# Load Balance in the Phylogenetic Likelihood Kernel

Alexandros Stamatakis
*The Exelixis Lab*
*Dept. of Computer Science*
*Technische Universität München*
*Garching b. München, Germany*
*Email: stamatak@cs.tum.edu*

Michael Ott
*Lehrstuhl für Rechnertechnik und Rechnerorganisation*
*Dept. of Computer Science*
*Technische Universität München*
*Garching b. München, Germany*
*Email: ottmi@cs.tum.edu*

***Abstract*—Recent advances in DNA sequencing techniques have led to an unprecedented accumulation and availability of molecular sequence data that needs to be analyzed. This data explosion in combination with the multi-core revolution also affects the computational kernels for phylogenetic inference (reconstruction of evolutionary trees from molecular sequence data) under the widely-used Maximum Likelihood (ML) model. At present, analyses of so called multi-gene or phylogenomic alignments, i.e., input data sets that comprise concatenated sequence data of several genes, are becoming increasingly popular. Usually such multi-gene analyses are partitioned, i.e., a separate set of likelihood model parameters is estimated for each gene/partition. While the phylogenetic likelihood function exhibits intrinsic fine-grained parallelism, the parallel computation of the likelihood function in such partitioned multi-gene analyses can lead to significant load-balance problems. Here, we describe these problems for the first time, discuss the implications on the design of "classic" ML-based as well as Bayesian search algorithms, and provide an initial solution that yields up to eight-fold improvements in speedup values on AMD Barcelona and Sun x4600 16-core systems for realistic application scenarios.**

## I. Introduction

Emerging parallel multi- and many-core computer architectures pose new challenges for the field of Bioinformatics, since a large number of widely used applications will have to be ported to these systems. In addition, because of the continuous explosive accumulation of sequence data, which is driven by novel wet-lab techniques such as, e.g., pyrosequencing [1] or advances in Expressed Sequence Tag techniques (ESTs [2]), the application of high performance computing methods needs to become an integral part of Bioinformatics. Moreover, there is an increasing gap, which we call the "Bio-Gap" between the speed at which molecular data accumulates and the speed at which computer architectures become faster according to Moore's law (see [3] for a respective plot). Hence, we urgently need to devise appropriate parallelization schemes in order to keep pace with the biological data flood.

Many problems in Bioinformatics such as BLAST searches [4], statistical tests for host–parasite co-evolution [5], or computation of Bootstrap replicates [6] for phylogenetic trees are embarrassingly parallel and can be parallelized at coarse grained level. Nonetheless, they might soon require the introduction of an additional layer of parallelism, i.e., hybrid [5], [7] or multi–grain [8] parallelism to handle constantly growing dataset–sizes and the associated huge memory requirements. Moreover, for large embarrassingly parallel problems, hybrid parallelizations can potentially allow for more efficient exploitation of current computer architectures by achieving super-linear speedups because of increased cache efficiency (see [9]).

Here we focus on the parallelization of the Phylogenetic Likelihood Kernel (PLK [10]) which is among the most important statistical functions in Bioinformatics and forms the computational core of a plethora of programs (IQPNNI [11], PHYML [12], [13], PhyloBayes [14], MrBayes [15], TREEFINDER [16], TREE-PUZZLE [17], [18], GARLI [19], DPRML [20], LEAPHY [21], RAxML [22], [23], PAML [24], PAUP* [25]) that are used to reconstruct evolutionary (phylogenetic) trees from molecular sequence data under statistical models of sequence evolution. The aforementioned programs represent important tools for many biological disciplines such as microbial ecology, virology, conservation biology, paleontology, etc. and have accumulated over 20,000 citations to date.

While finding the optimal Maximum Likelihood (ML) evolutionary tree is NP-hard [26] the other major computational challenge lies in computing the PLK, i.e., the likelihood score on a given tree. The PLK consumes about 85%-98% of total execution time in all of the above programs and hence represents the computational kernel that needs to be optimized and parallelized.

The relatively straight–forward fine–grained parallelism in the PLK scales particularly well on relatively long, in terms of sequence length, input datasets (see, e.g., [27], [28]) and hence fits well to a current trend in systematics, which focuses on the analysis of so-called multi-gene or phylogenomic sequence alignments (see, e.g., [29]–[31] for recent phylogenomic analyses using RAxML). Phylogenomic alignments are typically assembled by concatenating the sequence data of several (typically 10 to over 100) genes for the organisms under study. Such large phylogenomic analyses under the ML model have huge computational resource requirements; in two recent real-world studies with biologists we used 2.25 million CPU hours on a BlueGene/L system and analyzed a dataset that required 89GB of main memory and contained more than 1,000 genes. Thus, the analysis of such large phylogenomic datasets which is also driven by advances in DNA sequencing techniques represents an important computational problem. Because of this trend we focus on exploiting fine–grained loop level parallelism in the PLK for phylogenomic datasets. We describe and study a new load balance problem in the PLK that arose during production runs on the aforementioned large-scale biological datasets. The load imbalance is caused by separately estimating certain parameters (see Section III) of the ML model on a per-gene basis, i.e., by conducting so called partitioned likelihood analyses. Biologically this makes sense, since different genes exhibit distinct evolutionary histories. Evidently, respective sequence data is not available for every gene of the organisms under study. Hence, such phylogenomic alignments tend to be "gappy", i.e., contain relatively large holes in the gene sampling that are filled up by alignment gaps (see [32] for a more detailed description). Moreover, there also exist strong computational arguments for conducting partitioned analyses, since only partitioned analyses can be accelerated by one to two orders of magnitude using a strategy that does not take into account these alignment gaps [32].

