Accuracy and Performance of Single versus Double Precision Arithmetics for Maximum Likelihood Phylogeny Reconstruction

Simon A. Berger and Alexandros Stamatakis*

The Exelixis Lab, Dept. of Computer Science, Technische Universität München
Boltzmannstr. 3, 85748 Garching b. München, Germany
{bergers,stamatak}@in.tum.de
http://wwwkramer.in.tum.de/exelixis/

Abstract. The multi-core revolution and the biological data flood that is generated by novel wet-lab techniques pose new technical challenges for large-scale inference of phylogenetic trees from molecular sequence data. We present the first assessment of accuracy and performance trade-offs between single and double precision arithmetics and the first SSE3 vectorization for computing the Phylogenetic Likelihood Kernel (PLK) which forms part of many state-of-the art tools for phylogeny reconstruction and consumes 90-95% of the overall execution time of these tools. Moreover, the PLK also dominates memory consumption, which means that deploying single precision is desirable to accommodate increasing memory requirements and to devise efficient mappings to GPUs. We find that the accuracy provided by single precision is sufficient for conducting tree searches, but that the increased amount of scaling operations to prevent numerical underflow, even when using SSE3 operations that accelerate the single precision PLK by 60%, generates run-time penalties compared to double precision on medium-sized datasets. However, on large datasets, single precision can yield significant execution time savings of 40% because of increased cache efficiency and also reduces memory footprints by 50%.

Key words: Phylogenetic inference, single versus double precision arithmetics, RAxML, Maximum Likelihood, SSE3 instructions

1 Introduction

The emergence of many-core architectures and accelerator devices as well as the molecular data flood generated by novel high-throughput sequencing techniques require new approaches for orchestrating compute-intensive Bioinformatics kernels.

Within this context, we assess speed and accuracy trade-offs between single precision (henceforth abbreviated as SP) and double precision (henceforth

* This work is funded under the auspices of the Emmy-Noether program by the German Science Foundation (DFG).

abbreviated as DP) floating point arithmetics for the Phylogenetic Likelihood Kernel (PLK [1]) that is used to reconstruct phylogenetic (evolutionary) trees from molecular sequence data.

A phylogenetic tree is an unrooted binary tree that represents the evolutionary relationships among species. The input of a phylogenetic analysis is a multiple sequence alignment comprising nucleotide or protein sequence data from organisms that are alive today. The alignment is an $n \times m$ data matrix, that contains, e.g., n DNA sequences which all have a length of m nucleotide characters (columns/sites). The output is an unrooted binary tree that represents the evolutionary history of those organisms. The tips (also called leaves or taxa) of the tree represent species alive today in contrast to internal (ancestral) nodes that represent species that have become extinct.

The PLK is one of the most widely used optimality criteria to score and thus chose among distinct evolutionary scenarios (phylogenetic trees). Many program packages are available that implement the PLK, either for standard Maximum Likelihood analyses (RAxML [2], GARLI [3]) or to conduct Bayesian phylogenetic inference (MrBayes [4], BEAST [5]). All PLK-based phylogenetic inference programs spend the largest part of overall run time (90-95%) in the computation of the likelihood function [6]. The aforementioned tools are widely used by biologists and have accumulated over 20,000 citations. Therefore, it is important to assess and devise HPC solutions for this important Bioinformatics kernel.

We present the first accuracy assessment between SP and DP arithmetics for the PLK and also exploit the usage of SSE3 instructions in the PLK. SP arithmetics can also solve memory bottlenecks in analyses of large-scale phylogenomic datasets that can already require up to 120GB of main memory under DP. The deployment of SP for the PLK can reduce memory requirements by almost 50%. SP arithmetics are also required to map the PLK onto GPUs, since at present SP arithmetics are approximately one order of magnitude faster than DP arithmetics on GPUs. We find that SP arithmetics are sufficiently accurate to conduct ML-based tree searches on trees with less than approximately 2,000 taxa and hence can be used for accelerating the kernel on Graphics Processing Units. We also achieve performance improvements of more than 40% (DP) and 60% (SP) via deployment of SSE3 instructions on general purpose CPUs. Finally, we demonstrate that SP can be used to significantly accelerate PLK computations on large phylogenomic datasets because of increased cache efficiency.

