Version dated: October 26, 2014

Supplementary material

Support for amino acid replacement models

For protein data, the library implements 18 common empirical amino acid replacement models: Dayhoff (Dayhoff et al. 1978), JTT (Jones et al. 1992), WAG (Whelan and Goldman 2001), mtREV (Adachi and Hasegawa 1996), MtMam (Yang and Nielsen 1998), VT (Müller and Vingron 2000), CpREV (Adachi et al. 2000), RtREV (Dimmic et al. 2002), MtArt (Abascal et al. 2007), MtZoa (Rota-Stabelli et al. 2009), HIVb/HIVw (Nickle et al. 2007), LG (Le and Gascuel 2008), LG4M (Le et al. 2012), Blosum62 (Henikoff and Henikoff 1992), DcMut/JTT-DcMut (Kosiol and Go 2005), PMB (Veerassamy et al. 2003), and FLU (Dang et al. 2010).

Experiments with the PThreads and MPI versions

To evaluate the performance of the parallel PLL implementations, we used 128 taxon datasets with randomly generated alignments of varying lengths. For each dataset we executed 1000 full tree traversals using the sequential version of the PLL and the corresponding PThreads as well as MPI versions with 12, 24, and 48 cores on a AMD 6174 Opteron shared memory system with four processors (each processor consists of 12 cores). We ran each experiment (processor count and dataset combination) five times and use average values to compute the speedup. In addition, for experiments using less than the total number of cores we tested distinct pinning strategies that may have a substantial effect on performance,

mainly because of cache memory effects (see (Klug et al. 2011)). For instance, the runs with 24 threads and processes were assigned to the physical cores $1, 2, 3, ..., 24$ and using the alternative pinning of $1, 3, 5, ..., 47$ as well. We did not observe any substantial run-time variations though.

The parallel speedup for the PThreads version is depicted in Figure 1 and for the MPI version in Figure 2.

As expected, the speedup generally depends on the alignment length, albeit there is some variation. The longer the alignment the better the speedup becomes for a constant number of threads or processes. In cases where the alignment is long enough, we even observe super-linear speedups because of increased cache efficiency. For the largest test dataset with $1,048,576$ sites we observe a speedup of 14.5 using 12 threads and 12 cores and of 15 using 12 MPI processes on 12 cores. It is not surprising that the MPI version achieves the best overall speedup on 48 cores, since state-of-the-art MPI implementations are highly optimized for NUMA-based (Non Uniform Memory Access) shared memory systems.

Additional MPI and PThreads performance data for using PLL with a real application is available in the paper describing the integration with DPPDiv (Darriba et al. 2013).

BEAGLE versus PLL: Details of comparison

The comparison against BEAGLE is as fair as possible, since it was set up and refined while D. Ayres, the main author of BEAGLE visited our lab in March 2014. In the course of this visit, the scaling procedure in BEAGLE was improved, leading to results that are less favorable for PLL since initial speedups of the PLL over BEAGLE were higher. Nonetheless, we consider that a further improvement of BEAGLE performance is a positive effect of the PLL project and of benefit for the entire community.

We have conducted the experiments by implementing a tool that evaluates the log likelihood score of a fixed tree topology given the alignment, with fixed model parameters. In

