Large-Scale Inference of Phylogenetic Trees

Alexandros Stamatakis

Institute of Computer Science
Foundation for Research and Technology-Hellas
Heraklion, Greece
Large-Scale Inference of Phylogenetic Trees

Alexandros Stamatakis

As of July 1st 2006 Swiss Institute of Bioinformatics at Lausanne
So what will I be serving you?
So what will I be serving you?

1. Using RAxML-VI in practice
2. CS-related work
Outline

- **Introduction**
- **Part I (for Biologists):**
  - How to use RAxML-VI in practice
  - Performance comparison with other ML programs
  - Example: Current multi-gene analysis of mammals
- **Part II (for Computer Scientists):**
  - Recent performance improvements
  - Exploiting the 3 Levels of Parallelism
  - GTR+CAT: a work-around for GTR+Γ
- **Outlook**
Phylogenetic Analysis

- Motivation
  - Tree-of-life
  - New insights in medical & biological research
Phylogenetic Analysis

- **Motivation**
  - Tree-of-life
  - New insights in medical & biological research

- **Applications of phylogenetic trees and RAxML**
  - Greengenes Project greengenes.lbl.gov 16S rRNA gene database and workbench
  - Integration of RAxML into the CIPRES ([www.phylo.org](http://www.phylo.org)) software
Phylogenetic Analysis

- Motivation
  - Tree-of-life
  - New insights in medical & biological research
  - Applications of phylogenetic trees

- GreenGenes Project greengenes.lbl.gov 16S rRNA gene database and workbench
- Integration of RAxML into the CIPRES (www.phylo.org) software

Largest published ML-based phylogenetic analysis to date
Developing and Using RAxML for large-scale phylogeny reconstruction on the CIPRES cluster at the San Diego Supercomputing Center.

RAxML (Randomized Axelerated Maximum Likelihood) is a program for Maximum Likelihood-based inference of large phylogenetic trees. The cluster is currently being used to further develop RAxML and to conduct analyses of large-scale real-world biological problems.

The program is evolving to address previously unsolvable issues for extremely large datasets, either in terms of the number of taxa included or the sequence length. For example, a 20,000-taxon alignment of proteobacteria with an alignment length of 1,500 base pairs had a run time on a single CPU of the cluster of only 13.5 days, with a memory consumption of only 1.5GB.

Through OpenMP-enabled parallelization of RAxML, the program can also efficiently exploit the SMP (Symmetric Multi Processing) capabilities of the cluster with its 8-way SMP nodes. This type of parallelization is especially useful for very large alignments. For example, together with Jeff Brown-Stevens and Karen Robson, we are currently working on a multigene analysis of at least 10 genes for a total of 1,160 nuclear species. The resulting multigene alignment is approximately 100,000 base pairs and therefore scales nicely on the SMPs.

The current version of RAxML, which will be released within the next 10 days, also offers the possibility to carry out parallel distributed inference on the output alignment from distributed Maximum Likelihood starting trees as well as parallel bootstrap analysis. This parallelization is based on the Message Passing Interface (MPI) such that all CPUs of the cluster can be used simultaneously to carry out large bootstrap analyses.

On the computer-science side of things, the cluster is also currently being used by my student, Michael Gir, to develop the distributed MPI parallelization of the RAxML algorithm. This type of parallelization will allow for the parallel inference of a single large tree on all CPUs of the cluster.

Future plans include investigating novel ways to distribute the best data structure among CPUs, given that memory consumption is currently limiting our capabilities to computer trees of more than 25,000 taxa. In addition, together with Jason Willshaw, we will assess how the high-performance RAxML meta-method can improve upon the performance of the new RAxML version.

Dr. Stamatakis is currently at Institute of Computer Science, Foundation for Research and Technology-Hellas, Crete, and will give Donald Metz at the Swiss Bioinformatics Institute in Lausanne as of July 1st 2006.