The remainder of this paper is organized as follows: In Section 2 we briefly describe how the likelihood is calculated on phylogenetic trees. Thereafter, we cover related work on floating point implementations and usage of accelerators for the PLK (Section 3). In Section 4 we describe the SSE3 and SP implementations and provide experimental results in the subsequent Section 5. We conclude in Section 6.

2 Computing the Likelihood of a Tree

The input of a standard phylogenetic analysis consists of a multiple sequence alignment with n sequences (taxa/tips) and m alignment columns. The output is an unrooted binary tree; the n taxa are located at the leaves of the tree and the inner nodes represent common extinct ancestors. The branch lengths essentially represent the relative time of evolution between nodes in the tree. To compute the likelihood on a fixed tree topology several additional ML model parameters are required: the instantaneous nucleotide substitution matrix Q which contains the transition probabilities for time dt between nucleotide (4×4 matrix) or Amino Acids (20×20 matrix) characters. Additionally, the prior probabilities of observing the nucleotides, e.g., $\pi_A, \pi_C, \pi_G, \pi_T$ (for DNA data), and the α shape parameter that forms part of the Γ model [7] of rate heterogeneity need to be determined. The Γ model accounts for the fact that different sites evolve at different speeds. Finally, one also requires the $2n - 3$ branch lengths in the unrooted tree topology.

To compute the likelihood of a fixed *unrooted* binary tree topology given these parameters, initially one needs to compute the entries for all internal probability vectors (located at the inner nodes) that contain the probabilities $P(A), P(C), P(G), P(T)$, of observing an `A,C,G,` or `T` at each site/column c of the input alignment at the specific inner node. Those probability vectors are computed bottom-up from the tips towards a virtual root that can be placed into any branch of the tree using a procedure known as the Felsenstein pruning algorithm [1]. Under certain standard model restrictions (time-reversibility of the model) the likelihood score will be the same, regardless of the placement of the virtual root.

Every probability vector entry $\boldsymbol{L}(c)$ at a position c ($c = 1...m$) of the tips and the inner nodes of the tree topology, contains the four probabilities $P(A)$, $P(C)$, $P(G)$, $P(T)$ of observing a nucleotide `A, C, G, T`, at a specific column c of the input alignment. The probabilities at the tips (leaves) of the tree for which observed data *is* available are set to 1.0 for the observed nucleotide character at the respective position c, e.g., for the nucleotide `A`: $\boldsymbol{L}(c) = [1.0, 0.0, 0.0, 0.0]$. Given a parent node k, and two child nodes i and j (with respect to the virtual root), their probability vectors $\boldsymbol{L}^{(i)}$ and $\boldsymbol{L}^{(j)}$, the respective branch lengths leading to the children b_i and b_j, and the transition probability matrices $P(b_i), P(b_j)$, the probability of observing an `A` at position c of the ancestral (parent) vector $\boldsymbol{L}_A^{(k)}(c)$ is computed as follows:

$$\boldsymbol{L}_A^{(k)}(c) = \Big(\sum_{S=A}^{T} P_{AS}(b_i) \boldsymbol{L}_S^{(i)}(c) \Big) \Big(\sum_{S=A}^{T} P_{AS}(b_j) \boldsymbol{L}_S^{(j)}(c) \Big) \tag{1}$$

The transition probability matrix $P(b)$ for a given branch length is obtained from Q via $P(b) = e^{Qb}$. Once the two probability vectors $\boldsymbol{L}^{(i)}$ and $\boldsymbol{L}^{(j)}$ to the left and right of the virtual root (vr) have been computed, the likelihood score $l(c)$ for an alignment column c ($c = 1...m$) can be calculated as follows, given

the branch length b_{vr} between nodes i and j:

$$l(c) = \sum_{R=A}^{T} \Big(\pi_R \boldsymbol{L}_R^{(i)}(c) \sum_{S=A}^{T} P_{RS}(b_{vr}) \boldsymbol{L}_S^{(j)}(c) \Big) \tag{2}$$

The overall score is then computed by summing over the per-column log likelihood scores: $LnL = \sum_{c=1}^{m} log(l(c))$.