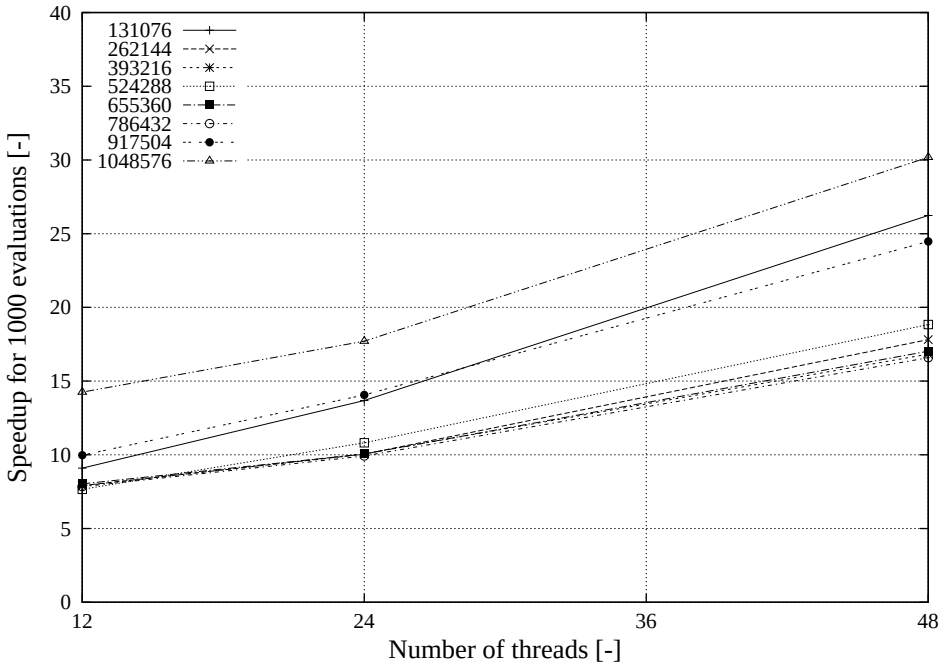

Figure 1: Speedups of the PThreads implementation over the sequential version for $1,000$ likelihood evaluations using full traversals, on datasets with 128 taxa and with varying alignment lengths. The speedup was computed for 12, 24, and 48 threads.

order to properly assess performance and decrease the impact of library initialization times, we compute the likelihood of the full tree 100 times in a loop. The performance difference between the two libraries is mainly due to the numerical scaling implementations that prevent numerical underflow. The PLL incorporates numerical scaling by default (scaling can not be disabled), that is, it always checks whether rescaling is necessary. BEAGLE provides a more fine-grained control of numerical scaling. It allows the user to disable the checks completely for obtaining improved performance at the risk of experiencing underflows. To guarantee that the evaluation of the likelihood is numerically stable for an arbitrary number of taxa, an arbitrary tree shape, and an arbitrary alignment size in our experiments, we have compared PLL with the standard scaling option in BEAGLE enabled. Furthermore, we have also computed the number of conditional likelihood vectors (CLVs) entry updates per second.

