BIOINFORMATICS

Vol. 00 no. 00 2005
Pages 1–3

RAxML-Light: A Tool for computing TeraByte Phylogenies

A. Stamatakis [1,*], A.J. Aberer [1], C. Goll [1], S.A. Smith [2], S.A. Berger [1], F. Izquierdo-Carrasco [1]

[1] The Exelixis Lab, Scientific Computing Group, Heidelberg Institute for Theoretical Studies, Schloss-Wolfsbrunnenweg 35, D-68159 Heidelberg
[2]Blackrim Lab, Department of Ecology and Evolutionary Biology, University of Michigan, 2071A Kraus Natural Science Building, 830 North University Ann Arbor, MI 48109-1048

Received on XXXXX; revised on XXXXX; accepted on XXXXX

Associate Editor: XXXXXXX

ABSTRACT

Motivation: Because of advances in molecular sequencing and the increasingly rapid collection of molecular data, the field of phyloinformatics is transforming into a computational science. Therefore, new tools are required that can be deployed in supercomputing environments and that scale to hundreds or thousands of cores.

Results: We describe RAxML-Light, a tool for large-scale phylogenetic inference on supercomputers under maximum likelihood. It implements a light-weight checkpointing mechanism, deploys 128-bit (SSE3) and 256-bit (AVX) vector intrinsics, offers two orthogonal memory saving techniques, and provides a fine-grain production-level MPI (Message Passing Interface) parallelization of the likelihood function. To demonstrate scalability and robustness of the code, we inferred a phylogeny on a simulated DNA alignment (1481 taxa, 20,000,000 bp) using 672 cores. This dataset requires one TeraByte of RAM to compute the likelihood score on a single tree.

Code Availability: `https://github.com/stamatak/RAxML-Light-1.0.5`

Data Availability: `http://www.exelixis-lab.org/onLineMaterial.tar.bz2`

Contact: Alexandros.Stamatakis@h-its.org

1 INTRODUCTION

Phyloinformatics is facing a paradigm shift toward becoming a 'real' computational science. Molecular sequencing technologies are developing at a rapid pace, generating enormous amounts of new data. Due to the necessity to process (and store) huge amounts of data, we expect the field to undergo an analogous transition that physics or computational fluid dynamics underwent 20 to 30 years ago.

Projects such as the 1000 insect transcriptome sequencing project (`www.1kite.org`) already face these challenges. Such evolutionary studies require software that scales beyond a single node, that can be checkpointed and restarted, and that can accommodate the memory requirements of whole-genome datasets under likelihood-based models. RAxML-Light, is a production-level tool for phylogenetic inference on supercomputers that implements new approaches for handling load imbalance, checkpointing, and reducing the RAM requirements of likelihood computations. Implementation details are discussed in the on-line supplement.

[*]to whom correspondence should be addressed

2 FEATURES

We briefly discuss the features that distinguish RAxML-Light from standard RAxML, other likelihood-based phylogeny programs, and the BEAGLE library (Ayres *et al.*, 2011). One important feature (in contrast to BEAGLE, MrBayes (Ronquist & Huelsenbeck, 2003), or GARLI (Zwickl, 2006)) is that, RAxML-Light implements a fine-grain MPI parallelization to compute the likelihood on a single huge dataset and a single tree across several nodes. We introduced the proof-of-concept implementation in Ott *et al.* (2007). The work is split by distributing alignment sites or entire partitions (depending on the selected command line options) among processors.

Another essential feature is the light-weight checkpointing and restart capability, that is required on typical HPC systems that have queues with 24- or 48-hour run time limits. Light-weight means that only those data-structures are stored in a checkpoint which are really required to restart the code. The design goal is to minimize checkpoint writing/reading times and file sizes.

To the best of our knowledge, RAxML-Light comprises the only fine-grain parallelization of the likelihood function that also incorporates load balance mechanisms as described in Stamatakis & Ott (2009) and Zhang & Stamatakis (2012). Load imbalance can deteriorate parallel efficiency in partitioned phylogenetic analyses.

RAxML-Light also contains a production-level implementation of two orthogonal memory saving techniques that can be used simultaneously (described in Izquierdo-Carrasco *et al.* (2011*a*) and in Izquierdo-Carrasco *et al.* (2011*b*)). These techniques allow for deploying RAxML-Light on systems that do not have enough RAM to store all conditional probability vectors required for likelihood calculations.

Finally, RAxML-Light also uses 256-bit wide AVX vector intrinsics to accelerate likelihood computations on Intel Sandy-Bridge and AMD Bulldozer CPUs that will become available in many HPC systems over the next 2-3 years.

3 PERFORMANCE & STRESS TESTS

Parallel Scalability: We measured the relative speedup of the MPI version of RAxML-Light on a dataset with 150 taxa and 20,000,000 bp (extracted from the above simulated dataset) on an AMD Magny-Cours based cluster with a Qlogic Infiniband interconnect and a total of 50 48-core nodes equipped with 128GB (46 nodes) or 256GB (4 nodes) of RAM per node.

