Boosting the performance of Bayesian divergence time estimation with the Phylogenetic Likelihood Library

D. Darriba[1,*¶], A.J. Aberer[†], T. Flouri[2,†], T.A. Heath[3,‡], F. Izquierdo-Carrasco[4,†] and A. Stamatakis[†§]
[*]*Department of Biochemistry, Genetics and Immunology*
University of Vigo, Vigo, Spain
Email: ddarriba@uvigo.es
[†]*Heidelberg Institute for Theoretical Studies*
Heidelberg, Germany
Email: {Tomas.Flouri, Fernando.Izquierdo, Alexandros.Stamatakis}@h-its.org
[‡]*Department of Integrative Biology, University of California, Berkeley*
Email: tracyh@berkeley.edu
[§]*Karlsruhe Institute of Technology*
Institute for Theoretical Informatics
Postfach 6980, 76128 Karlsruhe, Germany
[¶]*Department of Electronics and Systems*
University of A Coruña
A Coruña, Spain

***Abstract*—We present a substantially improved and parallelized version of DPPDiv, a software tool for estimating species divergence times and lineage-specific substitution rates on a fixed tree topology. The improvement is achieved by integrating the DPPDiv code with the Phylogenetic Likelihood Library (PLL), a fast, optimized, and parallelized collection of functions for conducting likelihood computations on phylogenetic trees. We show that, integrating the PLL into a likelihood-based application is straight-forward since it took the first author (DD) a programming effort of only one month, without having prior knowledge of DPPDiv, nor the PLL. We achieve sequential speedups that range between a factor of two to three and near-optimal parallel speedups up to 48 threads on sufficiently large datasets. Hence, with a programming effort of one month, we were able to improve DPPDiv's time-to-solution on parallel systems by two orders of magnitude and also to substantially improve its ability to infer divergence times on large-scale datasets.**

***Keywords*-phylogenetics; parallel computing; divergence time estimation; programming effort; phylogenetic likelihood**

I. Introduction

Bayesian methods for phylogenetic inference are dominated by phylogenetic likelihood computations that are used to effectively sample the joint posterior distribution of evolutionary parameters and tree topology. Hence, Bayesian methods require efficient, optimized, and parallelized functions for calculating the likelihood of the data conditional on a tree topology, branch lengths, and model parameters.

One such Bayesian tool is DPPDiv [1], [2]. Given a fixed tree topology, it estimates branch-specific substitution rates and species divergence times under a nonparametric mixture model. More specifically, DPPDiv uses a Dirichlet process prior (DPP) to model variation of substitution rates across the branches of the tree [1]. The Dirichlet process is useful for modeling mixtures since it assumes that data elements (branches of the tree) can be categorized into specific parameter classes [3], [4]. To model variable nucleotide substitution rates among branches, the DPP assumes that the branches of a tree form part of distinct substitution rate categories. Under this model, the number of rate classes, and the assignment of branches to those classes are treated as random variables. Special cases of this model are (i) the so-called global molecular clock that only has one single rate category [5], (ii) the local molecular clock model, where closely related lineages (subtrees) share the same substitution rate [6], [7], and (iii) the independent rates model, where the rate for each lineage is independently drawn from an underlying parametric distribution [8], [9]. In addition to the above models, using a DPP allows to model scenarios where distantly related branches located in distant subtrees evolve under the same rate of evolution. In evolutionary biology, accurate estimates of lineage divergence times are important and widely-used for investigating biological processes such as historical biogeography, species diversification, and rates of continuous trait (morphological properties) evolution. Analyses under the DPP yield robust estimates of branch rates and divergence times for simulated data [1]. The increased flexibility of this model makes it applicable for analyzing large phylogenetic trees.

[1]Supported by MSI TIN2010-16735 & XG CN2012/211.
[2]Supported by DFG project STA-860/4.
[3]Supported by NSF DBI-0805631, NIH GM-069801 & GM-086887.
[4]Supported by DFG project STA-860/3.

DPPDiv uses Markov chain Monte Carlo (MCMC) to sample the posterior densities of the model parameters and to estimate lineage-specific substitution rates as well as node ages. The proposal mechanism for updating the assignment of rates to branches and to sample the rate parameter values associated with each rate-category for the DPP relies on Gibbs sampling (Algorithm 8 in [10]). Each time the Markov chain proposes a branch rate change, the Phylogenetic Likelihood Function (PLF) is invoked. The PLF needs to be computed for each branch in the tree and for each rate category. In fact, DPPDiv spends over 90% of overall execution time for PLF calculations. The computational cost of the PLF-based proposal mechanisms limits the scalability of DPPDiv as the number of tips (branches) and/or number of sites increases. Thus, without an optimized and parallelized PLF implementation, divergence time analyses under the Dirichlet process model in DPPDiv is not feasible on large-scale datasets.

To improve performance and scalability of DPPDiv, we optimized the compute-intensive PLF part of DPPDiv. The code was optimized by replacing the native PLF implementation by corresponding functions from the *Phylogenetic Likelihood Library* (PLL) which we are currently developing. Our intention is to show that (i) it requires a relatively low programming effort to integrate the PLL with a PLF-based program and (ii) that substantial sequential and parallel speedups can be achieved by using the PLL which is based on the highly efficient PLF implementation in RAxML [11]. Combining the optimized serial PLL implementation with the parallel version of the PLL, we obtained performance improvements with respect to the original DPPDiv implementation ranging between a factor of 2 and a factor of 357

The remainder of this paper is organized as follows: In Section II, we briefly review related work on PLF libraries and cover some state-of-the-art implementations for divergence time estimation. Then, we provide an overview of the PLL in Section III. In the following Section IV we describe how we integrated the PLL with DPPDiv. Thereafter, we demonstrate sequential and parallel speedups obtained by integrating PLL with DPPDiv (Section V). We conclude in Section VI and address directions of future work.

II. Related work

A relatively large number of methods for estimating lineage divergence times has already been proposed. For large datasets, there exist several fast (not based on the PLF) methods for obtaining estimates of node ages, such as non-parametric rate smoothing or penalized likelihood [12], [13], [14], distance-based least-squared approaches [15], or methods that use the relative rates between sister lineages [16]. Although these methods are fast and thus capable of estimating divergence times on large datasets, they do not rely on the PLF and therefore lack the inherent advantages of Bayesian inference methods.