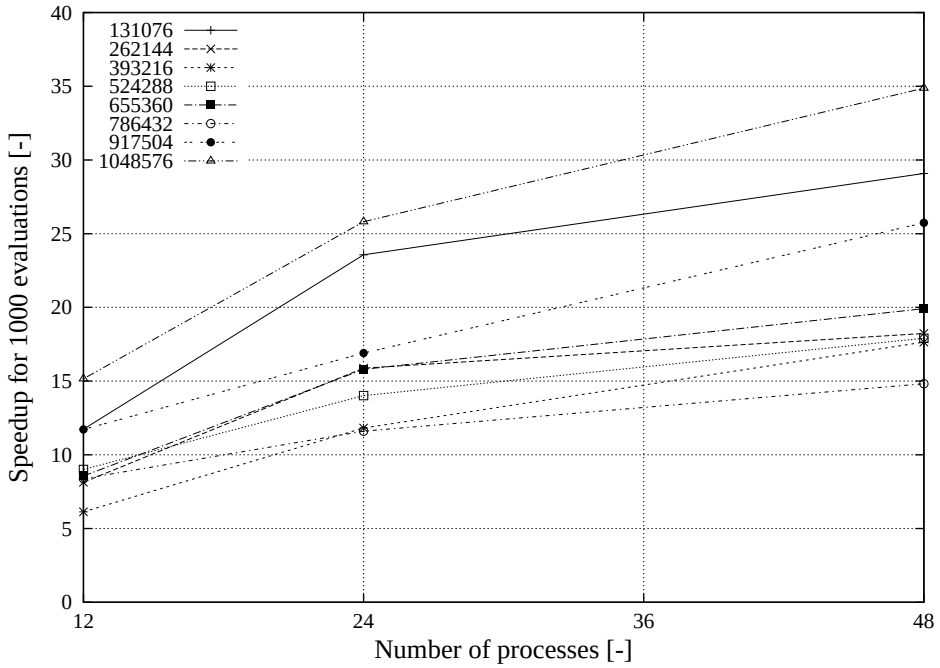

Figure 2: Speedups of the MPI implementation over the sequential version for 1,000 likelihood evaluations using full traversals, on datasets with 128 taxa and with varying alignment lengths. The speedup was computed for 12, 24, and 48 MPI processes.

To obtain an as a fair as possible comparison between the two libraries, we evaluated the likelihood of the tree 100 times in a loop such that each iteration is independent of the rest, that is, an iteration is not allowed to reuse intermediate results (such as scaling factors) from the preceding iterations. To calculate CLV updates per second we used the computation time of the fastest iteration. The number of CLV entry updates for a full tree traversal (including the computation of the overall log likelihood at the virtual root)

$$(n-1) \cdot (s \cdot m \cdot r)$$

is then divided by the run time of the fastest iteration to obtain the number of CLV entry updates per second. The constant represents the CLV size, times the number of the updates, i.e. n is the number of taxa, s the number of states (4 for DNA), m the sequence size and r the number of rate categories (4 for Γ). We also calculated the ratio for the overall execution

times (all 100 iterations) between BEAGLE and PLL to verify the above results.

The experiments were executed on a 4-core Intel i7-2600 system with 16GB RAM and swapping disabled, and on an AMD 6174 Opteron. We used randomly generated topologies of 128 and 256 taxa, and randomly generated alignments with lengths of $131,072$, $262,144$, $393,216$, $524,288$, $656,360$, $786,432$, $917,504$ and $1,048,576$ bp. The results for the 128 taxon datasets on the Intel system are presented in Table 1 and the corresponding performance plot is shown in the main paper. The entries for $917,504$ in the BEAGLE part are missing because the process was killed due to lack of memory. PLL values are available because PLL uses approximately 12% less memory than BEAGLE.

For the sake of completeness we also provide plots of the runs on the AMD system, which has sufficient memory to evaluate all 128 and 256 taxon datasets with both libraries, and therefore assess performance with datasets that require more than 16GB of memory. We compared the SSE versions of the Phylogenetic Likelihood Function (PLF) of both libraries, and the performance comparison results (in terms of CLV updates per second) are shown in Figure 3 for the 128 taxon run and in Figure 4 for the 256 taxon run. On the AMD system, PLL was upto 3.5 times faster than BEAGLE.

Comparing the parallel performance of the two libraries is not straight-forward because multi-core support in BEAGLE is implemented via OpenCL and automatically uses all available cores on a system, that is, it is not possible to perform experiments with different core counts. Moreover, it works only on Intel processors. Furthermore, the automatic vectorization that OpenCL provides, selects the most advanced vector instruction set available on the host machine and hence, if AVX is present, it is not possible to test SSE versions. Therefore, we only provide a parallel efficiency comparison of the AVX [1] versions of the two libraries for all cores on the test system. Figure 5 illustrates the performance of the

[1]Note that, BEAGLE does not incorporate a native AVX implementation in its single-thread version, and the multi-core AVX version is automatically vectorized by OpenCL