References and Availability:
• Related papers and the software are freely available at www.cs.forth.gr/~stamatak.
• RAxML has recently been updated and is now available for download at RAxML-alpha-0.8.3 at www.cs.forth.gr/~stamatak.
• A manual for RAxML is available at www.cs.forth.gr/~stamatak.
Phylogenetic Methods

- Input: “good” multiple Alignment
- Output: unrooted binary tree
- Various models for phylogenetic inference
  - Models differ in computational complexity & accuracy of final trees
  - Fast & simple models
    - Neighbor Joining
    - Maximum Parsimony (MP)
  - Slow & complex models
    - Maximum Likelihood (ML)
    - Bayesian Methods
Phylogenetic Methods

- Input: “good” multiple Alignment
- Output: unrooted binary tree
- Various models for phylogenetic inference:
  - Models differ in computational complexity and accuracy of final trees
  - Fast & simple models:
    - Neighbor Joining
    - Maximum Parsimony (MP)
  - Slow & complex models:
    - Maximum Likelihood (ML)
    - Bayesian Methods

Complex Models required to reconstruct large & complex trees!
Phylogeny of great Apes

common ancestor

Orangutan  Gorilla  Chimp  Homo Cretensis

© Alexandros Stamatakis, March 2006
The number of trees
The number of trees
The number of trees
The number of trees
The number of trees explodes!
Maximum Likelihood calculates:

1. Topologies
2. Branch lengths $v[i]$
3. Likelihood of the tree
Maximum Likelihood

Maximum Likelihood calculates:

1. Topologies
2. Branch lengths $v[i]$
3. Likelihood of the tree

**Goal:** Obtain topology with maximum likelihood value
Maximum Likelihood calculates:

1. Topologies
2. Branch lengths $v[i]
3. Likelihood of the tree

**Goal:** Obtain topology with maximum likelihood value

**Problem I:** Number of possible topologies is exponential in $n$
Maximum Likelihood calculates:
1. Topologies
2. Branch lengths $v_i$
3. Likelihood of the tree

Goal: Obtain topology with maximum likelihood value

Problem I: Number of possible topologies is exponential in $n$

NP-completeness for ML has only recently been demonstrated (2005)
Maximum Likelihood

Maximum Likelihood calculates:
1. Topologies
2. Branch lengths \( v[i] \)
3. Likelihood of the tree

**Goal:** Obtain topology with maximum likelihood value

**Problem I:** Number of possible topologies is exponential in \( n \)

**Problem II:** Computation of likelihood function is expensive
Maximum Likelihood calculates:
1. Topologies
2. Branch lengths $v[i]$
3. Likelihood of the tree

**Goal:** Obtain topology with maximum likelihood value

**Problem I:** Number of possible topologies is exponential in $n$

**Problem II:** Computation of likelihood function is expensive

**Problem III:** Probably high score accuracy required
Maximum Likelihood calculates:
1. Topologies
2. Branch lengths $v[i]$
3. Likelihood of the tree

**Goal:** Obtain topology with maximum likelihood value

**Problem I:** Number of possible topologies is exponential in $n$

**Problem II:** Computation of likelihood function is expensive

**Problem III:** Probably high score accuracy required

**Problem IV:** High memory consumption
Maximum Likelihood

Maximum Likelihood calculates:
1. Topologies
2. Branch lengths \( v[i] \)
3. Likelihood of the tree

Goal: Obtain topology with maximum likelihood value

Problem I: Number of possible topologies is exponential in \( n \)

Problem II: Computation of likelihood function is expensive

Problem III: Probably high score accuracy required

Problem IV: High memory consumption

Solution: new heuristics + algorithmic optimizations + HPC
Maximum Likelihood calculates:
1. Topologies
2. Branch lengths $v[i]$
3. Likelihood of the tree

**Goal:** Obtain topology with maximum likelihood value

**Problem I:** Number of possible topologies is exponential in $n$

**Problem II:** Computation of likelihood function is expensive

**Problem III:** Probably high score accuracy required

**Problem IV:** High memory consumption

**Solution:** new heuristics + algorithmic optimizations + HPC

Focus of this talk
Outline

- Introduction
- Part I (for Biologists):
  - How to use RAxML-VI in practice
  - Performance comparison with other ML programs
  - Example: Current multi-gene analysis of mammals
- Part II (for Computer Scientists):
  - Recent performance improvements
  - Exploiting the 3 Levels of Parallelism
  - GTR+CAT: a work-around for GTR+Γ
- Outlook