Estimating divergence times in a Bayesian framework provides a natural way for accommodating and quantifying uncertainty in the estimates of phylogenetic parameters under a wide range of complex models for rate variation and lineage diversification [17], [18], [19], [20], [21], [8]. Moreover, Bayesian methods (because of their intrinsic flexibility) allow for straight-forward integration of geological information–including fossil data–to calibrate node ages in the tree using units of absolute time. This is achieved by deploying so-called calibration prior densities [22], [23], [2]. Because of these methodological advantages and despite their high computational cost, Bayesian methods represent the most widely-used approach to estimate the times of species divergence events in biological studies (e.g. [24], [25], [26], [27], [28])

Bayesian methods for divergence time estimation are implemented in several tools [21], [29], [30], [31], [32], [1], [33]. Because of the computational burden of Bayesian inference using MCMC in conjunction with the PLF, some implementations use computational shortcuts to approximate the PLF via a multivariate normal distribution [18], [21], [32], [34]. These approaches provide reasonably accurate divergence time estimates given that the assumptions of the rate variation model are not seriously violated [34]. Because of this limitation, it is thus preferable to calculate the exact likelihood score of the sequences given a set of branch lengths. Furthermore, in conjunction with hardware advances, appropriately adapted efficient implementations of the PLF allow for accurate divergence time estimation under more complex and parameter-rich models.

Divergence time estimation algorithms as implemented in the popular Bayesian software programs BEAST [31] and MrBayes 3.2 [33] leverage the efficient likelihood functions available in the BEAGLE library [35]. However, the BEAGLE library has some shortcomings in comparison to the PLL. Firstly, it does not offer support for 256-bit wide x86 AVX vector intrinsics, nor does it implement a fine-grain MPI parallelization of the PLF, thereby only allowing for exploiting intra-node shared-memory parallelism. This is a limitation when analyzing very large datasets that may require more RAM to compute the PLF than available on a single node (see [36]). In addition, BEAGLE does not offer the capability to conduct partitioned analyses where different (independent) sets of parameters are sampled for distinct sites (columns) of the input alignment. As a consequence, it does also not implement appropriate load balancing mechanisms [37], [38] to improve the parallel efficiency of partitioned analyses. Finally, BEAGLE does not offer some of the advanced memory saving techniques [39], [40] that form part of the PLL. Excessive memory requirements can constitute a limiting factor for large-scale PLF-based analyses.

III. Phylogenetic Likelihood Library

The design and implementation of reusable software components in the form of software libraries has substantially contributed to the rapid development and deployment of software tools in the field of bioinformatics.

The advantage of such libraries is that bugs in the code can be reduced to a minimum because of a large and active user community. Moreover, libraries allow users to instantly take advantage of low-level code optimizations and thereby fully exploit the computational power of the underlying hardware without having to re-invent the wheel. Thus, researchers become more productive because they can focus on developing new algorithms and models, instead of tuning and parallelizing fundamental kernel functions.

The PLL is a parallelized and highly optimized software library for the PLF that has been derived from the corresponding PLF implementation in RAxML-Light [36], a tool for inference of large phylogenies under maximum likelihood [41].

The PLL prototype version implements functions for computing conditional likelihood arrays and overall log likelihood scores on phylogenetic trees for DNA and protein sequence data. Moreover, it also offers functions for optimizing branch lengths and other model parameters that are required for building maximum likelihood codes. It uses manually tuned functions that rely on SSE3 and AVX vector intrinsics. Moreover, it offers a fine-grained parallelization of the PLF that relies either on PThreads or MPI (Message Passing Interface) for exploiting inter-node parallelism. It also allows for conducting partitioned analyses and uses the aforementioned load balancing techniques [37], [38] to improve parallel efficiency. We are currently also developing a GPU version of the PLL.

IV. Integration with DPPDiv

The integration of the PLL into DPPDiv was relatively straight-forward. In terms of programming effort, it took the first author of this paper (DD) approximately one month to fully complete the integration. This was achieved without prior knowledge of DPPDiv or the PLL. Given these circumstances, the effort in terms of man hours that were spent is very low, particularly given the speedups in computation gained.

In Figure 1, we provide a top-level view of the Markov chain Monte Carlo (MCMC) proposal implementation in DPPDiv. Note that, in DPPDiv the tree topology remains fixed and that either the number or the values of model parameters are being sampled/changed by a proposal. Thus, to decide whether to reject or accept a proposal, we need to re-compute the log likelihood score on the tree for the altered models parameters. For this reason, we only use the respective subset of PLL functions to compute conditional likelihood arrays and calculate the overall log likelihood score at the root of the tree.

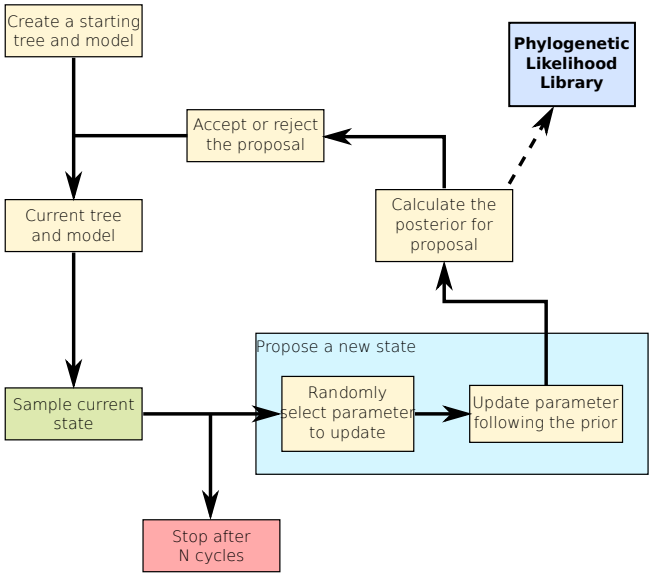

Figure 1. Outline of the Markov Chain Monte Carlo (MCMC) implementation used in DPPDiv.

Our main design criterion for integrating the PLL into DPPDiv was to obtain optimal performance with the least amount of programming effort. For this reason we directly used the tree and model data structures in the PLL. In order to circumvent complicated, and potentially error-prone, modifications to the way DPPDiv updates model parameters in its native, rooted tree data structure, we designed a one-to-one mapping of the DPPDiv tree data structure to the unrooted PLL tree data structure (see Figure 2). Note that, exchanging tree data structures between an application and the PLL currently represents a challenging software engineering issue for the library. The BEAGLE library is completely tree-agnostic and defers the responsibility of designing a tree data structure to the application programmer. We opted for a different approach in PLL, because the availability of such a data structure in the library allows for rapid prototyping and facilitates the use of the library.