Alignment	BEAGLE SSE				PLL						
					SSE			AVX			
	MPAR	GFLOP	time	log-L	MPAR	GFLOP	time	MPAR	GFLOP	time	log-L
131,072	171.369682	3.913329	157.300s	-70472791.622764	301.010840	4.552506	92.680s	427.312464	6.484879	63.337s	-70472791.622758
262,144	170.891152	3.891404	313.116s	-140943953.194489	323.681962	4.898266	166.472s	478.952075	7.278620	113.223s	-140943953.194475
393,216	171.835598	3.924084	467.673s	-211440247.646654	323.755212	4.899271	249.940s	478.035837	7.264815	167.566s	-211440247.646636
524,288	169.806928	3.858923	631.745s	-281866155.938091	324.102334	4.904544	333.905s	477.078372	7.263155	224.247s	-281866155.938067
655,360	171.825510	3.926355	779.396s	-352366192.243255	321.507359	4.864999	418.631s	463.254091	7.037001	280.444s	-352366192.243210
786,432	171.365657	3.912479	937.291s	-422829193.638436	323.775481	4.899593	502.909s	476.265642	7.237517	341.037s	-422829193.638400
917,504					324.084967	4.904243	584.862s	477.027164	7.248935	403.230s	-493322676.724308

Table 1: Detailed listing of the performance figures for the single-thread versions of the BEAGLE and PLL phylogenetic likelihood functions. Column MPAR presents the number of CLV updates per second (in millions) for the fastest likelihood evaluation out of 100, GFLOP lists the number of billions of floating operations per second for the fastest likelihood evaluation, and time is the total run-time for the 100 independent likelihood evaluations. We also provide the resulting log-likelihood values to illustrate the equivalence between the likelihood scores of the two libraries.

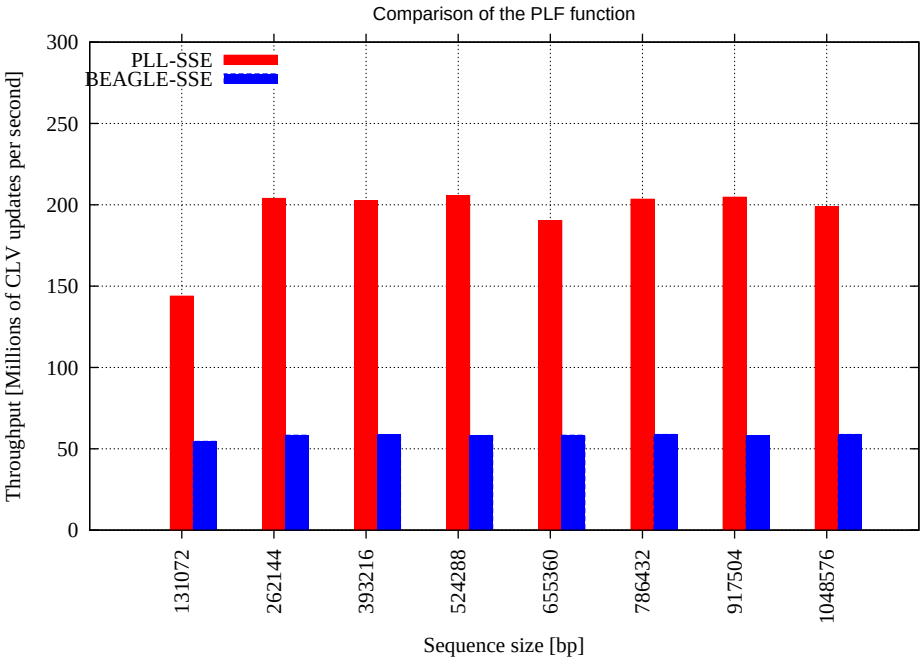

Figure 3: Comparison of the PLF performance using 128 taxa (AMD machine).