For comparison, we also measured execution times of the PThreads-based version on a stand-alone 48-core AMD server with 256GB RAM. In Figure 1 we provide execution times for the PThreads and MPI versions on multiples of 48 cores under the CAT (Stamatakis, 2006) and Γ (using 4 discrete rate categories) models of rate heterogeneity. The test dataset requires almost 256GB of RAM under Γ which explains the bad initial performance under Γ on one and two cluster nodes. The nodes in the cluster have slightly different swapping configurations than the stand-alone node (same server type) we used. Overall, the code scales well up to 1536 cores. On 1536 cores, RAxML-Light requires less than two hours (6108 secs) to complete a full tree search under Γ.

Load balance: The initial work on improving load balance for partitioned datasets (Stamatakis & Ott, 2009) is hard-coded in RAxML-Light and can improve parallel efficiency by more than 50%. The recent work on the 'multi-processor scheduling problem in phylogenetics' (Zhang & Stamatakis (2012), `-Q` option) should only be applied when the number of partitions/genes is substantially larger than the cores that shall be used. This alternative data distribution scheme improved parallel run times by one order of magnitude on a partitioned protein alignment with 1000 partitions under CAT.

Computing a TeraByte Tree: To conduct a thorough stress test, we simulated a DNA alignment with 1481 taxa and 20,000,000 bp using SeqGen (Rambaut & Grass, 1997). The 1481 taxon tree we used to generate the alignment is a ML tree inferred on a real-world single gene dataset. Under the CAT model of rate heterogeneity, this dataset requires about 1TB of RAM to compute the likelihood on a single tree. We executed a single tree search on 672 cores (14 48-core nodes) which required 40 hours to converge for the standard RAxML search algorithm. The relative RF (Robinson-Foulds) distance to the true tree was 7%.

Computing a 116,334 Taxon Tree: For the NSF plant tree of life grand challenge project, we deployed the PThreads version (running on a single 48-core node) to carry out 100 ML searches (each search starting from distinct randomized stepwise addition order parsimony starting tree) on a real-world DNA alignment (116,334 taxa, 16,079 bp) under CAT and a partitioned model. With the ML search convergence criterion enabled (see Stamatakis, 2011) the runs required 2 automatic restarts (using appropriate Sun Grid Engine scripts) from checkpoints to complete within three 48 hour queue slots. While the runs were successful, the resulting trees did not make 'biological sense'. This is in part a result of trying to construct, with the data available at the time in GenBank (ca. 2008), alignments with more than 100,000 species. Viridiplantae did not have data for more than 100,000 species with any of traditionally well sampled gene regions. So we had to 1) expand the dataset with less well sampled gene regions for plants, 2) include ribosomal regions 18S and 26S, and 3) include a large Fungi outgroup. These complications allowed us to construct a large dataset with more than 100,000 species, but led to some unexpected taxonomic placements. Nonetheless, we make the alignment and trees available for benchmarking purposes.

Memory Saving Techniques: The scalability of RAxML-Light is limited by the number of sites in the alignment, because RAxML-Light is parallelized over sites/partitions. Hence, it does not make sense to analyze datasets as the one above (116,334 taxa) in parallel on more than one node. On such gappy datasets with missing data, we can deploy the subtree equality vector technique (`-S` option) to substantially reduce memory requirements. The key idea of this technique is to keep track of subtrees in partitions (genes) that entirely consist of missing data and omit computing as well as storing the ancestral probability vectors for these 'empty' subtrees. In the above case, using `-S` led to a reduction of RAM requirements from 66GB down to 26.5GB. This allowed us to also execute some tree searches on single nodes of the Texas Advanced Computing Center, that only have 32GB of RAM available. The `-S` option generally also decreases execution times, because a large number of unecessary computations are omitted (see Izquierdo-Carrasco *et al.* (2011*b*) for performance details). Note that, performance of this technique also depends heavily on the memory allocator being used (see supplementary material).

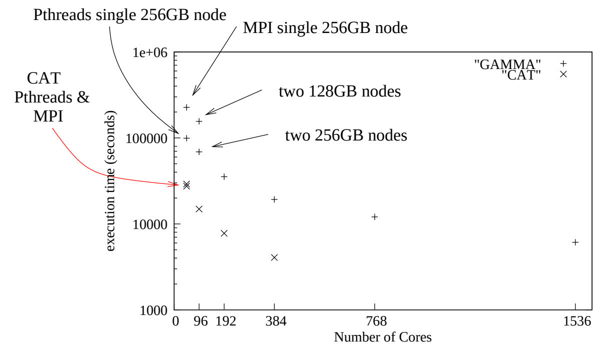

Fig. 1. Parallel execution times of the MPI and PThreads versions under CAT and Γ on a DNA dataset with 150 taxa and 20,000,000 sites.

Table 1. Execution times of unvectorized, SSE3- and AVX-vectorized RAxML versions

Data	Model	unvectorized	SSE3	AVX
DNA	CAT	100	87	76
DNA	Γ	520	433	353
PROT	CAT	117	83	49
PROT	Γ	423	249	187

Unvectorized execution times have been measured using the standard RAxML version.