The tip nodes (n_0 to n_{N-1}, where N is the number of taxa) and inner nodes (n_N to n_{2N-3}), excluding the root node, of the DPPDiv tree data structure are mapped to the unrooted PLL tree data structure via a bijective function. This bijective function guarantees that each node in the PLL tree data structure is connected to the same neighbors as the corresponding tree in DPPDiv, once again, excluding the root node. Branch lengths are linked in such a way that they are guaranteed to connect corresponding nodes in the respective tree data structures.

The root node used in DPPDiv is represented by placing a virtual root into the unrooted PLL tree data structure. the virtual root is located between the two children of the

root node. The length of the branch on which the virtual root is located is simply the sum of the two branch lengths that lead from the root to the children in the DPPDiv tree representation.

Given this tree mapping from DPPDiv to the PLL, we can now directly use the PLL likelihood function implementation without any additional modifications to the DPPDiv source code.

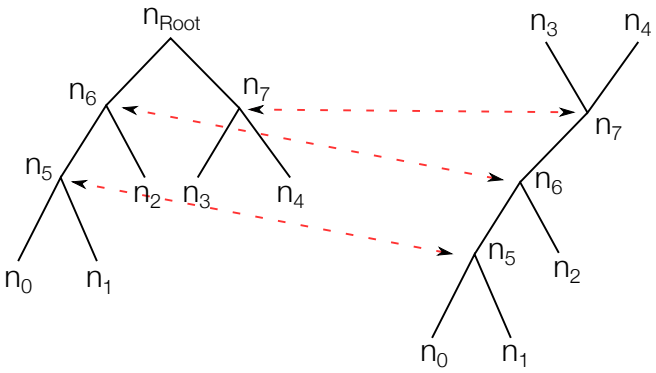

Figure 2. Mapping of the rooted native tree representation in DPPDiv to the unrooted internal tree representation of the PLL.

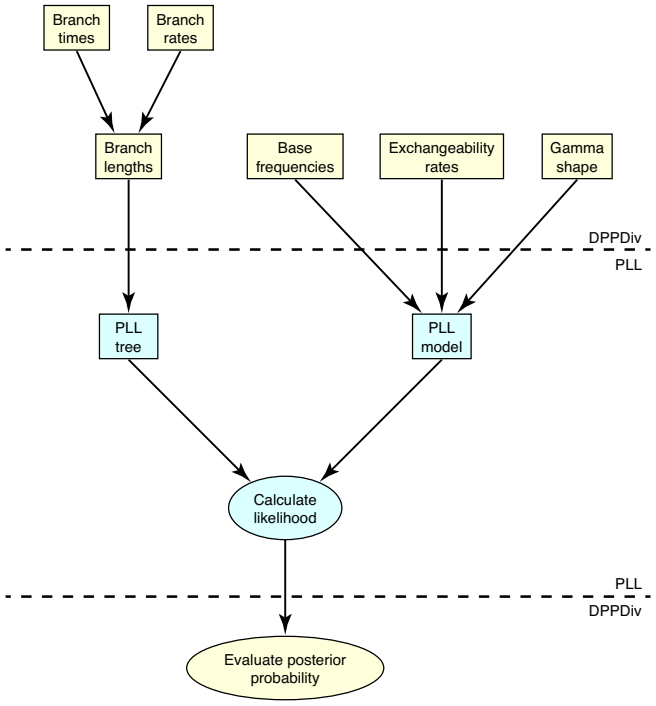

Figure 3. Outline of the interaction between DPPDiv and the PLL for evaluating a proposal. This Figure also shows the different types of model parameters that can be changed by a DPPDiv proposal.

The model parameters (e.g., branch times and rates, see Figure 3) that are changed/proposed by DPPDiv can be mapped to corresponding PLL data structures by using the appropriate PLL interface functions in a straight-forward manner. Once these functions for updating model parameters in the PLL and subsequently re-computing the overall log likelihood using the PLL have been integrated, no further changes to the MCMC proposal mechanism in DPPDiv are required. The proposal mechanism initially alters one of the model parameters shown in Figure 3. We then invoke a function to change this model parameter accordingly in the PLL instance. Subsequently, we use the PLL to compute the log likelihood score of the tree for the changed parameter. The log likelihood score is then returned to DPPDiv so that the posterior probability of the proposal can be computed.

For DPPDiv, the execution of the PLF evaluation by the PLL represents a black box. It is hence entirely obfuscated to DPPDiv, whether the likelihood function is executed sequentially, using SSE3/AVX intrinsics, or in parallel using either PThreads or MPI. All of the above technical issues, including data distribution to threads or processes are hidden in the library. The user only needs to decide which type of intrinsics (SSE3 versus AVX) and which parallel implementation (MPI versus PThreads) he wants to compile and use. The user will also have to specify how many threads or processes to use.

Apart from improving performance (see below) the integration of the PLL also enhances the numerical stability of DPPDiv. Unlike the native DPPDiv PLF implementation, the PLL implementation uses a likelihood scaling technique that prevents numerical underflow. Numerical underflow in likelihood computations (essentially we are multiplying a large number of probabilities with each other in likelihood computations) needs to be handled by appropriate techniques (for details see [42]), in particular for trees where the number of taxa is larger than approximately 100. Thus, better numerical stability comes for free with the PLL integration.

V. Experimental Setup and Results

To evaluate the performance of the PLL-based versions of DPPDiv we measured execution times using a fixed random number seed to generate reproducible results. We used 2 biological datasets [43] and generated several simulated datasets using the APE package [44] for generating random non-ultrametric tree with branches according to a exponential with mean 1/10=0.1 and *seq-gen* [45] to simulate sequence data on these trees and GTR+G models. Hence, a set of DNA datasets with distinct numbers of taxa, sites, and unique site patterns were available for testing which are summarized in Table I. Note that, the number of unique site patterns is more relevant for quantifying performance, since identical alignments sites can and are compressed into site patterns prior to PLF calculations by DPPDiv and the PLL.

We used two shared-memory multi-core systems for testing: a Sandy Bridge node with 32GB RAM and 2 Intel

Xeon E5-2630 hexa-core Sandy Bridge processors (a total of 12 cores) and Hyperthreading disabled, and a Magny-Cours node with 128GB of RAM and 4 AMD Opteron 6174 12-core processors (a total of 48 cores). We compiled 4 PLL instances with SSE3 and AVX intrinsics, PThreads and MPI using gcc v4.7.0. AVX intrinsics are only available on the Sandy Bridge node. Both nodes run Linux Red Hat 4.4.6-4.

For testing the MPI performance of the PLL we used several nodes (e.g, an execution with 96 cores on the Sandy Bridge cluster used 8 nodes and 12 cores per node). The nodes are interconnected through an Infiniband QDR (8 Gbit/s) interconnect.