We study this load balance problem for partitioned phylogenomic analyses and develop an initial solution using the Pthreads-based development version of RAxML [22] which is a widely used program (over 4,600 downloads from distinct IP addresses; over 25,000 jobs submitted to the RAxML web-servers [33]) for ML-based inference of phylogenetic trees. However, we only consider RAxML as being one possible implementation of the PLK; our findings have implications on the algorithmic design and parallelization of all PLK-based programs (either "classic" ML or Bayesian inference). We tested our approach on several current multi-core architectures with 8 to 16 cores using real-world as well as simulated datasets and achieved up to eight-fold improvements in speedups for partitioned phylogenomic analyses.

The remainder of this paper is organized as follows: In Section II we briefly outline related work on exploiting fine-grained parallelism in the PLK. Thereafter, we describe how the likelihood score is computed on a tree at an abstract level (Section III). In the following Section IV we present an initial solution to improve scalability of partitioned analyses and provide respective performance results in Section V. We conclude with an overview of current and future work.

## II. Related Work

As already mentioned, this paper represents—to the best of our knowledge—the first description and attempt to solve the load balance problem in the Phylogenetic Likelihood Kernel. Hence, we are not aware of any related work that covers the problem we describe here.

Apart from our own work on exploiting loop-level and multi-grain parallelism in RAxML on Graphics Processing Units [34], shared memory systems [9], FPGAs [35], multi-core systems [32], the IBM Cell [8], the SGI Altix [36], the IBM BlueGene/L [27], and for comparing performance of Pthreads *versus* OpenMP *versus* MPI for the PLK [28], there is relatively few related work in this field.

Minh *et al* [7] implemented a hybrid OpenMP/MPI version of IQPNNI that exploits loop-level as well as coarse-grained parallelism. The PKL in GARLI [19] which represents a widely used program for phylogenetic inference has also been parallelized with OpenMP. RAxML was also initially parallelized with OpenMP [9] and then with Pthreads, mainly for software engineering reasons [28]. Finally, a recent proof-of-concept parallelization of the Bayesian program PBPI (Parallel Bayesian Phylogenetic Inference) on the BlueGene/L has been presented by Feng *et al* [37].

However, none of the above programs (GARLI, IQPNNI, PBPI) currently offers the possibility to conduct analyses of partitioned multi-gene datasets with a per-partition estimate of ML model parameters. However, a GARLI version that can handle partitioned analyses is currently under preparation (Derrick Zwickl, personal communication). Currently, MrBayes [15], RAxML, PAML [24], and PAUP* [25] provide this option while PhyloBayes [14] uses a slightly different ML model.

## III. The Phylogenetic Likelihood Kernel

The input of a phylogenomic 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. In order to be able to compute the likelihood on a fixed tree topology one also needs several ML model parameters: the instantaneous nucleotide substitution matrix $Q$ which contains the transition probabilities for time $dt$ between nucleotide (4x4 matrix) or Amino Acids (20x20 matrix) characters, the prior probabilities of observing the nucleotides, e.g., $\pi_A, \pi_C, \pi_G, \pi_T$ for DNA data, which can be determined empirically from the alignment, the $\alpha$ shape parameter that forms part of the $\Gamma$ model [38] of rate heterogeneity that accounts for the fact that different columns in the alignment evolve at different speeds, and finally the $2n-3$ branch lengths.

Given all these parameters, in order to compute the likelihood of a fixed *unrooted* binary tree topology, initially one needs to compute the entries for all internal likelihood arrays (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` for each site $m$ of the input alignment at the specific inner node, bottom-up from the tips towards a virtual root that can be placed into any branch of the tree. Under certain standard model restrictions (time-reversibility of the model) the final likelihood score will be the same regardless of the placement of the virtual root. Once the partial likelihood arrays to the left and right of the virtual root have been computed, the log likelihood score can then be calculated by combining and summing over the entries in the two likelihood arrays (for mathematical details please refer to [10]). An important property of the likelihood function is the assumption, that sites evolve independently, hence all entries `i` of the likelihood arrays, where `i=1...m` can be computed independently. This property represents the main source of fine-grained parallelism in the PLK.

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 $\alpha$ shape parameter, must also be optimized via an ML estimate. For the $Q$ matrix and the $\alpha$ shape parameter the most common approach in "classic" ML implementations consists in using Brent's algorithm [39]. In order to evaluate changes in $Q$ or $\alpha$ 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 likelihood improvement is smaller than some pre-defined $\epsilon$. Note that, if only one branch length is changed, the tree does not need to be completely re-traversed, since the likelihood score is invariant to the placement of the virtual root.