An important property of the likelihood function is the assumption, that sites evolve independently, i.e., all entries c of the probability vectors $\boldsymbol{L}$ can be computed independently. This property represents the main source of fine-grain parallelism in the PLK [6].

In order to compute the *Maximum* Likelihood value for a fixed tree topology all individual branch lengths, as well as the parameters of the Q matrix and the α shape parameter, must also be optimized via an ML estimate. For the Q matrix and the α shape parameter the most common approach consists in using Brent's algorithm. In order to evaluate changes in Q or α the entire tree needs to be re-traversed, i.e., a *full* tree traversal needs to be conducted in order to correctly re-compute the likelihood. For the optimization of branch lengths, the Newton-Raphson method is commonly used. In order to optimize the branches of a tree, the branches are repeatedly visited and optimized one-by-one until the achieved branch length change is smaller than some pre-defined ϵ. The bulk of all of the likelihood computations consists of `for`-loops over the length m of the vectors $\boldsymbol{L}$. These `for`-loops require for instance 95% of total execution time in RAxML.

Avoiding Numerical Underflow: The methods deployed for avoiding numerical underflow represent an important implementation and performance issue. As can be derived from Formula 1, the values in the probability vectors $\boldsymbol{L}$ at the inner nodes of the tree will progressively become smaller as we approach the virtual root, since we are conducting successive multiplications with the probability values in the transition probability table P. Especially for trees with many taxa, measures need to be taken to avoid numerical underflow in the probability vectors.

The scaling in RAxML is conducted as follows: At a column c of an ancestral probability vector $\boldsymbol{L}$ we scale the entries if $\boldsymbol{L}_A(c) < \epsilon \wedge \boldsymbol{L}_C(c) < \epsilon \wedge \boldsymbol{L}_G(c) < \epsilon \wedge \boldsymbol{L}_T(c) < \epsilon$, where $\epsilon = 1/2^{256}$ for DP and $\epsilon = 1/2^{32}$ for SP. If probability vector column c at vector $\boldsymbol{L}$ needs to be scaled, we simply multiply all entries $\boldsymbol{L}_A(c), \boldsymbol{L}_C(c), \boldsymbol{L}_G(c), \boldsymbol{L}_T(c)$ by 2^{256} (DP) and 2^{32} (SP) respectively. In order to annihilate the scaling events at the virtual root we keep track of the total number of scaling events conducted per column by using integer vectors $\boldsymbol{U}$ that maintain the scaling events and correspond to the respective probability vectors. At the virtual root, given $\boldsymbol{L}^{(i)}, \boldsymbol{L}^{(j)}$ and the corresponding scaling vectors $\boldsymbol{U}^{(i)}, \boldsymbol{U}^{(j)}$ we compute the likelihood as follows:

$$l(c) = \frac{1}{2^{256}}^{U^{(i)}(c)+U^{(j)}(c)} \Big(\sum_{R=A}^{T} \Big(\pi_R \boldsymbol{L}_R^{(i)}(c) \sum_{S=A}^{T} P_{RS}(b_{vr}) \boldsymbol{L}_S^{(j)}(c) \Big) \Big) \tag{3}$$

If we take the logarithm of $l(c)$ and $\epsilon = 1/2^{256}$ this can be re-written as:

$$log(l(c)) = (U^{(i)}(c) + U^{(j)}(c))log(\epsilon) + log\Big(\sum_{R=A}^{T}\Big(\pi_R \boldsymbol{L}_R^{(i)}(c) \sum_{S=A}^{T} P_{RS}(b_{vr})\boldsymbol{L}_S^{(j)}(c)\Big)\Big) \quad (4)$$