Table I
ALIGNMENTS USED TO BENCHMARK DPPDIV PERFORMANCE. IN COLUMN *Size*, *N* INDICATES THE NUMBER OF SEQUENCES, *L* THE LENGTH OF THE ALIGNMENT AND *U.P.* THE NUMBER OF UNIQUE SITE PATTERNS. *Num. cycles* INDICATES THE NUMBER OF MCMC ITERATIONS EXECUTED FOR EACH ALIGNMENT. ALIGNMENTS 1 THROUGH 5 ARE SIMULATED ALIGNMENTS, WHILE 6 AND 7 ARE REAL-WORLD DATASETS. COLUMN *Seq.exec.time* SHOWS THE EXECUTION TIME OF THE ORIGINAL DPPDIV IMPLEMENTATION, IN HOURS.

Data set Abbreviation	Size N	Size L	Size U.P.	Seq.exec. time (h)	Num. cycles
Align1	25	7,200	3,034	1.37	100,000
Align2	82	5,167	4,763	14.69	100,000
Align3	118	924	924	3.55	100,000
Align4	10	146,875	25,014	2.85	100,000
Align5	10	333,170	43,100	4.50	100,000
Align6	125	29,149	19,436	17.47	10,000
Align7	169	35,603	29,064	20.03	2,000

We define the parallel speedup as T_1/T_n, where T_1 is the time required by the serial execution of the PLL implementation, and T_n is the respective execution time with n PThreads or MPI processes.

A. Sequential SSE3 and AVX Performance

In Figure 4 we present the sequential execution time improvements of the PLL-based SSE3 and AVX versions over the original DPPDiv implementation. The PLL likelihood implementation performs particularly well on trees with a large number of taxa. This may also be associated with the fact that the PLL implements a numerical scaling technique to prevent numerical underflow and is hence numerically more stable on these large datasets. We measured the denormalized floating point exceptions in both the original DPPDiv implementation and the PLL-based using the Alignment7 dataset. This test shows that the PLL-based implementation does not throw any of these exception, while the original DPPDiv implementation throws 20×10^9 floating point exceptions. We assume that this disproportional slowdown is associated with the original DPPDiv implementation generating so many denormalized floating point values (and hence exceptions) because it lacks a numerical scaling procedure. Hence, we achieved the best

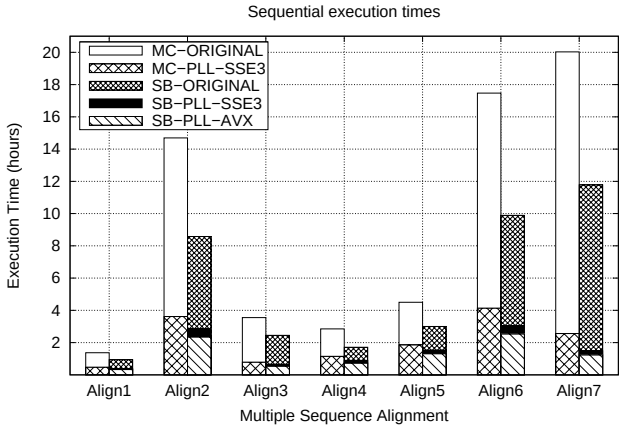

Figure 4. Comparison between the sequential versions of the native DPPDiv (*ORIGINAL*) implementation and the sequential PLL-based DPPDiv implementation using SSE3 (*PLL-SSE3*) and AVX (*PLL-AVX*). The suffix indicates the system: MC for the Magny-Cours node and SB for the Sandy Bridge.

speedups with alignments 6 and 7. The number of unique site patterns is not a determining factor for these tests, despite the fact that, performance also improved slightly with an an increasing number of site patterns. Overall, we achieve a two- to three-fold sequential performance improvement by using the PLL.

B. PThreads Performance

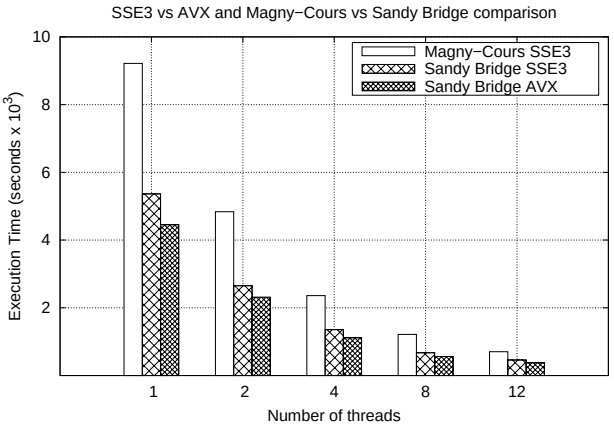

Figure 7. Comparison between the absolute runtimes on Sandy Bridge (with SSE3 and AVX intrinsics) and Magny-Cours nodes (with SSE3 intrinsics) using PThreads on up to 12 cores.

We assessed the performance of the PLL PThreads version using SSE3 intrinsics on one of the Magny-Cours nodes and both SSE3 and AVX intrinsics on one of the Sandy Bridge

