Hybrid Parallelization of the MrBayes & RAxML Phylogenetics Codes

Wayne Pfeiffer (SDSC/UCSD) &
Alexandros Stamatakis (TUM)
February 25, 2010

What was done? Why is it important? Who cares?

- **Hybrid MPI/OpenMP version of MrBayes was developed**
  - **OpenMP code was added** to previous MPI-only code
- **Hybrid MPI/Pthreads version of RAxML was developed**
  - **MPI code was added** to previous Pthreads-only code
- **These enhancements allow multiple multi-core nodes in a cluster to be used in a single run**
  - Typical problems now run well on 4 to 10 nodes **(32 to 80 cores)** of Abe & Dash as compared to only on one node (8 cores) before
- **Hybrid, multi-grained codes are available on TeraGrid via CIPRES portal**
  - Work was done as part of ASTA project supporting Mark Miller of SDSC, who oversees the portal
  - Number of cores (processes * threads) is selected automatically
  - Portal simplifies use of the codes by typical biologists

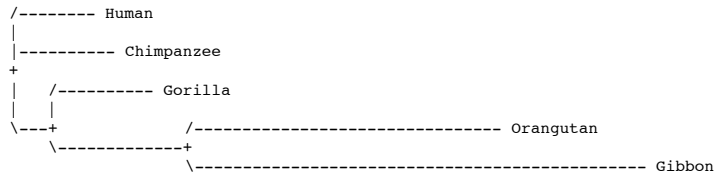
SDSC SAN DIEGO SUPERCOMPUTER CENTER
at the UNIVERSITY OF CALIFORNIA, SAN DIEGO
UCSD

What does a phylogenetics code do?

It starts from a
multiple sequence alignment (matrix of taxa versus characters)

```
                          .   .   .   ......  .   .
Human           AAGCTTCACCGGCGCAGTCATTCTCATAAT...
Chimpanzee      AAGCTTCACCGGCGCAATTATCCTCATAAT...
Gorilla         AAGCTTCACCGGCGCAGTTGTTCTTATAAT...
Orangutan       AAGCTTCACCGGCGCAACCACCCTCATGAT...
Gibbon          AAGCTTTACAGGTGCAACCGTCCTCATAAT...
```

& generates a phylogeny (usually a tree with taxa at the tips)



SDSC SAN DIEGO SUPERCOMPUTER CENTER
at the UNIVERSITY OF CALIFORNIA, SAN DIEGO
UCSD

A little more about molecular phylogenetics

- **Multiple sequence alignment**
  - Specified by DNA bases, RNA bases, or amino acids
  - Obtained by separate analysis
- **Possible changes to a sequence**
  - **Substitution** (point mutation or SNP: treated in MrBayes & RAxML)
  - **Insertion & deletion** (also handled by MrBayes & RAxML)
  - Structural variations (e.g., duplication, inversion, & translocation)
  - Recombination & horizontal gene transfer (important in bacteria)
- **Common methods, typically heuristic & based on models of molecular evolution**
  - Distance
  - Parsimony
  - **Maximum likelihood** (used by RAxML)
  - **Bayesian** (used by MrBayes)

SDSC SAN DIEGO SUPERCOMPUTER CENTER
at the UNIVERSITY OF CALIFORNIA, SAN DIEGO
UCSD

Key similarities between MrBayes & RAxML

- **Both compute a likelihood score that depends upon**
  - **Tree topology**
  - **Branch lengths**
  - **Parameters for model** of molecular evolution, which may be **partitioned**, i.e., vary between genes in multi-gene alignments

- **Both are programmed in C**

Key differences between MrBayes & RAxML

- **MrBayes**
  - Assumes **prior probabilities** for statistical parameters (consistent with Bayesian approach)
  - Optimizes tree topology, branch lengths, and model parameters using Metropolis-Coupled Markov-Chain Monte-Carlo approach or $(MC)^3$
  - Obtains statistical support by sampling results during stationary phase
- **RAxML**
  - Optimizes tree topology using a variant of subtree pruning and regrafting (SPR) called lazy subtree rearrangement (LSR)
  - Optimizes branch lengths using Newton-Raphson method
  - Optimizes model parameters using Brent's algorithm
  - Obtains statistical support from separate bootstrap searches
  - Generally runs **faster**