Memory Requirements: The memory requirements for ML-based phylogeny programs are dominated by the space required for the inner probability vectors $\boldsymbol{L}$ and the inner scaling vectors $\boldsymbol{U}$. Depending on the memory organization and data structures used, we need to assign at least one probability vector and one scaling vector to each of the $n-2$ inner nodes of the tree. Since for the values at the leaves we only have 15 alternative probability vector entries using ambiguous DNA character encoding, we only need to store one vector $\boldsymbol{L}$ of length 15 which can then be accessed using the input sequences as index. The input sequences can be stored as simple `char` arrays. Hence, the memory requirements for computing the likelihood on a DNA alignment (without accommodating for rate heterogeneity) with n taxa and m columns requires $n \cdot m \cdot 1$ bytes for the input sequences, $(n-2) \cdot m \cdot 4 \cdot 8$ bytes for the probability vectors and $(n-2) \cdot m \cdot 4$ bytes for the scaling vectors. If we use the standard Γ model of rate heterogeneity the space requirements for the probability vectors increase to $(n-2) \cdot m \cdot 16 \cdot 8$ bytes. Hence, the memory requirements are dominated by the space required for the inner probability vectors and can be reduced by factor 2 using SP arithmetics. This is an important issue since we are receiving an increasing number of reports from RAxML users that encounter memory shortages.

3 Related Work

We are not aware of any related work that assesses accuracy and speed tradeoffs between SP and DP floating point arithmetics for the PLK. However, such analyses have been conducted for standard numerical linear algebra kernels, e.g., systems of linear equations [8] where the authors propose a mixed precision approach, i.e., an initial optimization under SP and a final refinement under DP. Such a procedure that dynamically switches from SP to DP, could also be applied to phylogenetic inference, i.e., one could initially infer a rough tree structure (the big picture) under SP and then refine it under DP. However, we find that this is not necessary and that the loss of accuracy is insignificant with respect to the tree topology (see Section 5).

Nonetheless, there is some on-going work to port GARLI [3] to SP (Derrick Zwickl, personal communication) for the same reasons as RAxML. Surprisingly, no efforts have been undertaken and published with respect to deploying SSE instructions to improve performance of the PLK on new-generation x86 architectures. The only documented usage of SIMD instructions for the PLK on the CELL processor is described in [9].

MrBayes [4], which is a program for Bayesian phylogenetic inference, has been ported down to SP five years ago, mainly to better accommodate the significantly larger memory requirements caused by the multiple heated and cold

Markov Chains in the Metropolis-Coupled search procedure. Bayesian floating point implementations are more straight-forward since no iterative procedures (Newton-Raphson, Brent's algorithm) are required to optimize ML model parameters. We are also not aware of any study that deals with the accuracy trade-offs regarding tree topologies in MrBayes following the transition from DP to SP. In addition, the MrBayes source code also contains SSE3 instructions, but potential performance gains have not been documented and SSE3 does not form part of the standard distribution. Finally, increased scaling events for SP also occur in MrBayes. According to profiling runs of MrBayes using `gprof` within the framework of an OpenMP parallelization, we found that the scaling procedure requires approximately 20% of overall execution time.

The Bayesian program BEAST [5] has also recently been ported to SP in order to be mapped to a GPU (*Bioinformatics*, in press, preprint at `http://tree.bio.ed.ac.uk/publications/390/`). The porting to SP was mainly conducted for efficiently computing 60-state Codon models on GPUs for which impressive speedups of two orders of magnitude are achieved. However, the speedup obtained in comparison to a DP C implementation for DNA data between CPU and GPU is only around factor 4 since the mathematical operations that are required for a 4-state transition matrix can not be mapped as efficiently to a GPU. Moreover, the performance comparison between GPU and CPU could be improved in favor of the CPU. The code on the multi-core CPU, an Intel Core 2 Extreme with a total of 4 cores, is run sequentially and not using an OpenMP or Pthreads-based fine-grained parallelization of the PLK, i.e., a speedup of factor 4 could be achieved via multi-threading. If SSE3 instructions were deployed for the PLK, an additional two-fold speedup over the GPU could be achieved. Finally, the performance analysis is only conducted using a single 63 taxon dataset. Hence, a potential performance degradation caused by increased scaling events as more taxa are added to the alignment is not assessed.