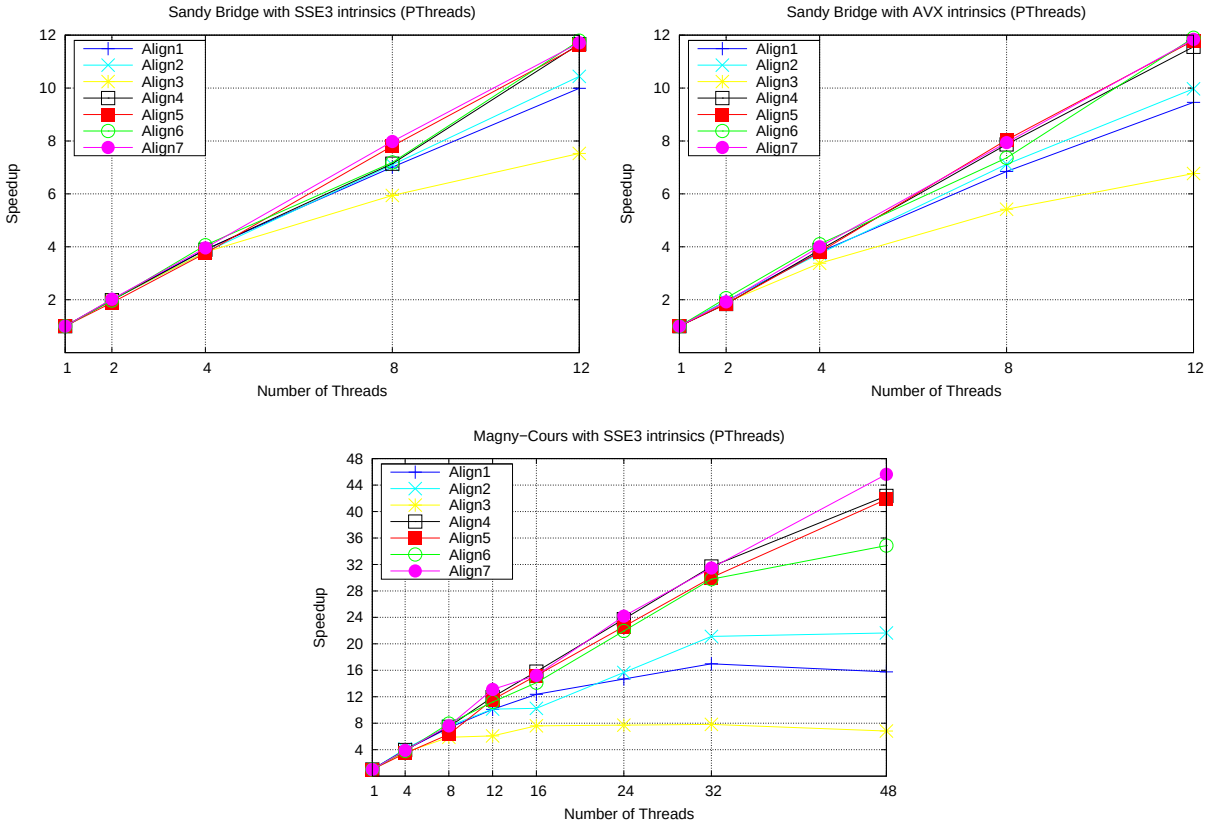

Figure 5. Scalability of DPPDiv using the Phylogenetic Likelihood Library with PThreads (using SSE3 and AVX intrinsics) on the Sandy Bridge and the Magny-Cours nodes.

Table II
COMPARISON OF THE RUNTIMES (IN SECONDS) BETWEEN THE ORIGINAL DPPDIV IMPLEMENTATION AND PLL-BASED IMPLEMENTATION WITH PTHREADS USING SSE3 INTRINSICS ON THE MAGNY-COURS NODE.

	ORIG	**1**	**4**	**8**	**12**	**16**	**24**	**32**	**48**
Align1	4,928	1,715	434	235	170	139	117	101	109
Align2	52,880	13,030	3,164	1,724	1286	1,271	831	617	602
Align3	12,770	2,845	796	484	468	374	369	364	417
Align4	10,243	4,151	1,051	557	347	263	175	131	98
Align5	16,202	6,695	1,949	1,057	579	440	296	223	160
Align6	62,890	14,917	3,921	1,876	1332	1,056	680	501	428
Align7	72,105	9,217	2,361	1,214	703	605	381	293	202

nodes. The alignment site patterns are evenly distributed among the threads by the PLL as described in [46]. The scalability of the PThreads and MPI (see below) versions of the PLL mainly depends on the number of unique site patterns in the alignment (see, e.g., [46]). The speedup plots in Figure 5 and the parallel efficiency plots in Figure 6 illustrate these results. For the sake of completeness we also show the absolute execution times for the original, sequential PLL and PThreads-based PLL versions of DPPDiv in Table II, and a comparison between the two node types in Figure 7. The best overall improvement in terms of time to solution of the PLL-based PThreads implementation over the original DPPDiv implementation was observed for alignment 7. We achieve a 357-fold (72,105 versus 202 seconds) on this dataset. Overall, given that the alignment is long enough (has enough unique site patterns), we observe good parallel scalability for the PLL-based DPPDIV version. For alignments 6 and 7, we observe almost linear speedups even for 48 cores. Hence, using the fast PLL kernel implementation in conjunction with the PThreads version can improve DPPDiv performance by more than a factor of 100 compared to the original, sequential version. More

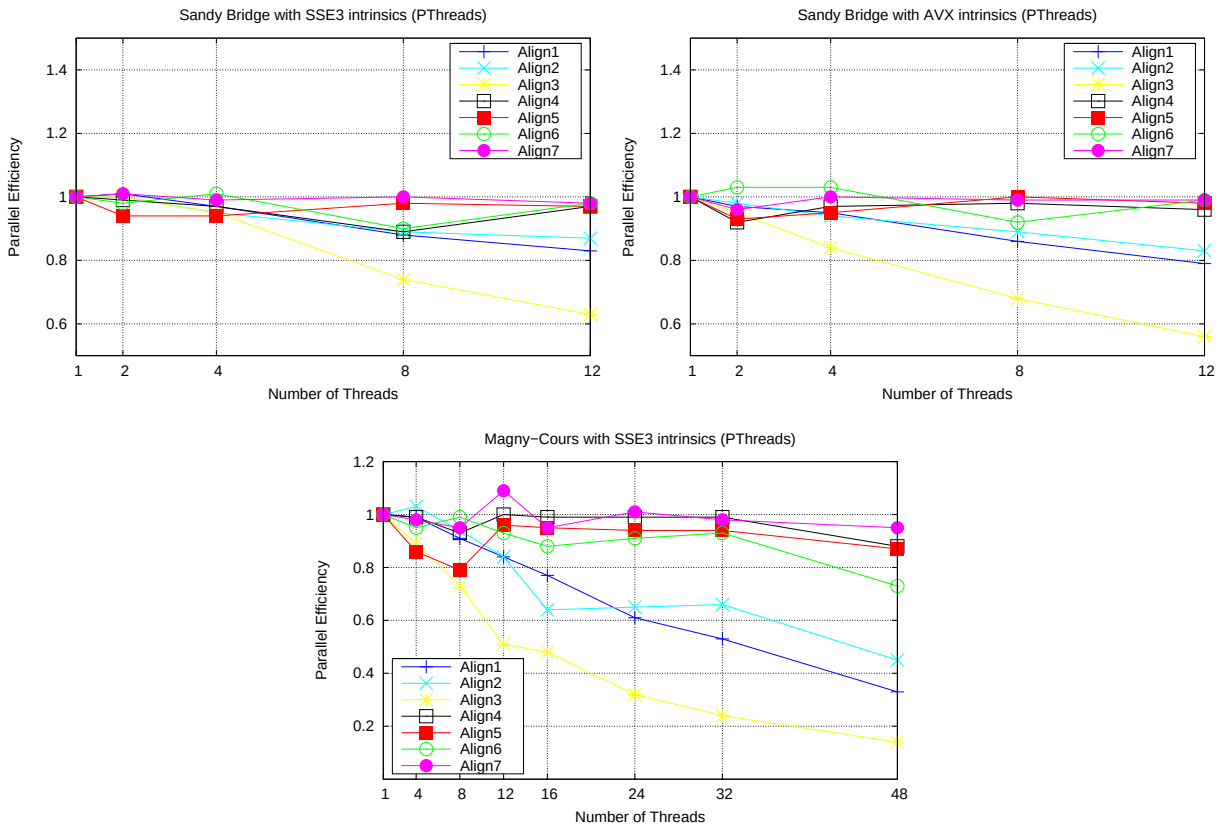

Figure 6. Parallel efficiency of DPPDiv using the Phylogenetic Likelihood Library with PThreads (using SSE3 and AVX intrinsics) on the Sandy Bridge and the Magny-Cours nodes.

importantly, the parallelism essentially comes for free with the PLL.

