10.39_Intel_oneAPI-2022_update-1
```

Authors

- Jacobo Monsalve Guzman

3.9 Accelerators

This section describes the configurations and features that allow you to use the cluster accelerators through their drivers or development packages.

3.9.1 CUDA Multi-Process Service (MPS)

CUDA MPS[1] is an alternative to the CUDA application programming interface (CUDA API). The MPS runtime architecture is designed to transparently enable the use of Hyper-Q to CUDA applications that require the use of multiple cooperative processes, usually MPI jobs, on NVIDIA graphics processing units (those based on the Kepler architecture). Hyper-Q allows CUDA kernels to be processed concurrently on the same GPU; thus benefiting performance when it is sub -used by a single process

CUDA-8.0

Table of Contents

Basic information

- **Instalation Date:** 16/06/2017
- **Official name:** CUDA Multi-Process Service (MPS)
- **Apolo version:** Apolo II

[1] nvidia, "MULTI-PROCESS SERVICE", Mayo del 2015. https://docs.nvidia.com/deploy/pdf/CUDA_Multi_Process_Service_Overview.pdf. Accedido por última vez el 11 de julio del 2017

Dependencies and Instalation

After compiling and building the driver for a compatible nvidia accelerator, it is required to install the toolkit for the CUDA parallel computing platform. This installation will create in the /usr/bin/ directory the executables **nvidia-cuda-mps-control** and **nvidia-cuda-mps-server**.

Both files must be present before enabling the service for use.

Activation

The activation of the service is performed by executing nvidia-cuda-mps-control, which uses the environment variables CUDA_VISIBLE_DEVICES, CUDA_MPS_PIPE_DIRECTORY and CUDA_MPS_LOG_DIRECTORY to determine the configuration to be used at runtime. The table found below explains in detail the meaning of each one:

Hostname	IP address
CUDA_VISIBLE_DEVICES	Is used to specify which GPU will be visible to a CUDA application. The variable can contain positive integer values (separated by commas) starting with the number zero (0).
CUDA_MPS_PIPE_DIRECTORY	The different MPS clients communicate through named pipes located by default in the /tmp/nvidia-mps directory, which can be modified by assigning a different path to this variable.
CUDA_MPS_LOG_DIRECTORY	The MPS control daemon uses the control.log file to record the status of the MPS servers, commands executed by a user as well as their output, and starting and stopping information for the daemon. The MPS server maintains the log server.log, where it stores data about its startup, shutdown and status of each client. Both are stored by default in the /var/log/nvidia-mps directory, which can be modified by assigning a different path to this variable.

Daemon initiation and single server

By using the /var/log/nvidia-mps directory to store the activity logs, both the daemon and the MPS server restrict the use of the service to the root user, since the permissions on the /var path make it impossible for other users access its content. In order to start the daemon and the server without requiring the intervention of a superuser, it is necessary to use the environment variables described above as illustrated in the following example:

```
$ mkdir /tmp/mps_0
$ mkdir /tmp/mps_log_0
$ export CUDA_VISIBLE_DEVICES=0
$ export  CUDA_MPS_PIPE_DIRECTORY=/tmp/mps_0
$ export  CUDA_MPS_LOG_DIRECTORY=/tmp/mps_log_0
$ nvidia-cuda-mps-control -d
```

The use of CUDA_MPS_PIPE_DIRECTORY is optional, since the default path is accessible to all users of the system; while CUDA_MPS_LOG_DIRECTORY must contain a path in which the user seeking to start the service has read and write permissions

Initiation of multiple daemons and multiple servers

When having multiple nvidia accelerators, a daemon and an MPS server must be activated for each one. With this, and assuming that we have two accelerators, the structure described in the example in the section "Daemon initiation and a single server" must be transformed as follows:

```
NGPUS=2

for (( i = 0; i < $NGPUS; ++i  ))
do
    mkdir /tmp/mps_$i
    mkdir /tmp/mps_log_$i

    export CUDA_VISIBLE_DEVICES=$i
    export  CUDA_MPS_PIPE_DIRECTORY="/tmp/mps_$i"
    export  CUDA_MPS_LOG_DIRECTORY="/tmp/mps_$i"

    nvidia-cuda-mps-control -d
done
```

- It is necessary to clarify that this procedure is functional if and only if the cards are in the same node, where it is feasible for the nvidia driver to assign an id to each accelerator.
- Certain GPUs, such as the Tesla K80, internally contain two different graphics processing cards, each one being recognized as independent by the controller. In these controllers the procedure described in this section must be used to start the MPS service.

Detention

Stopping the daemon and a single server

The stopping process requires that the environment variables used by the MPS daemon during service initialization retain their value. Returning to the example of the subsection "Initiating the daemon and a single server", stopping the service should be carried out as follows:

```
$ export CUDA_VISIBLE_DEVICES=0
$ export  CUDA_MPS_PIPE_DIRECTORY=/tmp/mps_0
$ export  CUDA_MPS_LOG_DIRECTORY=/tmp/mps_log_0
$ echo "quit" | nvidia-cuda-mps-control
```

Stopping multiple daemons and multiple servers

Taking the explanation from the section "Starting multiple daemons and multiple servers", it is feasible to conclude that when starting a server for each card present it is necessary to stop them individually, as well as their respective daemons, using nvidia-cuda-mps-control. Taking the example from the section "Stopping the daemon and a single server" and assuming that you have two cards, the basic structure of a script to stop the MPS service would be:

```
NGPUS=2

for (( i = 0; i < $NGPUS; ++i ))
do
    export CUDA_MPS_PIPE_DIRECTORY="/tmp/mps_$i"
    echo "quit" | nvidia-cuda-mps-control
```

(continues on next page)

(continued from previous page)

```
    rm -fr /tmp/mps_$i
    rm -fr /tmp/mps_log_$i
done
```

How to Use

Once the service is activated, it can be used by two types of applications:

1. Those whose dependence on CUDA is not linked to MPI.
2. Those that run using MPI

In the former, it is enough to define the card on which the program will be executed through the assignment of the variable CUDA_VISIBLE_DEVICES, while the latter require the creation of a script that uses the variable CUDA_MPS_PIPE_DIRECTORY to indicate to the service which MPI process it will carry out. your instructions based on which server. The structure of this script is explained below:

- Initially, it is necessary to obtain the identifier of each MPI process, so that all can be distributed equally on the activated MPS servers; This identifier is known as rank according to the MPI standard and is stored in an environment variable, whose name may vary depending on the MPI implementation used: openmpi, mpich, mvapich, etc. The example illustrated below makes use of the variable OMPI_COMM_WORLD_LOCAL_RANK belonging to openmpi:

```
lrank="$OMPI_COMM_WORLD_LOCAL_RANK"
```

- After obtaining the identifier of each process, it is necessary to define the value of the variable CUDA_MPS_PIPE_DIRECTORY, ensuring that the distribution carried out is the same on each MPS server. The example illustrated below assumes that the maximum number of processes to use is four (4), the system has two cards and the directories where the named pipes used by each MPS server are located correspond to those described throughout the section Activation:

```
 case ${lrank} in
    [0])
        export CUDA_MPS_PIPE_DIRECTORY=/tmp/mps_0
        ;;
    [1])
        export CUDA_MPS_PIPE_DIRECTORY=/tmp/mps_1
        ;;
    [2])
        export CUDA_MPS_PIPE_DIRECTORY=/tmp/mps_0
        ;;
    [3])
        export CUDA_MPS_PIPE_DIRECTORY=/tmp/mps_1
        ;;
esac
```

- Finally, the program from which each of the MPI processes will be created is executed.

An example of the correct way of running a program that uses MPI and benefits from the MPS service is illustrated below.