MrBayes has parallelism at multiple algorithmic levels

- **A typical analysis employs 2 "runs" with 4 chains each**
  - Each run starts from a different initial tree
  - The chains correspond to different amounts of heating in the Metropolis coupling
- **(MC)^3 has coarse-grained parallelism across 8 run-chain instances that can be exploited using MPI (in v 3.1.2)**
- **Computation of likelihood score can exploit fine-grained parallelism across patterns (i.e., distinct columns in alignment) using OpenMP (in new, hybrid code: v 3.1.2h)**

5 benchmark data sets & 4 benchmark computers were considered

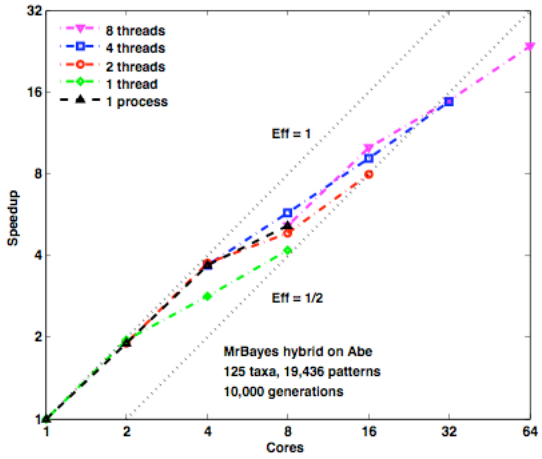
Benchmark data sets (all DNA or RNA)

Taxa	Characters	Patterns	Recommended bootstraps
354	460	348	1,200
150	1,269	1,130	650
218	2,294	1,846	550
404	13,158	7,429	700
125	29,149	19,436	50

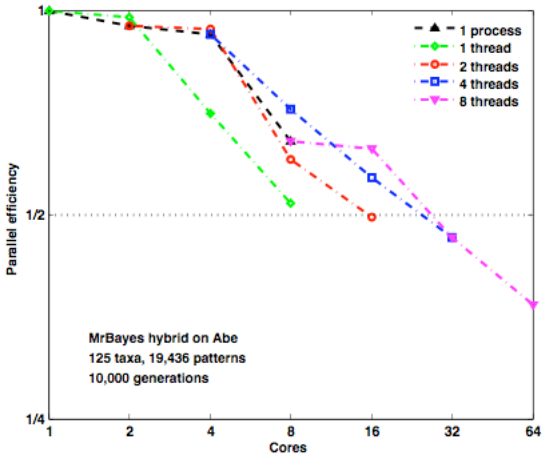
Benchmark computers (all with quad-core x64 processors)

Abe at NCSA	8-core nodes with 2.33-GHz Intel Clovertowns
Dash at SDSC	8-core nodes with 2.4-GHz Intel Nehalems
Ranger at TACC	16-core nodes with 2.3-GHz AMD Barcelonas
Triton PDAF at SDSC	32-core nodes with 2.5-GHz AMD Shanghais

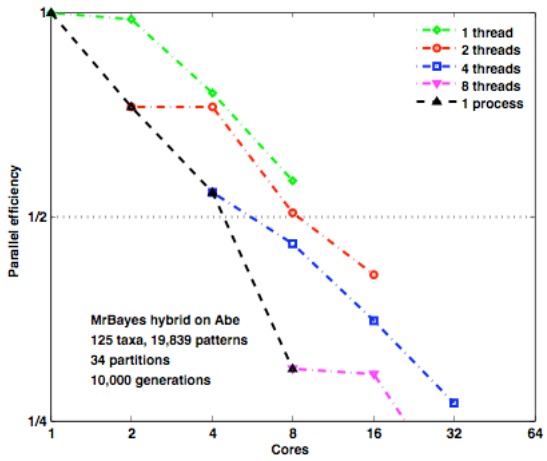
For problem with 19k patterns, MrBayes achieves speedup of 23 on 64 cores of Abe using 8 MPI processes with 8 threads each; speedup is 5.6 compared to MPI-only code on 8 cores



Parallel efficiency plot of same data clarifies performance differences; on > 8 cores, using 8 threads is optimal; on 8 cores, using 2 MPI processes & 4 threads is 1.4x faster than 8 MPI processes

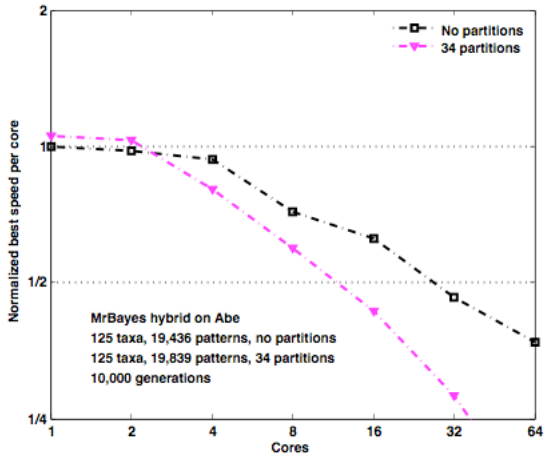


Scaling for same problem separated into 34 partitions is much worse; this is because load balance is poor with OpenMP; on 8 cores, using 8 MPI processes is fastest