C. MPI Performance

The MPI performance tests show to which extent the speedups depend on alignment length. Note that, individual Sandy cores are substantially faster than Magny-Cours cores (see Table III). Hence, the performance penalty induced by waiting for communication is larger on the Sandy cores, since less time is required to conduct the per-process likelihood computations. In other words, the communication to computation ratio on these nodes is less favorable than on the Magny-Cours system. In Figure 8 we show the scalability and parallel efficiency results for the PLL-based version of DPPDiv that uses MPI. However, the scalability of the MPI version of the PLL will increase with an increasing number of distinct site patterns. Such large whole-genome scale datasets already exist and are being analyzed (e.g., www.1kite.org). The MPI version in the PLL has been adapted from RAxML-Light [36] and shows good scalability using a large number of processes on whole-genome alignments. The MPI version will also help to accommodate the immense memory requirements of large genomic alignments that can not typically be handled by a single node.

VI. Conclusion and Future Work

We have presented an initial version of our phylogenetic likelihood library and used it to boost performance of DPPDiv, a new program for Bayesian inference of divergence times estimates using a Dirichlet process prior. Source code and datasets are available at https://github.com/ddarriba/pll-dppdiv. Since DPPDiv heavily relies on likelihood computations it represented an ideal candidate to demonstrate the advantages of the PLL. Integrating the PLL into DPPDiv took a visiting PhD student in our lab (DD) only about a month without having prior knowledge of DPPDiv or the PLL. By integrating the PLL, we obtained sequential speedups of 2.4 to 7.8 over the original DPPDiv implementation. By deploying fine-grain loop-level parallelism with PThreads, that comes for free with the PLL and is transparent to the developer of the target application, we obtained near-optimal speedups on sufficiently large input datasets. Using the PThreads version on a 48-core multi-core system in conjunction with the fast sequential

Table III
COMPARISON OF THE RUNTIMES (IN SECONDS) OF THE PLL-BASED IMPLEMENTATION BETWEEN THE MAGNY-COURS (MC) AND THE SANDY BRIDGE (SB) CLUSTERS USING MPI.

	1		4		8		24		48		96	
	MC	**SB**	**MC**	**SB**	**MC**	**SB**	**MC**	**SB**	**MC**	**SB**	**MC**	**SB**
ALIGN1	1,715	1,177	480	345	247	251	133	156	158	192	197	224
ALIGN2	13,030	8,462	3,712	2,370	1,856	1,605	747	839	820	908	940	1,070
ALIGN3	2,845	1,926	912	572	590	634	639	696	704	929	963	1,132
ALIGN4	4,151	2,636	1,101	689	541	418	188	146	111	106	92	90
ALIGN5	6,695	4,726	1,712	1,144	880	685	279	225	169	144	119	111
ALIGN6	14,917	9,139	3,978	2,239	1,984	1,370	664	501	420	365	342	326
ALIGN7	9,217	4,453	2,309	1,107	1,208	557	383	232	203	135	161	98