Provided the prolegomena, the computation of the ML score after a change to the tree topology, requires all these parameters to be re-optimized. However, most modern search algorithms (GARLI, RAxML, PHYML), do not re-optimize *all* branch lengths, do not re-compute all partial likelihood arrays, and do not re-estimate all model parameters after a change in tree topology. They rather carry out local optimizations as outlined in Figure 1 in the neighborhood of the tree region that is most affected by the topological change. Hence, we do not need to conduct full tree traversals and to re-compute all inner likelihood arrays, but only execute partial traversals on those likelihood arrays that are affected by the topological change. Typically current algorithms alternate between tree search phases and model optimization phases, i.e., the ML score is improved by changing the topology, re-estimating only a part of the branch lengths, and then by re-estimating the model parameters on the improved topology after several topological moves.

The main bulk of all of the above likelihood computations consists of `for`-loops over the length $m$ of the multiple sequence input alignment, or more precisely over the number

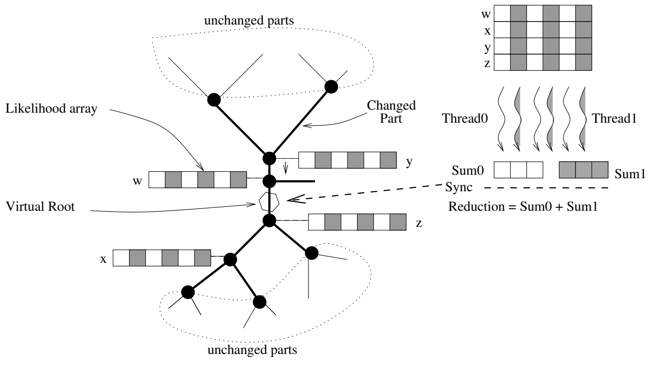

Figure 1. Approximate computation of the maximum likelihood score after a topological change in the tree and general parallelization outline

$m'$ of distinct column patterns in the alignment. Operations like branch length optimization and likelihood score computation, that need to compute a likelihood score, i.e., that require to sum over all $m'$ entries at the current virtual root represent natural synchronization points for the fine-grained parallelization because of the reduction operations that need to be conducted and are outlined in more detail in the next Section.

## IV. Parallelization of Partitioned Analyses

The general parallelization scheme of the Pthreads-version of RAxML is outlined in Figure 1. We use a cyclic distribution of the $m'$ distinct alignment patterns to threads, mainly to allow for better load-balance in phylogenomic datasets that can contain DNA as well as AA (protein) data. The computation of a likelihood vector entry for a protein pattern position requires significantly more floating point operations, since there are 20 instead of 4 likelihood entries to compute.

After a change in tree topology the master thread generates a partial tree traversal list that contains references to the inner likelihood vector arrays that need to be recomputed. Thereafter, it also orchestrates the optimization of the local branch lengths that are most affected by the change. In the tree search phase, both the number of likelihood arrays as well as the number of branches to be optimized is significantly smaller than the respective overall number ($n-2$ inner vectors, $2n-3$ branch lengths).

In the model optimization phase that typically contributes less to overall execution time in RAxML (approximately 20-30%) the tree needs to be fully traversed. Hence, the master thread generates a full tree traversal list, that remains fixed during the model parameter optimization process because the tree topology is not changed. When a parameter has been changed, every worker thread can independently update its fraction of the likelihood array entries for the full tree traversal and the threads only need to be synchronized when the virtual root is reached and the likelihood score is computed. Therefore, every thread conducts a significantly larger fraction of independent work per alignment pattern during the model parameter optimization phase.

In the tree search phase the worker threads will only need to update 3-4 inner likelihood vectors on average. Regardless of the algorithmic state (i.e., tree search or model parameter optimization phase), synchronization is always required between optimizations of distinct branches in the tree.

While the above approach has shown to be highly efficient and scalable for unpartitioned phylogenomic analyses [27], [28], new problems arise for partitioned analyses. As shown in Figure 2 the phylogenomic alignment and the likelihood model can be divided into distinct partitions that correspond to the individual genes. For every partition the $Q$ matrix, $\alpha$ shape parameter, and the branch lengths can be separately optimized. One can also conduct a joint branch length estimate over all partitions, however the aforementioned approach to more efficiently compute likelihood scores on gappy multi-gene alignments requires a per-partition branch length estimate [32]. Because of the great computational potential of this approach ( [32] only represents proof-of-concept work, i.e., we have not yet implemented tree searches under this model) we strongly argue in favor of using per-gene branch length estimates. The load-balance problems arise for the iterative optimization procedures (Brent for $Q$ and $\alpha$, Newton-Raphson for branches) used. Evidently, the number of iterations required to converge will be different for each partition. Hence, the original, relatively straight-forward approach consisted in optimizing parameters for one partition at a time. While this is not that critical for the optimization of $Q$ and $\alpha$ because a full tree traversal and hence more work is conducted by every thread on each of its columns that form part of the current partition, the problem is more severe for branch length optimization. If we conduct branch length optimization, that occurs both during the tree search as well as during the model parameter optimization phase (albeit with a smaller contribution to overall execution time in the latter case) on a per-partition basis, this will significantly increase the synchronization overhead and decrease the amount of work a thread can conduct per branch length optimization cycle that is triggered by the master. In the worst case, which actually occurred in practice on a large phylogenomic dataset using an SGI Altix supercomputer, i.e., many short partitions and a large number of threads, it *can* happen, that there are more threads available than distinct patterns in a specific partition which means that some threads will be idling. This can also affect the performance of the $Q$ and $\alpha$ parameter optimization, even in analyses that use a joint branch length estimate across all partitions.