```
$ cat mps_runnable.sh

    #!/bin/bash
```

(continues on next page)

(continued from previous page)

```
    lrank="$OMPI_COMM_WORLD_LOCAL_RANK"

    case ${lrank} in
        [0])
            export CUDA_MPS_PIPE_DIRECTORY=/tmp/mps_0
            ;;
        [1])
            export CUDA_MPS_PIPE_DIRECTORY=/tmp/mps_1
        ;;
        [2])
            export CUDA_MPS_PIPE_DIRECTORY=/tmp/mps_0
            ;;
        [3])
            export CUDA_MPS_PIPE_DIRECTORY=/tmp/mps_1
            ;;
    esac

    program_that_uses_gpu

$ mpirun -np 4 ./mps_runnable.sh
```

Troubleshooting

MPS cannot reserve memory at startup

"This error (ERROR_OUT_OF_MEMORY) is associated with the reservation of virtual memory over the address range of the UVA (Unified Virtual Addressing) by the processes that try to run on the MPS server on the GPU. This error normally occurs in old operating systems, this being the case of CentOS 6.6 (Rocks 6.2), since the program in charge of making said reservation (prelink) causes interference problems between dynamic libraries and virtual memory reservation in the UVA address range. The recommendation in this case is to temporarily or permanently disable the prelink" (Taken from[1])

Process

1. Enter the server where you want to run the MPS service with the root user.
2. Edit the **/etc/sysconfig/prelink** file and change **PRELINKING=yes** to **PRELINKING=no**
3. Manually run the following cron: **/etc/cron.daily/prelink**

In this way the **prelink** will be permanently disabled.

MPS processes change their status to "Zombie" after stopping the service

Although it is rare for this problem to occur, one of the ways to identify it is to verify the existence of a process named nvidia-cuda-mps whose status is shown in Z.

In case of performing the process of starting and stopping the MPS service in an automated way (i.e. through a script and not directly in an interactive session), it is advisable to detect the existence of one or more Zombie processes as shown below:

[1] nvidia, "MULTI-PROCESS SERVICE", Mayo del 2015, https://docs.nvidia.com/deploy/pdf/CUDA_Multi_Process_Service_Overview.pdf. Last accessed July 11, 2017

```
ps aux | grep nvidia-cuda-mps | grep -v grep > /dev/null
if [ [ $? -eq 0 ] ]; then
    declare -r error_message="Some error message"
    logger $error_message
    # Do something like "pkill -u <user owning the zombies>"
fi
```

GPU has problems after disabling MPS

It is a rare error, the causes of which are very varied. The best way to detect it if you have an automated stopping process for the MPS service is:

```
nvidia-smi > /dev/null
if [ [ $? -ne 0 ] ]; then
    declare -r error_message="Some error message"
    logger $error_message
    # Do something
fi
```

Links

- http://on-demand.gputechconf.com/gtc/2015/presentation/S5584-Priyanka-Sah.pdf
- http://cudamusing.blogspot.com.co/2013/07/enabling-cuda-multi-process-service-mps.html
- https://docs.nvidia.com/deploy/pdf/CUDA_Multi_Process_Service_Overview.pdf
- http://www.builddesigncreate.com/index.cgi?mode=page_details&pageid=2011080413332724848

References

Authors

- Tomás Felipe Llano Ríos
- Mateo Gómez-Zuluaga

CHAPTER 4

How to Acknowledge

Maintaining a Supercomputing center is very expensive, the Scientific Computing Center requests to our users to acknowledge the use of the resources in scientific production such as papers, books, patents or other publications of research that benefited from the utilize of the computational and staff resources. This acknowledgements help us to acquire funding and get more capacities that will fulfill the needs of our users.

We suggest the following paragraph to acknowledge the scientific computing center:

Note: The authors acknowledge supercomputing resources made available by the Centro de Computación Científica Apolo at Universidad EAFIT (http://www.eafit.edu.co/apolo) to conduct the research reported in this scientific product.

In addition, if applies, please also acknowledge the grant related with your research project:

Note: The computational resources used in this product was funded by the {Funding Institution} with grant {Grant Code ID}.

We would also appreciate to inform us about your publications in which you acknowledge one of the clusters.

CHAPTER 5

Report a Bug

Please report any issue related to the Apolo Scientific Computing Center in our bug tracker.

Bibliography

[Kura14] Kuratomi, Toshio: [ansible/el6] Fix ansible-vault for newer python-crypto dependency. fedoraproject.org, March 14 2014. Retrieved September 13, 2018 from https://lists.fedoraproject.org/pipermail/scm-commits/Week-of-Mon-20140310/1207203.html