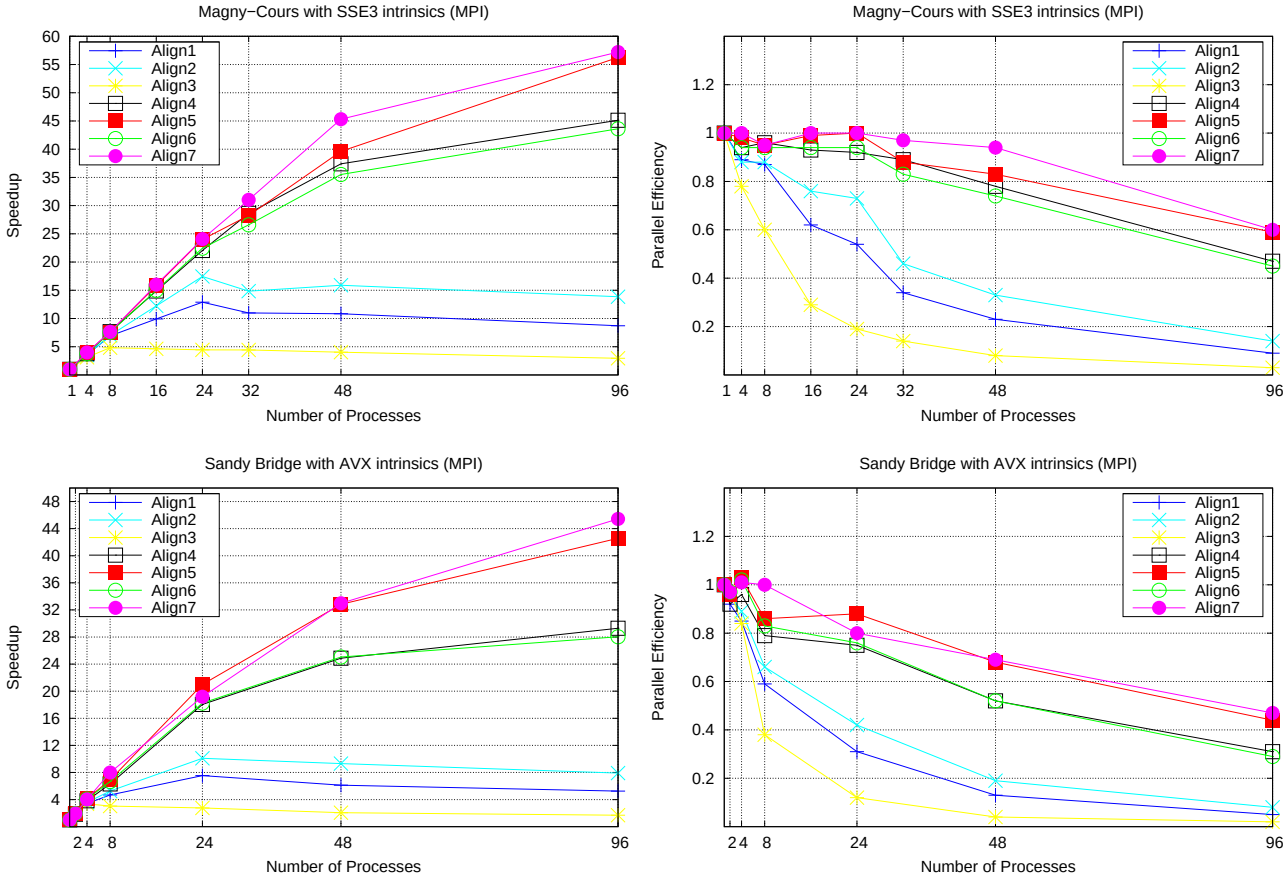

Figure 8. Performance and parallel efficiency of DPPDiv using the Phylogenetic Likelihood Library with MPI and SSE3 and AVX intrinsics.

implementation that relies on vector intrinsics, we were able to reduce the time to solution of DPPDiv by more than a factor of 100 (over factor 350 in the best case), that is, by two orders of magnitude.

Regarding future work, we intend to closely collaborate with the developers of DPPDiv to design a version of the code that can handle partitioned alignments, a feature that is already available in the PLL. Furthermore, we will improve the documentation of the PLL, add additional models and data-types, and design a production-level GPU implementation of our kernels.

REFERENCES

[1] T. A. Heath, M. T. Holder, and J. P. Huelsenbeck, "A dirichlet process prior for estimating lineage-specific substitution rates," *Molecular Biology and Evolution*, vol. 29,

no. 3, pp. 939–955, 2012. [Online]. Available: http://mbe.oxfordjournals.org/content/29/3/939.abstract

[2] T. A. Heath, "A hierarchical Bayesian model for calibrating estimates of species divergence times," *Systematic Biology*, vol. 61, pp. 793–809, 2012.

[3] T. S. Ferguson, "A Bayesian analysis of some nonparametric problems," *Annals of Statistics*, vol. 1, pp. 209–230, 1973.

[4] C. E. Antoniak, "Mixtures of Dirichlet processes with applications to non-parametric problems," *Annals of Statistics*, vol. 2, pp. 1152–1174, 1974.

[5] E. Zuckerkandl and L. Pauling, "Molecular disease, evolution, and genetic heterogeneity," in *Horizons in Biochemistry*, M. Kasha and B. Pullman, Eds. Academic Press, New York, 1962, pp. 189–225.

[6] Z. Yang and A. D. Yoder, "Comparison of likelihood and Bayesian methods for estimating divergence times using multiple gene loci and calibration points, with application to a radiation of cute-looking mouse lemur species," *Systematic Biology*, vol. 52, pp. 705–716, 2003.

[7] A. J. Drummond and M. A. Suchard, "Bayesian random local clocks, or one rate to rule them all," *BMC Biology*, vol. 8, p. 114, 2010.

[8] A. J. Drummond, S. Y. Ho, M. J. Phillips, and A. Rambaut, "Relaxed phylogenetics and dating with confidence," *PLoS Biology*, vol. 4, p. e88, 2006.

[9] T. Lepage, D. Bryant, H. Philippe, and N. Lartillot, "A general comparison of relaxed molecular clock models," *Molecular Biology and Evolution*, vol. 24, pp. 2669–2680, 2007.

[10] R. M. Neal, "Markov chain sampling methods for Dirichlet process mixture models," *Journal of Computational and Graphical Statistics*, vol. 9, pp. 249–265, 2000.

[11] 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.

[12] M. J. Sanderson, "Estimating absolute rates of molecular evolution and divergence times: a penalized likelihood approach," *Molecular Biology and Evolution*, vol. 19, pp. 101–109, 2002.

[13] M. J. Sanderson, "r8s: inferring absolute rates of molecular evolution and divergence times in the absence of a molecular clock," *Bioinformatics*, vol. 19, no. 2, pp. 301–302, 2003.

[14] S. A. Smith and B. C. O'Meara, "treepl: divergence time estimation using penalized likelihood for large phylogenies," *Bioinformatics*, vol. 28, no. 20, pp. 2689–2690, 2012. [Online]. Available: http://bioinformatics.oxfordjournals.org/content/28/20/2689.abstract

[15] X. Xia and Q. Yang, "A distance-based least-square method for dating speciation events," *Molecular Phylogenetics and Evolution*, vol. 59, no. 2, pp. 342 – 353, 2011. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1055790311000443

[16] K. Tamura, F. U. Battistuzzi, P. Billing-Ross, O. Murillo, A. Filipski, and S. Kumar, "Estimating divergence times in large molecular phylogenies," *Proceedings of the National Academy of Sciences*, 2012. [Online]. Available: http://www.pnas.org/content/early/2012/10/31/1213199109.abstract

[17] Z. Yang and B. Rannala, "Bayesian phylogenetic inference using DNA sequences: a Markov chain Monte Carlo method," *Molecular Biology and Evolution*, vol. 14, pp. 717–724, 1997.

[18] J. Thorne, H. Kishino, and I. S. Painter, "Estimating the rate of evolution of the rate of molecular evolution," *Molecular Biology and Evolution*, vol. 15, pp. 1647–1657, 1998.

[19] J. P. Huelsenbeck, B. Larget, and D. L. Swofford, "A compound Poisson process for relaxing the molecular clock," *Genetics*, vol. 154, pp. 1879–1892, 2000.

[20] H. Kishino, J. L. Thorne, and W. Bruno, "Performance of a divergence time estimation method under a probabilistic model of rate evolution," *Molecular Biology and Evolution*, vol. 18, pp. 352–361, 2001.

[21] J. Thorne and H. Kishino, "Divergence time and evolutionary rate estimation with multilocus data," *Systematic Biology*, vol. 51, pp. 689–702, 2002.

[22] Z. Yang and B. Rannala, "Bayesian estimation of species divergence times under a molecular clock using multiple fossil calibrations with soft bounds," *Molecular Biology and Evolution*, vol. 23, pp. 212–226, 2006.

[23] S. Y. W. Ho and M. J. Phillips, "Accounting for calibration uncertainty in phylogenetic estimation of evolutionary divergence times," *Systematic Biology*, vol. 58, pp. 367–380, 2009.