SDSC SAN DIEGO SUPERCOMPUTER CENTER
at the UNIVERSITY OF CALIFORNIA, SAN DIEGO
UCSD

Comparing best speeds per core at each core count clearly shows that runs with partitions are appreciably slower for 8 or more cores



SDSC SAN DIEGO SUPERCOMPUTER CENTER
at the UNIVERSITY OF CALIFORNIA, SAN DIEGO
UCSD

RAxML has parallelism at multiple algorithmic levels

- **Computation of likelihood score within a tree can exploit fine-grained parallelism across patterns using Pthreads (in v 7.0.0)**
- **Three types of searches can exploit coarse-grained parallelism across trees using MPI (in new, hybrid code: v 7.2.4 and later)**
  - **Multiple ML searches** on the same data set starting from different initial trees to explore solution space better
  - **Multiple bootstrap searches** on resampled data sets to obtain confidence values on interior branches of tree (i.e., statistical support)
  - **Comprehensive analysis** that combines the two previous analyses to give a complete, publishable analysis in a single run

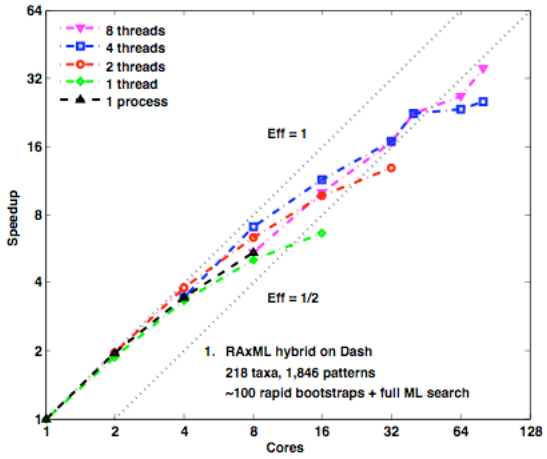
Comprehensive analysis combines many rapid bootstraps followed by a full ML search

- **Four stages (with typical numbers of searches)**
  - 100 rapid bootstrap searches
  - 20 fast ML searches
  - 10 slow ML searches
  - 1 thorough ML search
- **Coarse-grained parallelism via MPI**
  - In first three stages, but decreasing with each stage
- **Fine-grained parallelism via Pthreads**
  - Available at all stages
- **Tradeoff in effectiveness between MPI and Pthreads**
  - Typically 10 or 20 MPI processes max
  - Optimal number of Pthreads increasing with number of patterns, but limited to number of cores per node

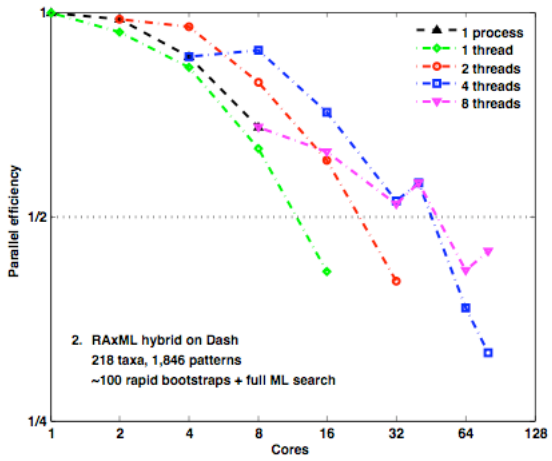
Some noteworthy points regarding the MPI parallel implementation

- **A thorough search is done for every MPI process (instead of just a single search as in Pthreads-only code)**
  - Increases run time only a little, because load is reasonably balanced
  - Often leads to better quality solution, so extra work is **useful**
- **Only two significant MPI calls are made**
  - MPI_Barrier after bootstraps
  - MPI_Bcast after thorough searches to select best one for output
  - Much simpler than older MPI-only implementation that used master/worker approach and more efficient given reasonable load balance
- **Treatment of random numbers is reproducible**
  - At least for a given number of MPI processes