4 Implementation

Single Precision Version: The SP version of RAxML was implemented using a similar strategy as in MrBayes. We still conduct a large portion of the numerically sensitive operations, like base frequency computations or Eigenvalue/Eigenvector decomposition that are required to compute $P(t) = e^{Qt}$ in DP and then cast the $P(t)$ matrix to SP. We also compute the derivatives of $P(t)$ that are required for conducting the Newton-Raphson procedure for branch length optimization in DP and then cast them to SP. Thus, only the main bulk of operations as outlined in Equations 1 and 2 is actually conducted under SP. Based on prior experience with several unsuccessful attempts to port RAxML to SP, this approach seems to yield the numerically most stable implementation.

Finally, we also empirically adapted (increased) various ϵ settings that determine the number of iterations in the Newton-Raphson as well as in the iterative procedures for optimization of the remaining ML model parameters Q and α. The convergence parameters were adapted in such a way that the SP version

carries out approximately as many iterations for branch length and ML model parameter optimization as the DP version. These increased settings yield slightly worse likelihood scores than the DP version, but we find that this has no significant impact on the relative likelihood-based order of trees (see Section 5).

SSE3 for Likelihood Computations: We vectorize computations that are special cases of a general dense matrix multiplication; the computations on $\boldsymbol{L}^{(i)}$ and $\boldsymbol{L}^{(j)}$ in Formula 1 over all sites c and all nucleotides A, C, G, T are matrix products of the form $P \cdot \boldsymbol{L}^{(i)}$ and $P \cdot \boldsymbol{L}^{(j)}$. We assessed the usage of highly optimized ATLAS-BLAS routines [10], but because of the unfavorable matrix dimensions (multiplication of the 4×4 matrix P with the $4 \times m$ matrix $\boldsymbol{L}$) we even observed a slowdown. We also deploy the horizontal addition instructions in SSE3 for the reduction operations that are required to efficiently complete the scalar product as indicated in Formula 1.

SSE3 for Likelihood Scaling: We also vectorized the scaling procedure as outlined in Section 2 using SSE3. This is particularly important for the SP implementation, since the number of scaling events increases by one order of magnitude (see Figure 2). SSE3 instructions are used to efficiently determine the maximum value of $\boldsymbol{L}_A(c), \boldsymbol{L}_C(c), \boldsymbol{L}_G(c), \boldsymbol{L}_T(c)$ (see Section 2) and then compare the maximum to the ϵ value, thereby eliminating several conditional statements. SSE3 vectorization was implemented by inserting SSE3 intrinsics into the C code, rather than via inline assembly. This leaves room for further optimizations of the instruction schedule and register allocation by the compiler. It is important to note that, the Intel `icc` compiler (v 11.1) is not able to vectorize the `for`-loops of the PLK, despite numerous attempts to re-write the loops for facilitating auto-vectorization.

5 Experimental Setup & Results

To assess accuracy of the SP versus the DP version we initially generated collections of "good" trees that are encountered during a tree search under DP on 9 single-gene real-world DNA datasets containing 150 up to 1,908 taxa. Thereafter, we applied the RAxML function for scoring a set of trees (`-f n` option, for details please refer to the RAxML Manual) under the standard GTR+Γ model of nucleotide substitution to score the respective tree collections under SP and DP. The absolute likelihood values are not important for a tree search, but only the relative scores, i.e., how well can our SP implementation discriminate between alternative trees via the likelihood value. To this end, we computed the Spearman rank correlation coefficient ρ between the likelihood-based tree rankings obtained via DP and SP tree evaluations to assess if SP provides a sufficient degree of accuracy. We also used the above experiments to obtain execution times for the SP and DP versions, as well as for the SSE3-based and standard versions of the code. As test platform we used a SUN x4600 multi-core system equipped with 32 AMD Opteron cores running at 2.7GHz and a total of 64GB of RAM. We used `gcc` (v. 4.3.2) to compile all versions of the code.