© Alexandros Stamatakis, March 2006
A Brief Version History

- The name of the beast: **Randomized Axelerated Maximum Likelihood**
- **RAxML-II**: Sequential, Parallel & Distributed implementation of the basic hill-climbing algorithm
- **RAxML-III**: Implementation of additional substitution models
- **RAxML-V**: Introduction of simulated annealing search algorithm, OpenMP parallelization & protein models
- **RAxML-VI**: New rapid hill climbing algorithm
- **RAxML-VI-HPC** (released yesterday): Highly optimized GTR+CAT and GTR+$\Gamma$ likelihood functions, reduced memory footprint, improved parsimony starting tree computations
- **RAxML available** as open-source code along with PDFs of relevant papers and a comprehensive Manual at www.ics.forth.gr/~stamatak
How does it work?

Compute randomized stepwise addition sequence parsimony tree and do a couple of TBRs
How does it work?

Compute randomized stepwise addition sequence parsimony tree and do a couple of TBRs

Advantage of RAxML: search starts from distinct points in search space every time
How does it work?

Compute randomized stepwise addition sequence parsimony tree and do a couple of TBRs

Apply exhaustive subtree rearrangements
How does it work?

- Compute randomized stepwise addition sequence parsimony tree and do a couple of TBRs
- Apply exhaustive subtree rearrangements
- RAxML performs fast lazy rearrangements
How does it work?

- Compute randomized stepwise addition sequence parsimony tree and do a couple of TBRs

- Apply exhaustive subtree rearrangements

Iterate while tree improves
The Rearrangement Setting
The Rearrangement Setting

This setting has a significant impact on performance!
The Rearrangement Setting

- It is important to find a “good” rearrangement setting
- This setting is extremely dataset-dependent
- How to determine a good setting with RAxML
  1) Compute a randomized MP starting tree:
     ./raxml -y 1 -s alignmentName -n startingTree
  2) Use rapid hill-climbing algorithm with standard parameter settings on this tree:
     ./raxml -f d -t RAxML_parsimonyTree.startingTree -s alignmentName -n standardSettings
  3) Use rapid hill-climbing algorithm with user defined rearrangement setting:
     ./raxml -f d -T 10 -t RAxML_parsimonyTree.startingTree -s alignmentName -s userSettings
  4) For bootstrapping + multiple inferences use settings which yielded best ML score
- However: The standard settings usually work well
A Warning

- RAxML likelihood values can not be directly compared to those of other programs due to numerical implementation differences.
- To carry out a fair comparison of topologies based on LH values, the branch lengths & model parameters of all final trees should be evaluated with one and the same program.
- There also exist differences among the LH values computed e.g. by IQPNNI, PHYML, GARLI, MrBayes for the same topology.
Bootstrapping & Confidence

- RAxML offers non-parametric bootstrapping
- It is invoked by “–b 1234”, where 1234 is a random number seed
- To perform 100 bootstraps invoke RAxML e.g. with –b 12345 -# 100
- RAxML-VI-HPC now incorporates parallel bootstrapping
- Executing 100 inferences on the original alignment on distinct randomized MP trees can yield insights on the stability of the search algorithm
  - If you have computing resources available do 100 bootstraps + 100 inferences on the original alignment
Outline

- Introduction
- Part I (for Biologists):
  - How to use RAxML-VI in practice
  - Performance comparison with other ML programs
  - Example: Current multi-gene analysis of mammals
- Part II (for Computer Scientists):
  - Recent performance improvements
  - Exploiting the 3 Levels of Parallelism
  - GTR+CAT: a work-around for GTR+Γ
- Outlook
Experimental Setup

- Other ML programs:
  - IQPNNI version 3.0
  - GARLI version 0.93 (unpublished)
  - MrBayes version 3.1
  - PHYML-SPR preliminary version