As an initial solution, we have completely re-designed the iterative optimization procedures in RAxML to conduct computations on the full length of the likelihood vectors for as long as feasible, to provide as much work as available to the worker threads, and reduce the synchronization overhead. The basic idea consists in re-designing the routines that implement the Newton-Raphson and Brent procedures in such a way that they simultaneously optimize all partitions. Because the iterative optimization procedures on every partitions will converge after a variable number of iterations, we need to keep track of the convergence conditions for every partition separately and be cautious to avoid the invocation of the likelihood functions on partitions that have already converged via an appropriate boolean vector. While this approach sounds relatively straight-forward it was a major software engineering challenge due to the code complexity of RAxML that can handle concatenated datasets consisting of morphological, DNA, secondary structure, and protein data, i.e., we needed to design a production-level implementation. Despite the relative simplicity of the proposed approach it already works considerably well, as shown by our experimental results.

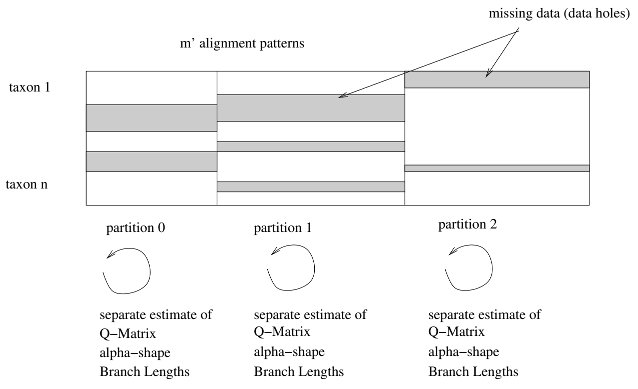
missing data (data holes)
m' alignment patterns
taxon 1
taxon n
partition 0
partition 1
partition 2
separate estimate of Q−Matrix alpha−shape Branch Lengths
separate estimate of Q−Matrix alpha−shape Branch Lengths
separate estimate of Q−Matrix alpha−shape Branch Lengths

Figure 2. A partitioned phylogenomic alignment with per-partition ML model parameter estimate

**Implications for Bayesian Inference:** At the technical level the implementation of the PLK for Bayesian inference programs is significantly easier, because they do not require the iterative optimization routines (Brent, Newton-Raphson) to optimize model parameters. This is conducted via random proposals in the Metropolis-Coupled Markov-Chain Monte-Carlo ($MC^3$) procedure. While the efficiency of the likelihood computations for proposals to change $Q$ or $\alpha$ will not decrease to such a large extent for the same reasons as for "classic" ML, the same problems arise for branch length change proposals. Thus, keeping future fine-grained parallelizations of Bayesian inference programs in mind, that will need to be implemented in order to handle the data flood and because Metropolis-Coupled Bayesian programs have higher memory requirements than ML (we need to assign separate memory space for the inner likelihood vectors for each chain) the proposal mechanisms need to be re-designed. Ideally, the mechanism and underlying statistics should be designed such as to allow for applying simultaneous changes to one of the parameter types across all partitions. Moreover, branch length changes should be simultaneously proposed for all partitions of the same topological connection in order to require the same amount of re-computations of inner likelihood vectors when the virtual root is re-located to the branch to be changed.

## V. Results

**Test Datasets:** To test the revised parallelization of the PKL we used 12 simulated DNA datasets, that were generated with SeqGen [40] (v1.3.2) on seed trees containing 10, 20, 50, and 100 taxa from real-world analyses. For each tree size we generated alignments with a length of 5,000, 20,000, and 50,000 nucleotides. We will refer to these alignments as dXX_YYYY, where XX stands for the number of taxa and YYYY for the number of alignment columns. We ensured that each alignment consists entirely of unique columns, hence $m = m'$. We then generated several partition files (for details see RAxML manual) that divide the respective alignments into $p$ partitions of length 1,000, 5,000, and 10,000 (designated as p1000, p5000, p10000). Note that a partition length of 1,000 columns (nucleotides) roughly corresponds to average gene length.

In addition, we used three real-world phylogenomic alignments: two alignments (r26_21451, r24_16916) of viral protein sequences with 26 and 24 taxa (26 partitions, 21,451 distinct patterns; 20 partitions, 16,916 distinct alignment patterns) and an alignment of 125 mammalian DNA sequences (r125_19839) with 19,839 distinct patterns and 34 partitions. The minimum and maximum partition lengths in terms of distinct alignment column patterns are: 173 and 2,695 for dataset r26_21451 as well as r24_16916 and 148 and 2,705 for r125_19839 respectively.