We also tested the MPI version of the recomputation technique (with reduction factors of `-r 0.2` and `-r 0.15`) on just a single 48-core node with 256GB RAM on the dense simulated 1TB dataset that does not contain any gaps. The recomputation technique saves memory by not storing all ancestral probability vectors, but only a fraction of them as specified by the `-r` switch (e.g., setting `-r 0.2` means that only 20% of the ancestral vectors are stored). When an ancestral vector needs to be read that has not been stored in RAM, we simply recompute it. Evidently, execution times will increase because of recomputations, but the increase is small ($\approx$ 40%) even when storing only 10% (`-r 0.1`) of the required vectors in RAM (Izquierdo-Carrasco *et al.*, 2011*a*). Our tests showed that, using the recomputation technique, a dataset requiring 1TB of RAM can be successfully and correctly analyzed on a single multi-core server with only 256GB RAM (for additional details see supplementary material).

Vector Intrinsics for Likelihood: We tested the performance of the AVX-vectorization in RAxML-Light using a DNA dataset with 150 taxa and 1269 bp (1130 distinct site patterns) and a protein dataset with 40 taxa and 1104 bp (958 distinct site patterns) on a single Intel i7-2620M core running at 2.7 GHz. We measured execution times under the CAT and under Γ and averaged execution times over 3 runs. For reference, we also included the execution times of the standard RAxML version without vectorization in Table 1.

4 CONCLUSION & FUTURE WORK

We have presented, RAxML-Light, a scalable, AVX-vectorized, and checkpointable open-source code for large-scale phylogenetic inference on supercomputers. User support will be provided via `groups.google.`

`com/group/raxml` and continued development will be provided via the github repository.

For partitioned whole-genome datasets with thousands of partitions, the code requires some substantial re-engineering (in addition to the techniques presented here) to further reduce communication costs. Under the current fork-join parallelization paradigm (also used in BEAGLE), communication to trigger parallel regions for partitioned whole-genome datasets becomes bandwidth-bound instead of latency-bound. This problem is independent of and orthogonal to the load balance issues discussed here and only became apparent in the course of some currently on-going partitioned whole-genome analyses. We also expect energy-efficiency and core failure tolerance to become important future research topics with respect to scaling phylogenetics codes to Exascale HPC systems.

REFERENCES

Ayres, D., Darling, A., Zwickl, D., Beerli, P., Holder, M., Lewis, P., Huelsenbeck, J., Ronquist, F., Swofford, D., Cummings, M. *et al.* (2011) BEAGLE: an Application Programming Interface and High-Performance Computing Library for Statistical Phylogenetics. *Systematic Biology,* .

Guindon, S., Dufayard, J., Lefort, V., Anisimova, M., Hordijk, W. & Gascuel, O. (2010) New algorithms and methods to estimate maximum-likelihood phylogenies: assessing the performance of phyml 3.0. *Systematic biology,* **59** (3), 307–321.

Izquierdo-Carrasco, F., Gagneur, J. & Stamatakis, A. (2011*a*). Trading Memory for Running Time in Phylogenetic Likelihood Computations. Technical report Heidelberg Institute for Theoretical Studies.

Izquierdo-Carrasco, F., Smith, S. & Stamatakis, A. (2011*b*) Algorithms, data structures, and numerics for likelihood-based phylogenetic inference of huge trees. *BMC Bioinformatics,* **12** (1), 470.

Ott, M., Zola, J., Aluru, S. & Stamatakis, A. (2007) Large-scale Maximum Likelihood-based Phylogenetic Analysis on the IBM BlueGene/L. In *Proc. of IEEE/ACM Supercomputing Conference 2007 (SC2007).*

Rambaut, A. & Grass, N. (1997) Seq-gen: an application for the monte carlo simulation of dna sequence evolution along phylogenetic trees. *Computer applications in the biosciences: CABIOS,* **13** (3), 235.

Ronquist, F. & Huelsenbeck, J. (2003) MrBayes 3: Bayesian phylogenetic inference under mixed models. *Bioinformatics,* **19** (12), 1572–1574.

Stamatakis, A. (2006) Phylogenetic Models of Rate Heterogeneity: A High Performance Computing Perspective. In *Proc. of IPDPS2006* HICOMB Workshop, Proceedings on CD, Rhodos, Greece.

Stamatakis, A. (2011) Phylogenetic search algorithms for maximum likelihood. *Algorithms in Computational Molecular Biology,* , 547–577.

Stamatakis, A. & Ott, M. (2009) Load Balance in the Phylogenetic Likelihood Kernel. In *Proceedings of ICPP 2009.* accepted for publication.

Zhang, J. & Stamatakis, A. (2012). The Multi-Processor Scheduling Problem in Phylogenetics. Technical report Heidelberg Institute for Theoretical Studies.

Zwickl, D. (2006). *Genetic Algorithm Approaches for the Phylogenetic Analysis of Large Biological Sequence Datasets under the Maximum Likelihood Criterion*. PhD thesis, University of Texas at Austin.