- Communicated with the respective authors via email to ensure a fair setup & comparison
- 4 real-world datasets from 1,000 to 1,728 taxa
- Model GTR+$\Gamma$ and GTR+CAT for RAxML
- Sequential Inference Time Limit set to 60 hours
- **Warning:** This is only a very preliminary analysis!
1,000 taxa
1,000 taxa Memory Consumption
1,497 taxa
1,663 taxa
1,728 taxa
Outline

- Introduction
- Part I (for Biologists):
  - How to use RAxML-VI in practice
  - Performance comparison with other ML programs
  - Example: Current multi-gene analysis of mammals
- Part II (for Computer Scientists):
  - Recent performance improvements
  - Exploiting the 3 Levels of Parallelism
  - GTR+CAT: a work-around for GTR+Γ
- Outlook
A Multi-Gene Analysis of Mammals

- Individual Sub-Alignments of 68 distinct mammalian genes
- A total of 2,100 mammalian species will be analyzed
- Sub-Alignments do not contain sequences of all species (60 – 1,500 taxa)
- There is a large agreement about the true tree
  ➔ We can evaluate the performance of computational methods on real data for which the tree is supposed to be known!
A Multi-Gene Analysis of Mammals

Gene 1  Gene 2  Gene 3

Species 1

Species 2,100

Gaps

Data
Experimental Setup I

- Separate inference on Sub-Alignments
  → merge individual (overlapping) trees into one supertree
Experimental Setup II

- Direct Inference of a comprehensive tree on the concatenated (multi-gene) alignment
- Inferences with
  1. A single substitution model for the whole alignment
  2. A more realistic individual substitution model for each gene
- The inference with the single substitution model took 1 week on 4 CPUs and required 2.9GB of memory under a realistic model of evolution
  - This is the largest data matrix that has ever been analyzed with ML
  - We need Supercomputers to compute this tree
  - The “big picture” of the tree inferred under the single substitution model looks good
Questions we want to answer:

- How good do computational methods for phylogeny reconstruction perform on real data?
- What is better: building a supertree or inference on a multi-gene alignment?
- Is the inference of a tree for the multi-gene alignment with a distinct model per gene worth the extra computational cost?
- Which mammalian gene is best-suited for phylogeny reconstruction?
- Do we need the asymptotic convergence phase (see next slide) to get reasonable topologies?
Asymptotic Convergence
Asymptotic Convergence

Do we really need this?
Outline

- Introduction
- Part I (for Biologists):
  - How to use RAxML-VI in practice
  - Performance comparison with other ML programs
  - Example: Current multi-gene analysis of mammals
- Part II (for Computer Scientists):
  - Recent performance improvements
  - Exploiting the 3 Levels of Parallelism
  - GTR+CAT: a work-around for GTR+Γ
- Outlook
Technical Optimizations
in RAxML-VI

- Before and after each individual LSR not all branch lengths need to be stored/re-stored → only locally
- Frequently storing topologies with thousands of branches is expensive → store changes in topology by LSR descriptors
- Reduce number of complete tree traversals to recompute the likelihood after one LSR has been completed to an absolute minimum
- Very simple optimizations yield speedups of 1.44 on 1,000 taxa over 30 on 4,000 taxa up to 67 on 25,000 taxa
LSR Technical Optimizations
LSR Technical Optimizations
LSR Technical Optimizations

optimize branch
LSR Technical Optimizations
LSR Technical Optimizations

Optimize these 3 branches
If likelihood improves store the insertion position & the 4 dotted branch lengths
LSR Technical Optimizations

ST1

ST2

ST3

ST5

ST4

ST6
LSR Technical Optimizations

Continue with remaining rearrangements
LSR Technical Optimizations

Only likelihood vectors within this area must be recomputed after all on ST6 have been completed.
RAxML-VI versus RAxML-V on 25,057 taxa under HKY85
RAxML-VI-HPC versus RAxML-VI on 25,057 taxa under GTR+CAT
RAxML-VI-HPC versus RAxML-VI on 25,057 taxa under GTR+CAT

13.5 days until convergence on a single (!) processor of the CIPRES cluster at SDSC
Outline