**Availability:** All test datasets as well as the source code, that can be compiled to yield the new or the old parallelization approach via a respective compiler switch are available for download at http://wwwkramer.in.tum.de/exelixis/software.html.

**Platforms:** We used the following general-purpose multicore architectures to test the new approach: a 4-way AMD Barcelona with a total of 16 cores, 128GB of main memory running at 2.2 GHz; a 8-way Sun x4600 with 16 cores, 64GB, at 2.6GHz; a 2-way Intel Nehalem pre-production system with 8 cores, 12 GB, at 2.933GHz; a 2-way Intel Clovertown with 8 cores, 8GB 2.66 GHz. The code on the AMD Barcelona was compiled with `gcc` version 4.3.2 and on the x4600 with version 4.1.3 respectively using the `-O3 -fomit-frame-pointer -funroll-loops` optimization flags. The code on the Nehalem and Clovertown systems was compiled using `icc` (v11.0.074) with flags `-O3 -xS` and `-O3 -xT` respectively.

**Experimental Setup:** For every possible combination of simulated datasets and corresponding partition schemes (e.g., dataset d10_5000 can not be executed with p10000) we executed 4 distinct analyses: An optimization of ML model parameters (without tree search) on a fixed input tree with joint and per-partition branch length estimates, as well as full ML tree searches (on a fixed input tree for reproducibility) with joint and per-partition branch length estimates. In addition, we measured parallel performance for unpartitioned ML parameter optimization and tree searches on all 12 simulated datasets.

The three real-world datasets where analyzed using the biologically meaningful per-gene partitions provided by our collaborators. We once again conducted model parameter optimization and tree search runs using a joint and a per-partition estimate of branch lengths.

We executed all of the above runs on 1, 8 and 16 cores where available (Sun x4600, AMD Barcelona).

**Results:** As expected (because of the significantly more favorable synchronization to computation ratio for joint branch length optimization) the run time differences between the old per-partition parallelization approach (oldPAR) and the new simultaneous parallelization approach (newPAR) where insignificant for analyses using a joint branch length estimate over all partitions. The average execution time improvement amounts to approximately 5%, both for full tree searches as well as for stand-alone model parameter optimization on a fixed tree (results not shown). Hence, we focus on the results and speedup improvements obtained for analyses with a per-partition branch length estimate. As already mentioned, the optimization of ML model parameters on a fixed tree (i.e., no tree search is performed), even with a per-partition branch length estimate, exhibits more computations per synchronization event, since the entire tree needs to be traversed to re-compute the likelihood score after changes to $Q$ and $\alpha$, i.e., there is significantly more computational work per alignment column. Therefore, the average execution time improvements range between 5% and 10% for model parameter optimization on a fixed tree depending on the dataset and platform used (data not shown).

Thus, also the practically most relevant case is that of full ML tree searches with per-partition branch length estimates. Evidently, the number and length of partitions in a dataset will have direct impact on the performance improvements achieved by newPAR, i.e., the more and the shorter the partitions are, the better the performance of newPAR versus oldPAR will become. We present the realistic (given the partition length that corresponds to the typical length of an average gene) worst-case scenario for oldPAR which at the same time is the best-case scenario, in terms of speedup over oldPAR, for newPAR.

In Figure 3 we provide absolute execution times for the four test systems (Nehalem, Clovertown, Barcelona, Sun x4600) for the sequential, the oldPAR, and the newPAR versions on 1, 8, and 16 threads (where applicable) for a full ML tree search on dataset d50_50000 with 50 partitions of 1,000 columns each. In Figure 4 we provide an analogous plot for dataset d100_50000 that is also partitioned into 50 partitions of 1,000 alignment columns. In Figure 5 we show that execution times on the real-world mammalian DNA dataset with 34 partitions of variable length (minimum length: 148; maximum length: 2,705) improve to a similar degree as for our simulated datasets. Note that the speedups were smaller (around 5-10%) on the two protein datasets (data not shown). While they are similar in number and length of the partitions to the mammalian dataset, the computation of the likelihood score for protein sequences that is based on a 20x20 instead of a 4x4 nucleotide substitution matrix requires a significantly higher amount (roughly by a factor of 20x20/4x4=25) of floating point operations per column. Hence, the load balance problem is less prevalent for protein data. Nonetheless, Figures 3, 4 and 5 clearly show that *firstly* there is a significant load balance problem in the PKL for analyses of realistically-dimensioned and partitioned DNA datasets with a per-partition branch length estimate, and *secondly* that up to eight-fold improvements in parallel efficiency can be achieved on current multi-core architectures using the novel approach we propose here. Moreover, the parallel slowdown observed on 16 cores (AMD Barcelona, Sun x4600) for oldPAR compared to run times on 8 cores can be alleviated by our newPAR method. Moreover, as underlined by Figure 6 on dataset d50_50000 (again with p1000) for the Intel Nehalem, the speedup achieved for the partitioned analysis with newPAR is nearly as good as the speedup obtained for a completely unpartitioned analysis, despite the load imbalance problem.