[24] J. Bahl, M. I. Nelson, K. H. Chan, R. Chen, D. Vijaykrishna, R. A. Halpin, T. B. Stockwell, X. Lin, D. E. Wentworth, E. Ghedin, Y. Guan, J. S. Malik Peiris, S. Riley, A. Rambaut, E. C. Holmes, and G. J. D. Smith, "Temporally structured metapopulation dynamics and persistence of influenza a h3n2 virus in humans," *Proceedings of the National Academy of Sciences*, 2011.

[25] S. Shultz, C. Opie, and Q. Atkinson, "Stepwise evolution of stable sociality in primates," *Nature*, vol. 479, no. 7372, pp. 219–222, 2011.

[26] D. Eastwood, D. Floudas, M. Binder, A. Majcherczyk, P. Schneider, A. Aerts, F. Asiegbu, S. Baker, K. Barry, M. Bendiksby *et al.*, "The plant cell wall–decomposing machinery underlies the functional diversity of forest fungi," *Science*, vol. 333, no. 6043, pp. 762–765, 2011.

[27] M. dos Reis, J. Inoue, M. Hasegawa, R. Asher, P. Donoghue, and Z. Yang, "Phylogenomic datasets provide both precision and accuracy in estimating the timescale of placental mammal phylogeny," *Proceedings of the Royal Society B: Biological Sciences*, vol. 279, no. 1742, pp. 3491–3500, 2012.

[28] A. Crottini, O. Madsen, C. Poux, A. Strauß, D. R. Vieites, and M. Vences, "Vertebrate time-tree elucidates the biogeographic pattern of a major biotic change around the K–T boundary in madagascar," *Proceedings of the National Academy of Sciences*, vol. 109, no. 14, pp. 5358–5363, 2012.

[29] Z. Yang, "PAML 4: Phylogenetic Analysis by Maximum Likelihood," *Molecular Biology and Evolution*, vol. 24, no. 8, pp. 1586–1591, 2007. [Online]. Available: http://mbe.oxfordjournals.org/cgi/content/abstract/24/8/1586

[30] N. Lartillot, T. Lepage, and S. Blanquart, "Phylobayes 3: a bayesian software package for phylogenetic reconstruction and molecular dating," *Bioinformatics*, vol. 25, no. 17, p. 2286, 2009.

[31] A. J. Drummond, M. A. Suchard, D. Xie, and A. Rambaut, "Bayesian phylogenetics with beauti and the beast 1.7," *Molecular Biology and Evolution*, vol. 29, no. 8, pp. 1969–1973, 2012.

[32] S. Guindon, "Bayesian estimation of divergence times from large sequence alignments," *Molecular Biology and Evolution*, vol. 27, no. 8, pp. 1768–1781, 2010.

[33] F. Ronquist, M. Teslenko, P. van der Mark, D. L. Ayres, A. Darling, S. Höhna, B. Larget, L. Liu, M. A. Suchard, and J. P. Huelsenbeck, "Mrbayes 3.2: Efficient bayesian phylogenetic inference and model choice across a large model space," *Systematic Biology*, vol. 61, no. 3, pp. 539–542, 2012.

[34] M. dos Reis and Z. Yang, "Approximate likelihood calculation on a phylogeny for bayesian estimation of divergence times," *Molecular Biology and Evolution*, vol. 28, no. 7, pp. 2161–2172, 2011.

[35] D. L. Ayers, A. Darling, D. J. Zwickl, P. Beerli, M. T. Holder, P. O. Lewis, J. P. Huelsenbeck, F. Ronquist, D. L. Swofford, M. P. Cummings, A. Rambaut, and M. A. Suchard, "BEAGLE: An application programming interface and high-performance computing library for statistical phylogenetics," *Systematic Biology*, vol. 61, pp. 170–173, 2012.

[36] A. Stamatakis, A. Aberer, C. Goll, S. Smith, S. Berger, and F. Izquierdo-Carrasco, "Raxml-light: a tool for computing terabyte phylogenies," *Bioinformatics*, vol. 28, no. 15, pp. 2064–2066, 2012.

[37] A. Stamatakis and M. Ott, "Load balance in the phylogenetic likelihood kernel," in *Parallel Processing, 2009. ICPP'09. International Conference on*. IEEE, 2009, pp. 348–355.

[38] J. Zhang and A. Stamatakis, "The multi-processor scheduling problem in phylogenetics," in *Parallel and Distributed Processing Symposium Workshops & PhD Forum (IPDPSW), 2012 IEEE 26th International*. IEEE, 2012, pp. 691–698.

[39] F. Izquierdo-Carrasco, S. Smith, and A. Stamatakis, "Algorithms, data structures, and numerics for likelihood-based phylogenetic inference of huge trees," *BMC bioinformatics*, vol. 12, no. 1, p. 470, 2011.

[40] F. Izquierdo-Carrasco, J. Gagneur, and A. Stamatakis, "Trading memory for running time in phylogenetic likelihood computations," *Heidelberg Institute for Theoretical Studies*, 2011.

[41] J. Felsenstein, "Evolutionary trees from dna sequences: a maximum likelihood approach," *Journal of molecular evolution*, vol. 17, no. 6, pp. 368–376, 1981.

[42] A. Stamatakis, "Orchestrating the phylogenetic likelihood function on emerging parallel architectures," *Bioinformatics–High Performance Parallel Computer Architectures, B. Schmidt, Ed. CRC Press*, pp. 85–115, 2011.

[43] R. W. Meredith, J. E. Janecka, J. Gatesy, O. A. Ryder, C. A. Fisher, E. C. Teeling, A. Goodbla, E. Eizirik, T. L. L. SimÃčo, T. Stadler, D. L. Rabosky, R. L. Honeycutt, J. J. Flynn, C. M. Ingram, C. Steiner, T. L. Williams, T. J. Robinson, A. Burk-Herrick, M. Westerman, N. A. Ayoub, M. S. Springer, and W. J. Murphy, "Impacts of the cretaceous terrestrial revolution and kpg extinction on mammal diversification," *Science*, vol. 334, no. 6055, pp. 521–524, 2011.

[44] E. Paradis, J. Claude, and K. Strimmer, "APE: analyses of phylogenetics and evolution in R language," *Bioinformatics*, vol. 20, pp. 289–290, 2004.

[45] A. Rambaut and N. Grass, "Seq-gen: an application for the monte carlo simulation of dna sequence evolution along phylogenetic trees," *Comput Appl Biosci*, vol. 13, no. 3, pp. 235–238, 1997.

[46] A. Stamatakis and M. Ott, "Efficient computation of the phylogenetic likelihood function on multi-gene alignments and multi-core architectures," *Philosophical Transactions of the Royal Society B: Biological Sciences*, vol. 363, no. 1512, pp. 3977–3984, 2008.