- Introduction
- Part I (for Biologists):
  - How to use RAxML-VI in practice
  - Performance comparison with other ML programs
  - Example: Current multi-gene analysis of mammals
- Part II (for Computer Scientists):
  - Recent performance improvements
  - Exploiting the 3 Levels of Parallelism
  - GTR+CAT: a work-around for GTR+$\Gamma$
- Outlook
<table>
<thead>
<tr>
<th>Levels of Parallelism</th>
</tr>
</thead>
<tbody>
<tr>
<td>Embarrassing Parallelism</td>
</tr>
<tr>
<td>MPI, CORBA, Grid Technologies</td>
</tr>
</tbody>
</table>
Levels of Parallelism

<table>
<thead>
<tr>
<th>Embarrassing Parallelism</th>
<th>MPI, CORBA, Grid Technologies</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inference Parallelism</td>
<td>MPI, algorithm-dependent</td>
</tr>
</tbody>
</table>
Levels of Parallelism

- **Embarrassing Parallelism**
  - MPI, CORBA, Grid Technologies

- **Inference Parallelism**
  - MPI, algorithm-dependent

- **Loop-Level Parallelism**
  - OpenMP, GPUs, OpenFPGA
Loop Level Parallelism

虚根

\[ P[i] = f(Q[i], R[i]) \]
Loop Level Parallelism

virtual root

This operation uses $\geq 90\%$ of total execution time!

$P[i] = f(Q[i], R[i])$
Loop Level Parallelism

This operation uses ≥ 90% of total execution time! → simple fine-grained parallelization

\[ P[i] = f(Q[i], R[i]) \]
Loop Level Parallelism

virtual root

P

Q

R
Loop Level Parallelism

virtual root

P

Q

R
Loop Level Parallelism

virtual root
RAxML and GPGPU

- GPGPU: General Purpose Computing on Graphics Processing Units
- Exploits same type of parallelism as OpenMP version
- High-level programming language: BrookGPU (graphics.stanford.edu/projects/brookgpu/)
- Initial Experiences: M. Charalambous, P. Trancoso, A. Stamatakis Proc. 10th Panhellenic Conference in Informatics (PCI 2005)

Disadvantages

😊 Tedious porting of RAxML to single precision
😊 Memory transfer GPU↔CPU inefficient
😊 Still many technical problems

Advantages

😊 It works
😊 Some speedup measured
😊 Future GPUs will be more programmable
😊 Improved price/performance & power consumption/performance ratios

What we would like to have

- OpenMP-like programming interface
- Clusters of GPUs
RAxML-Profiling

- Loop2: 47%
- Loop5: 19%
- Loop4: 12%
- Loop3: 5%
- Other: 10%
- Loop1: 7%
Initial GPU Results

<table>
<thead>
<tr>
<th></th>
<th>L2 Loop</th>
<th>all-perf</th>
<th>L2-perf</th>
<th>L2-real</th>
</tr>
</thead>
<tbody>
<tr>
<td>Speedup (cpu/gpu)</td>
<td>2.7</td>
<td>2.3</td>
<td>1.4</td>
<td>1.2</td>
</tr>
</tbody>
</table>
Inference Parallelism: Dependency Problem

Apply a Lazy Subtree Rearrangement (LSR) to currently best tree and evaluate likelihood
Inference Parallelism: Dependency Problem

If LSR improves tree likelihood keep altered topology
Inference Parallelism: Dependency Problem

- Each worker process evaluates the rearrangements for one subtree at a time
- One optimization cycle consists of 2 * #organisms LSRs
- Many improved topologies are encountered during one cycle
- Many sequential dependencies → hard to parallelize
- Use Non-determinism to solve problem
Asymptotic Convergence

Dependency problem severe during initial optimization phase

© Alexandros Stamatakis, March 2006
Other Parallelization Issues

- Representation of topology as string too long & inefficient for huge trees
  - Communicate only an LSR descriptor
  - E.g.: insert subtree rooted at node p into the branch between nodes q and r
- Parallel Model parameter optimization?
- Parallel global branch length optimization?
- Memory Distribution for MPPs
  - Workers can not hold the entire tree data structure in their local memory
  - 25,000 taxa require 1.5GB with RAxML
  - IBM Blue GeneL, only 1GB per node
HPC Issues: Summary