The absolute run-times for the different processor architectures are consistent with previous observations [28]: performance on Intel processors for sequential program runs is significantly better than performance on AMD processors. However, with 8 threads the AMD processors are on par with the Intel Clovertown. This is due to the fact that all 8 cores of the Clovertown system share a common front-side bus for main memory accesses, whereas the AMD NUMA architecture provides a higher aggregated memory bandwidth for parallel program runs. Because RAxML is memory-bound, the memory bandwidth available to each thread heavily influences execution times. For the same reason the Intel Nehalem system clearly outperforms all other systems, since it is also based on a NUMA (QuickPath Interconnect) architecture. Because of a memory bandwidth of approximately 30GB per second *and* per processor, the sequential runtime on the Nehalem is almost 40% lower than on the Clovertown.

## VI. Conclusion and Future Work

We have provided the first description and analysis of a practically highly relevant load balance problem in the Phylogenetic Likelihood Kernel that has implications on the future design and parallelization of "classic" ML as well as Bayesian methods for the reconstruction of evolutionary trees from molecular sequence data. We also provide an initial solution and production-level implementation for this load-balance problem that increases parallel efficiency of the PLK by a factor of two to eight for large-scale phylogenomic analyses on several current multi-core architectures. The focus is on multi-core architectures, since they already form part of the standard analysis environment of system-

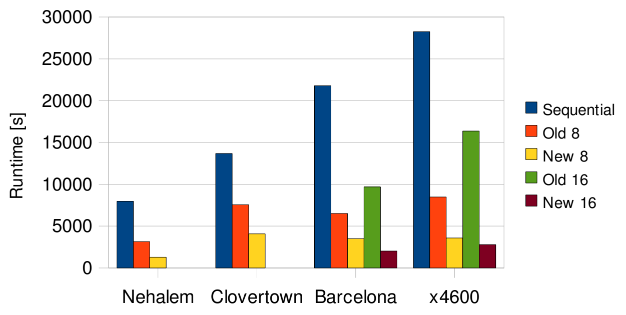

Figure 3. Sequential and parallel execution times for dataset d50_50000 with 50 partitions of 1,000 columns each on Nehalem, Clovertown, Barcelona, and Sun x4600 multi-core systems

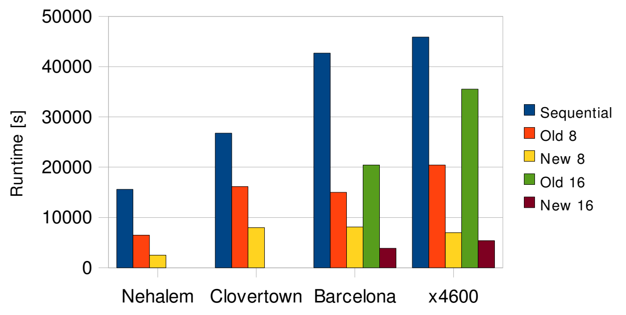

Figure 4. Sequential and parallel execution times for dataset d100_50000 with 50 partitions of 1,000 columns each on Nehalem, Clovertown, Barcelona, and Sun x4600 multi-core systems

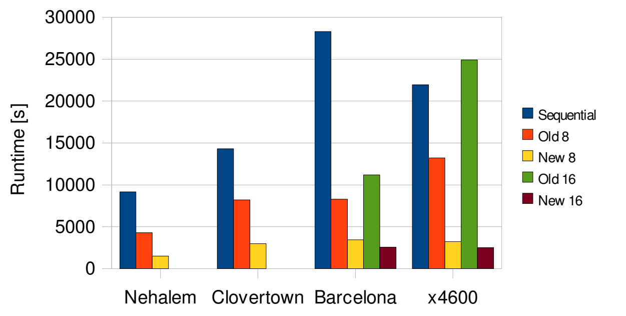

Figure 5. Sequential and parallel execution times for the real-world mammalian dataset r125_19839 with 34 partitions of variable length on Nehalem, Clovertown, Barcelona, and Sun x4600 multi-core systems

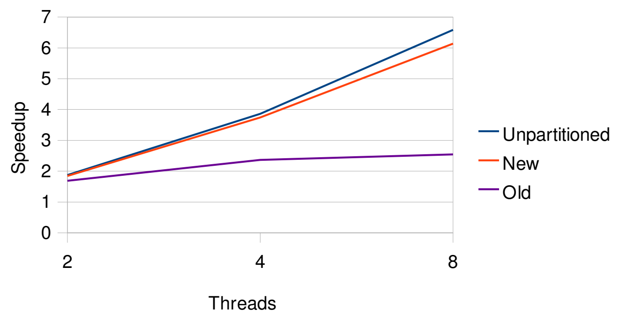

Figure 6. Speedup comparisons for dataset d50_50000 on the Intel Nehalem for an unpartitioned analysis as well as for the old parallelization approach and the new parallelization approach for partitioned analyses with 50 partitions of 1,000 columns each

atists. While the increase in parallel efficiency achieved via simultaneous iterative optimization of ML model parameters across all partitions is significant, the speedups are still slightly lower than for unpartitioned analyses that partially achieve substantial super-linear speedups [28]. We will hence further investigate this important problem, try to identify additional performance bottlenecks and assess scalability on supercomputers with several hundred cores. Finally, we also intend to implement tree searches under the computationally improved likelihood model for gappy phylogenomic alignments [32].