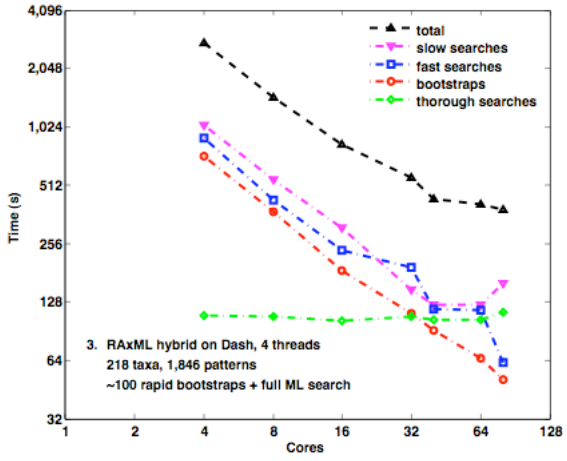
For problem with 1.8k patterns, RAxML achieves speedup of 35 on 80 cores of Dash using 10 MPI processes with 8 threads each; speedup is 6.5 compared to Pthreads-only code on 8 cores



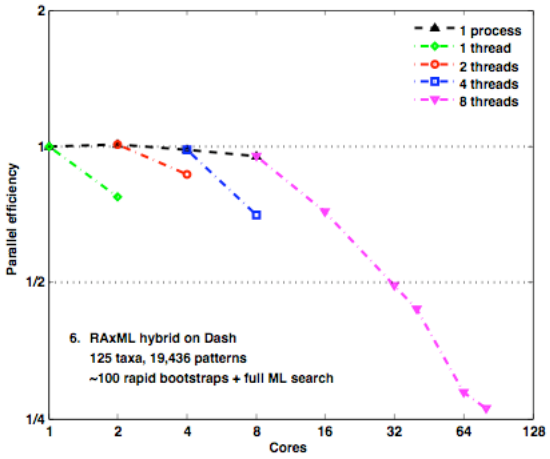
Parallel efficiency plot of same data clarifies performance differences; on ≥ 8 cores, using 4 or 8 threads is optimal; on 8 cores, using 4 threads is 1.3x faster than 8 threads



Bootstraps & fast searches scale well with MPI; slow & thorough searches limit scalability

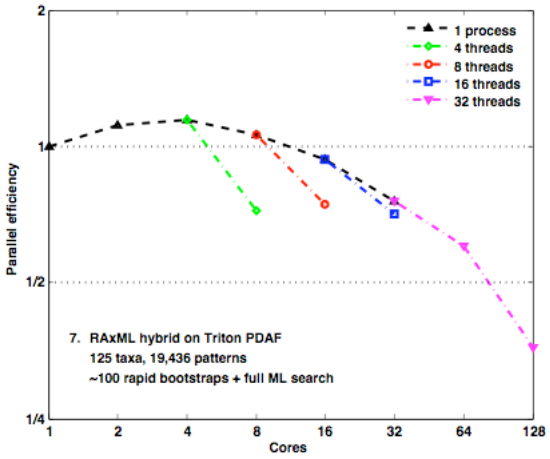


For problem with 19k patterns, performance on Dash is best using all 8 threads; scaling is poorer than for problem with 1.8k patterns



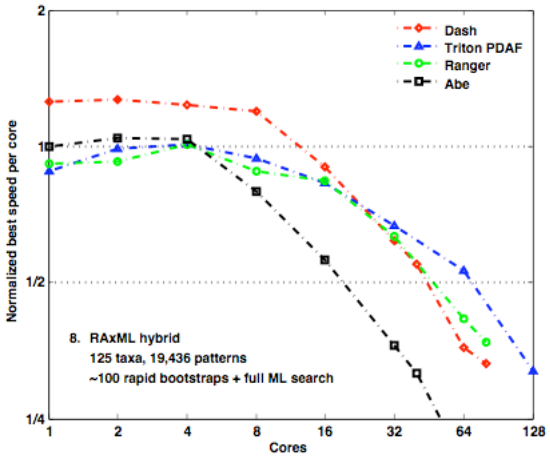
SDSC SAN DIEGO SUPERCOMPUTER CENTER
at the UNIVERSITY OF CALIFORNIA, SAN DIEGO
UCSD

For problem with 19k patterns, scaling is better on Triton PDAF than on Dash using all 32 threads available



SDSC SAN DIEGO SUPERCOMPUTER CENTER
at the UNIVERSITY OF CALIFORNIA, SAN DIEGO
UCSD

For problem with 19k patterns, Triton PDAF is faster than other computers at high core counts