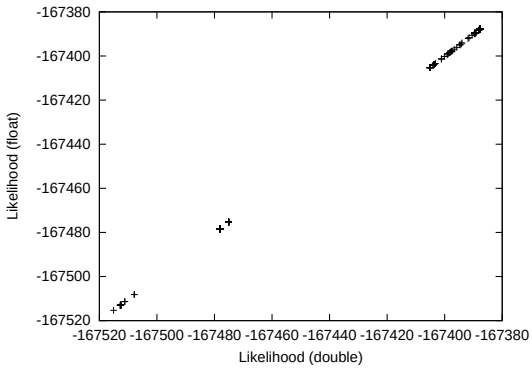

Fig. 1. Scatter plot of SP and DP tree scores for a datasets with 1,604 sequences.

In Table 1 we provide the number of taxa in the test datasets (column: # taxa), the number of trees in the respective tree collections (column: # trees), the Spearman correlation between SP and DP likelihood-based tree orders (column: ρ), the speedup between the standard SP and DP versions (column: S/D), between the SSE3-based SP and DP versions (column: S-SSE3/D-SSE3), between the standard and SSE3-based SP versions (column: S-SSE3/S) and between the standard and SSE3-based DP versions (column: D-SSE3/D). Overall, the order of trees induced by SP and DP likelihood scores is highly correlated and hence SP suffices to conduct full tree searches. In Figure 1 we provide a scatter plot of SP versus DP likelihood scores for the worst-case Spearman coefficient of 0.95 on the alignment with 1,604 taxa. The average slowdown of SP over DP is approximately factor 2, but improves to 1.5 for the respective SSE3 implementations. This improvement is due to the higher speed gains of approximately 60% in the SP SSE3 implementation compared to about 40% in the DP SSE3 implementation.

# taxa	# trees	ρ	SP/DP	SP-SSE3/DP-SSE3	SP-SSE3/SP	DP-SSE3/DP
150	200	0.99	1.39	0.83	0.36	0.61
218	260	0.99	2.58	1.71	0.42	0.64
354	160	0.97	1.44	0.78	0.36	0.67
500	300	0.99	2.31	1.62	0.44	0.63
628	180	0.95	1.39	0.80	0.36	0.62
714	320	0.99	2.41	1.70	0.43	0.61
1,512	520	0.99	2.63	1.82	0.44	0.63
1,604	220	0.95	2.38	1.69	0.40	0.63
1,908	360	0.99	1.29	0.78	0.40	0.66

Table 1. Test datasets, number of trees in tree collections, Spearman coefficients, and speedup values between all code versions

In Figure 2 we outline the number of scaling events over the number of taxa (note the log scale on the y-axis) for the evaluation of a single tree. As mentioned before, the SP version requires about one order of magnitude *more* scaling operations. In Figure 3 we provide corresponding execution times (note the log scale on the y-axis) for the evaluation of a single tree using the standard SP and DP versions of the code. The similar shapes of the two curves clearly show that scaling operations dominate SP run times.

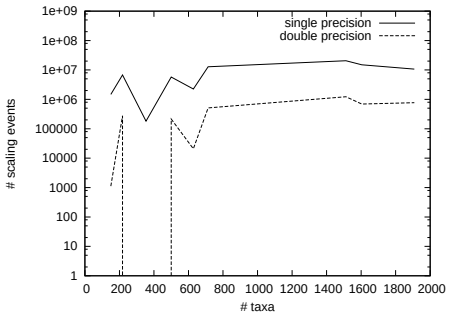

Fig. 2. Number of scaling events for SP and DP versions over the number of taxa.

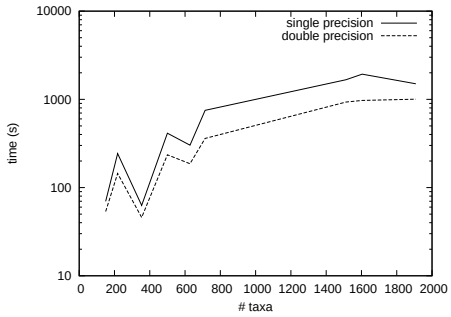

Fig. 3. Execution times in seconds for SP and DP versions over the number of taxa.