## Acknowledgments

This work is funded under the auspices of the Emmy-Noether program by the German Science Foundation (DFG) and by the Bavarian Competence Network for Technical and Scientific High Performance Computing (KONWIHR). The authors would like to thank N. Poulakakis, O.R.P. Bininda-Emonds, J.A. Jehle, Y. Wang, A.M.M. Abd Alla, and J.M.Vlak for providing the real-world phylogenomic alignments used in this study.

## References

[1] M. Ronaghi, "Pyrosequencing Sheds Light on DNA Sequencing," *Genome Research*, vol. 11, no. 1, pp. 3–11, 2001.

[2] M. Adams, J. Kelley, J. Gocayne, M. Dubnick, M. Polymeropoulos, H. Xiao, C. Merril, A. Wu, B. Olde, R. Moreno *et al.*, "Complementary DNA sequencing: expressed sequence tags and human genome project," *Science*, vol. 252, no. 5013, pp. 1651–1656, 1991.

[3] N. Goldman and Z. Yang, "Introduction. Statistical and computational challenges in molecular phylogenetics and evolution," *Phil. Trans. R. Soc. series B, Biol. Sci.*, vol. 363, no. 1512, p. 3889, 2008.

[4] A. Darling, L. Carey, and W. Feng, "The Design, Implementation, and Evaluation of mpiBLAST," *Proceedings of ClusterWorld*, vol. 2003, 2003.

[5] A. Stamatakis, A. Auch, J. Meier-Kolthoff, and M. Göker, "AxPcoords & Parallel AxParafit: Statistical Co-Phylogenetic Analyses on Thousands of Taxa," *BMC Bioinformatics*, vol. 8, 2007.

[6] J. Felsenstein, "Confidence Limits on Phylogenies: An Approach Using the Bootstrap," *Evolution*, vol. 39, no. 4, pp. 783–791, 1985.

[7] B. Minh, L. Vinh, H. Schmidt, and A. Haeseler, "Large maximum likelihood trees," in *Proc. of the NIC Symposium 2006*, 2006, pp. 357–365.

[8] F. Blagojevic, D. Nikolopoulos, A. Stamatakis, and C. Antonopoulos, "Dynamic Multigrain Parallelization on the Cell Broadband Engine," in *Proc. of PPoPP 2007*, San Jose, CA, March 2007.

[9] A. Stamatakis, M. Ott, and T. Ludwig, "RAxML-OMP: An Efficient Program for Phylogenetic Inference on SMPs." *PaCT*, pp. 288–302, 2005.

[10] J. Felsenstein, "Evolutionary trees from DNA sequences: a maximum likelihood approach," *J. Mol. Evol.*, vol. 17, pp. 368–376, 1981.

[11] B. Minh, L. Vinh, A. Haeseler, and H. Schmidt, "pIQPNNI: parallel reconstruction of large maximum likelihood phylogenies," *Bioinformatics*, vol. 21, no. 19, pp. 3794–3796, 2005.

[12] S. Guindon and O. Gascuel, "A simple, fast, and accurate algorithm to estimate large phylogenies by maximum likelihood." *Syst. Biol.*, vol. 52, no. 5, pp. 696–704, 2003.

[13] W. Hordijk and O. Gascuel, "Improving the efficiency of SPR moves in phylogenetic tree search methods based on maximum likelihood," *Bioinformatics*, vol. 21, no. 24, pp. 4338–4347, 2005.

[14] N. Lartillot, S. Blanquart, and T. Lepage, "PhyloBayes. v2. 3," 2007.

[15] F. Ronquist and J. Huelsenbeck, "MrBayes 3: Bayesian phylogenetic inference under mixed models," *Bioinformatics*, vol. 19, no. 12, pp. 1572–1574, 2003.

[16] G. Jobb, A. Haeseler, and K. Strimmer, "TREEFINDER: A powerful graphical analysis environment for molecular phylogenetics," *BMC Evol. Biol.*, vol. 4, 2004.

[17] H. Schmidt, K. Strimmer, M. Vingron, and A. Haeseler, "TREE-PUZZLE: maximum likelihood phylogenetic analysis using quartets and parallel computing," *Bioinformatics*, vol. 18, pp. 502–504, 2002.

[18] K. Strimmer and A. Haeseler, "Quartet puzzling: A quartet maximum likelihood method for reconstructing tree topologies," *Mol. Biol. Evol.*, vol. 13, pp. 964–969, 1996.

[19] D. Zwickl, "Genetic Algorithm Approaches for the Phylogenetic Analysis of Large Biological Sequence Datasets under the Maximum Likelihood Criterion," Ph.D. dissertation, University of Texas at Austin, April 2006.