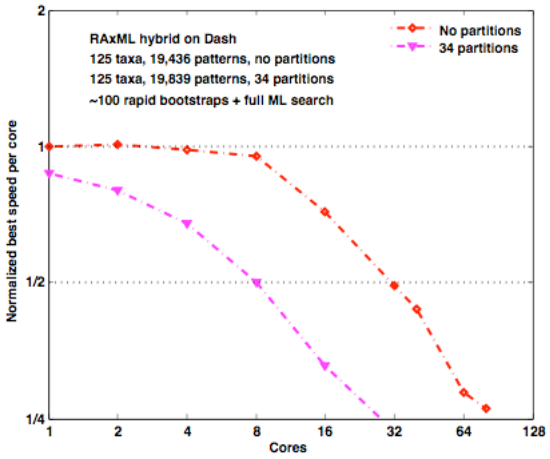
With more bootstraps (per bootstopping): speedup is better & parallel efficiency ≥ 0.44 on 80 cores; optimal number of threads drops; run time for longest problem goes from > 4 d to < 1.8 h

Taxa	Patterns	Best time (s) / threads on 1c	on 80c	Speedup on 80c	Computer/ bootstraps
Results for 100 bootstraps specified					
354	348	1,980	130 /4	15.23	Dash
150	1,130	2,325	95 /8	24.47	Dash
218	1,846	9,630	271 /8	35.54	Dash
404	7,429	72,866	1,828 /8	39.86	Dash
125	19,436	22,970	1,092 /8	21.03	Dash
125	19,436	32,627	847 /32 +	38.52 +	Triton PDAF
Results for >100 bootstraps specified					
354	348	15,703	443 /2	**35.45**	Dash/1,200
150	1,130	10,566	290 /4	**36.43**	Dash/ 650
218	1,846	33,738	845 /4	**39.93**	Dash/ 550
404	7,429	**355,724**	**6,270** /8	**56.73**	Dash/ 700

1c and 80c refer to the number of cores used in a run.
The run with a + was made on 64c.

SDSC SAN DIEGO SUPERCOMPUTER CENTER
at the UNIVERSITY OF CALIFORNIA, SAN DIEGO
UCSD

For problem with 19k patterns, RAxML speed per core is appreciably slower with partitions than without on Dash, similar to behavior of MrBayes on Abe



The hybrid, multi-grained codes provide some notable benefits

- **Higher core counts can be used to solve similar problems faster or to solve larger problems in same amount of time**
  - For unpartitioned problem with 19k patterns, speedup of hybrid **MrBayes** is **5.6** on 64 cores of Abe compared to MPI-only code on 8 cores
  - For problem with 1.8k patterns, speedup of hybrid **RAxML** is **6.5** on 80 cores of Dash compared to Pthreads-only code on 8 cores
- **Number of threads per node can be adjusted for optimal efficiency**
  - For same **MrBayes** problem, using 2 MPI processes & 4 threads each on a single node of Abe is **1.4x faster** than using 8 processes alone
  - For same **RAxML** problem, using 2 MPI processes & 4 threads each on a single node of Dash is **1.3x faster** than using 8 threads alone
- **Additional thorough searches in RAxML comprehensive analysis often lead to better solution**
  - Extra work is **useful**

There are practical considerations in selecting computers and concurrencies for TeraGrid access via portal

- **To solve larger problems: Abe is attractive because it has long queue with 7-day limit versus 2-day limit on most other TeraGrid computers**
  - Restart capabilities of both MrBayes & RAxML are limited
- **To solve problems faster: use more cores (= processes * threads), up to a point**
  - \+ Shorter run-time limit often means reduced wait in queue
  - \- Number of cores (& nodes) should be limited to keep parallel efficiency >= 50% and avoid expending too many SUs
  - \- Number of nodes for jobs in long queue seems to be limited, so wait in queue can increase when more nodes are requested
- **Number of processes & number of threads (hence cores & nodes) are selected automatically using rules dependent upon data set & analysis**
  - For MrBayes: typically 2 or 4 nodes are used instead of 1 (with 2 or 4 threads)
  - For RAxML: typically 5 nodes are used instead of 1 (with 4 or 8 threads)

Hybrid MrBayes & RAxML are available

- In source code from Exelixis Lab Web site: **http://wwwkramer.in.tum.de/exelixis/software.html**
- On TeraGrid via CIPRES portal: **http://www.phylo.org/portal2**

SDSC SAN DIEGO SUPERCOMPUTER CENTER
at the UNIVERSITY OF CALIFORNIA, SAN DIEGO
UCSD