- Probably the best way to parallelize RAxML is to run a parallel bootstrap or parallel independent inferences on different MP starting trees on a cluster.
- The inference times up to 10,000 taxa under GTR+CAT are typically < 60 hours on a single AMD Opteron (Cipres cluster).
  - This will run over a week-end.
- A parallelization of the search algorithm only makes sense if a grid of supercomputers is available.
- However: Such an MPI-version of RAxML-VI currently under development.
- Loop-level parallelism: A solution for very long multi-gene alignments.
Outline

- Introduction
- Part I (for Biologists):
  - How to use RAxML-VI in practice
  - Performance comparison with other ML programs
  - Example: Current multi-gene analysis of mammals
- Part II (for Computer Scientists):
  - Recent performance improvements
  - Exploiting the 3 Levels of Parallelism
  - GTR+CAT: a work-around for GTR+\Gamma
- Outlook
Rate Heterogeneity

- Not all sites of an alignment evolve at the same rate
- This has to be incorporated into our evolutionary models
- The standard method is the $\Gamma$ model

```
ACGGGGGGGGGGGGGTTTTCCCCC
ATGGGGGGGGGGGGGTTTCCCCCACCGGGGGGGGGGTTTTGCCCC
AGGGGGGGGGGGCTTTTCCCCC
```

© Alexandros Stamatakis, March 2006
Why I don’t like $\Gamma$:

- $\Gamma$ typically requires the quadruple amount of memory & floating point operations compared to flat models without rate heterogeneity
  - Analyses limited by inference times and memory consumption
  - Can $\Gamma$ be replaced by a model using individual per-site rates which are categorized into a small number of e.g. 25 rate categories, the CAT approximation?
- E.g. 25,000-taxon analysis would require 6GB with $\Gamma$ instead of 1.5GB main memory with CAT and probably more than one month of execution time
- Our 2,100 mammals require 9.5 GB under $\Gamma
CAT versus $\Gamma$

- Inferred trees on 19 real-world datasets under the same search algorithm (RAxML hill-climbing) under CAT with 25 rate categories and $\Gamma$
- Optimized branch lengths and model parameters of all final trees under $\Gamma$
  - Inference with CAT is 5.5 times faster on average
  - Amount of L2+L3 cache misses is 8.8 less
  - Average final $\Gamma$ likelihood for trees inferred under CAT is 1.000037 better
  - Remember comparison with other ML algorithms

- Paper describing this:
  - Preprint available at www.ics.forth.gr/~stamatak
A Warning for Biologists

- CAT likelihood values, are numerically unstable at present

⇒ Use $\Gamma$ for likelihood-based comparisons of topologies inferred under CAT
Outlook
Conclusion, Availability & Future Work

Conclusions
- RAxML enables ML-based inferences of up to 25,000 taxa in acceptable times
- RAxML can efficiently be executed on GPUs
- RAxML can be executed on the Grid
- CAT can be used as a work-around for $\Gamma$ on real data
- CAT significantly reduces memory footprint and inference times

Open Issues
- Coarse-grained Parallelization of the search algorithm
- Solid mathematical framework for CAT
- Random addition MP starting trees or random addition + rearrangements under MP
- Further memory reduction or memory distribution
- RAxML on FPGAs
- RAxML on clusters of GPUs
- Simultaneous tree building and multiple alignment

Availability
- RAxML-VI and performance results available at: www.ics.forth.gr/~stamatak
- RAxML-VI-HPC was released yesterday
- Send an email to stamatak@ics.forth.gr to be informed about updates
Acknowledgements

Two people have significantly contributed to RAxML-VI-HPC

- Usman Roshan: The experiments with Rec-I-DCM3(RAxML) on huge data lead to significant improvements of RAxML

- Derrick Zwickl: Our discussions and the very good performance of GARLI lead to further optimization of the RAxML source-code