[20] T. Keane, T. Naughton, S. Travers, J. McInerney, and G. McCormack, "DPRml: Distributed Phylogeny Reconstruction by Maximum Likelihood," *Bioinformatics*, vol. 21, no. 7, pp. 969–974, 2005.

[21] S. Whelan, "New Approaches to Phylogenetic Tree Search and Their Application to Large Numbers of Protein Alignments," *Syst. Biol.*, vol. 56, no. 5, pp. 727–740, 2007.

[22] A. Stamatakis, "RAxML-VI-HPC: maximum likelihood-based phylogenetic analyses with thousands of taxa and mixed models," *Bioinformatics*, vol. 22, no. 21, pp. 2688–2690, 2006.

[23] A. Stamatakis, T. Ludwig, and H. Meier, "RAxML-III: A Fast Program for Maximum Likelihood-based Inference of Large Phylogenetic Trees," *Bioinformatics*, vol. 21(4), pp. 456–463, 2005.

[24] Z. Yang, "PAML 4: Phylogenetic Analysis by Maximum Likelihood," *Molecular Biology and Evolution*, vol. 24, no. 8, p. 1586, 2007.

[25] D. L. Swofford, *PAUP$^*$: Phylogenetic analysis using parsimony ($^*$ and other methods), version 4.0b10.* Sinauer Associates, 2002.

[26] S. Roch, "A Short Proof that Phylogenetic Tree Reconstruction by Maximum Likelihood Is Hard," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, pp. 92–94, 2006.

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

[28] A. Stamatakis and M. Ott, "Exploiting Fine-Grained Parallelism in the Phylogenetic Likelihood Function with MPI, Pthreads, and OpenMP: A Performance Study." in *PRIB*, ser. Lecture Notes in Computer Science, M. Chetty, A. Ngom, and S. Ahmad, Eds., vol. 5265. Springer, 2008, pp. 424–435.

[29] C. Dunn, A. Hejnol, D. Matus, K. Pang, W. Browne, S. Smith, E. Seaver, G. Rouse, M. Obst, G. Edgecombe, M. Sorensen, S. Haddock, A. Schmidt-Rhaesa, A. Okusu, R. Kristensen, W. Wheeler, M. Martindale, and G. Giribet, "Broad phylogenomic sampling improves resolution of the animal tree of life," *Nature*, vol. 452, no. 7188, pp. 745–749, 2008.

[30] J. Hackett, H. Yoon, S. Li, A. Reyes-Prieto, S. Rummele, and D. Bhattacharya, "Phylogenomic Analysis Supports the Monophyly of Cryptophytes and Haptophytes and the Association of Rhizaria with Chromalveolates," *Molecular Biology and Evolution*, vol. 24, no. 8, p. 1702, 2007.

[31] F. Burki, K. Shalchian-Tabrizi, M. Minge, Å. Skjæveland, S. Nikolaev, K. Jakobsen, and J. Pawlowski, "Phylogenomics Reshuffles the Eukaryotic Supergroups," *PLoS ONE*, vol. 2, no. 8, 2007.

[32] A. Stamatakis and M. Ott, "Efficient computation of the phylogenetic likelihood function on multi-gene alignments and multi-core architectures." *Phil. Trans. R. Soc. series B, Biol. Sci.*, vol. 363, pp. 3977–3984, 2008.

[33] A. Stamatakis, P. Hoover, and J. Rougemont, "A Rapid Bootstrap Algorithm for the RAxML Web Servers," *Syst. Biol.*, vol. 57, no. 5, pp. 758–771, 2008.

[34] M. Charalambous, P. Trancoso, and A. Stamatakis, "Initial Experiences Porting a Bioinformatics Application to a Graphics Processor," in *Proc. of the 10th Panhellenic Conference on Informatics (PCI 2005)*, 2005, pp. 415–425.

[35] N. Alachiotis, E. Sotiriades, A. Dollas, and A. Stamatakis, "Exploring FPGAs for Accelerating the Phylogenetic Likelihood Function," in *Proc. of HICOMB2009*, 2009, accepted for publication.

[36] M. Ott, J. Zola, S. Aluru, A. Johnson, D. Janies, and A. Stamatakis, "Large-scale Phylogenetic Analysis on Current HPC Architectures," *Scientific Programming*, vol. 16, no. 2-3, pp. 255–270, 2008.

[37] X. Feng, K. Cameron, C. Sosa, and B. Smith, "Building the Tree of Life on Terascale Systems," in *Proc. of International Parallel and Distributed Processing Symposium (IPDPS2007)*, 2007.

[38] Z. Yang, "Maximum likelihood phylogenetic estimation from DNA sequences with variable rates over sites," *J. Mol. Evol.*, vol. 39, pp. 306–314, 1994.

[39] R. Brent, *Algorithms for Minimization without Derivatives*. Prentice Hall, 1973.

[40] A. Rambaut and N. Grass, "Seq-Gen: an application for the Monte Carlo simulation of DNA sequence evolution along phylogenetic trees," *Bioinformatics*, vol. 13, no. 3, pp. 235–238, 1997.