In a second set of experiments we measured execution times and inference accuracy for the Pthreads-based RAxML code on a large-scale phylogenomic protein dataset with 321,145 distinct alignment patterns and 232 taxa which requires approximately 40GB of RAM under DP and the Γ model of rate heterogeneity. On a 16-core SUN x4600 system (code without SSE3) we found that an inference under SP is 40% faster and yields an equally good final tree, when scored under DP, while reducing the memory requirements to 20GB. On a 32-core SUN x4600 system (code with SSE3) we found that the SP-based code using the CAT approximation of rate heterogeneity [11] is three times faster than a standard Γ-based inference under DP, yields a slightly better final likelihood score, and only requires 5GB of main memory.

Finally, we also assessed numerical stability on alignments containing $\geq$ 2,000 taxa and found that the code encounters problems with numerical stability on such large alignments under SP. Similar observations were made by Derrick Zwickl (personal communication) on his SP implementation of GARLI, i.e., problems with numerical stability on many-taxon datasets seems to be a general problem.

6 Conclusion

We have presented the first thorough assessment of accuracy and speed trade-offs with respect to using SP versus DP floating-point arithmetics for the Phylogenetic Likelihood Kernel in a Maximum Likelihood framework. In addition, we

have conducted the first SSE3-based vectorization of the PLK. Our results indicate that SP can be deployed to accurately infer phylogenetic trees with less than 2,000 taxa. In addition, SP arithmetics significantly reduce memory requirements of large phylogenomic analyses and substantially improve inference times via increased cache efficiency. Thereby, in combination with the CAT approximation of rate heterogeneity, we can design tools that enable large-scale phylogenomic inference "for the masses" by significantly reducing computational resource requirements. We also find that, SSE3 yields significant run time improvements; we achieve 60% for SP and approximately 40% for DP. Thus, users can chose between the DP-SSE3 and SP-SSE3 versions for DNA or protein data in the current RAxML release (v. 7.2.3), depending one their memory and CPU time constraints as well as on the alignment shape.

Future work, will cover a more detailed analysis of the numerical stability on many-taxon datasets as well as work on a GPU version of RAxML.

References

1. Felsenstein, J.: Evolutionary trees from DNA sequences: a maximum likelihood approach. J. Mol. Evol. **17** (1981) 368–376
2. Stamatakis, A.: RAxML-VI-HPC: maximum likelihood-based phylogenetic analyses with thousands of taxa and mixed models. Bioinformatics **22**(21) (2006) 2688–2690
3. Zwickl, D.: Genetic Algorithm Approaches for the Phylogenetic Analysis of Large Biological Sequence Datasets under the Maximum Likelihood Criterion. PhD thesis, University of Texas at Austin (April 2006)
4. Ronquist, F., Huelsenbeck, J.: MrBayes 3: Bayesian phylogenetic inference under mixed models. Bioinformatics **19**(12) (2003) 1572–1574
5. Drummond, A., Rambaut, A.: BEAST: Bayesian evolutionary analysis by sampling trees. BMC Evol Biol **7**(214) (2007) 1471–2148
6. Ott, M., Zola, J., Aluru, S., Stamatakis, A.: Large-scale Maximum Likelihood-based Phylogenetic Analysis on the IBM BlueGene/L. In: Proc. of IEEE/ACM Supercomputing Conference 2007 (SC2007). (2007)
7. Yang, Z.: Maximum likelihood phylogenetic estimation from DNA sequences with variable rates over sites. J. Mol. Evol. **39** (1994) 306–314
8. Kurzak, J., Dongarra, J.: Implementation of mixed precision in solving systems of linear equations on the Cell processor. Concurrency and Computation **19**(10) (2007) 1371
9. Blagojevic, F., Nikolopoulos, D.S., Stamatakis, A., Antonopoulos, C.D.: RAxML-Cell: Parallel Phylogenetic Tree Inference on the Cell Broadband Engine. In: Proc. of International Parallel and Distributed Processing Symposium (IPDPS2007). (2007)
10. Whaley, R., Dongarra, J.: Automatically tuned linear algebra software (ATLAS). In: Proc. Supercomputing. Volume 98. (1998)
11. Stamatakis, A.: Phylogenetic Models of Rate Heterogeneity: A High Performance Computing Perspective. In: Proc. of IPDPS2006. HICOMB Workshop, Proceedings on CD, Rhodos, Greece (April 2006)