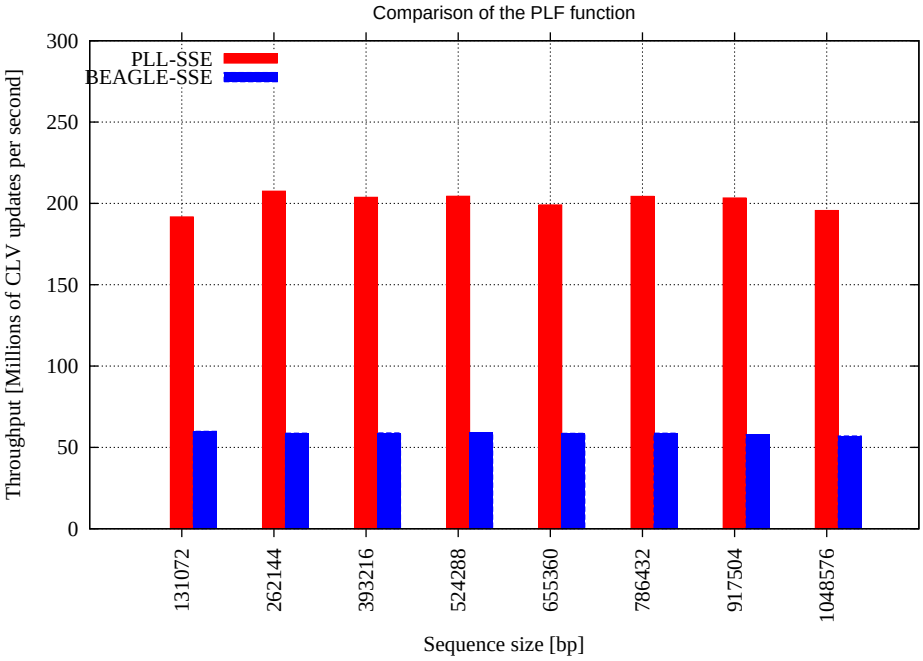

Figure 4: Comparison of the PLF performance using 256 taxa (AMD machine).

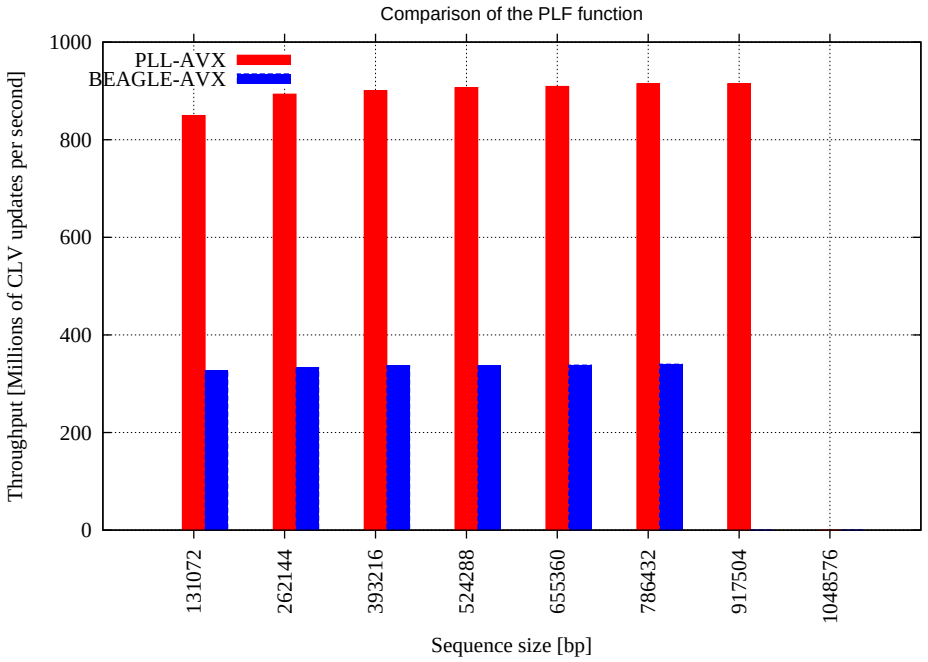

Figure 5: Comparison of the PLF performance on a multi-core system using 128 taxa and varying alignment lengths.

two libraries on randomly-generated 128 taxon DNA datasets. The AVX POSIX Threads implementation of the PLL is 3 times faster than the BEAGLE OpenCL AVX version. The experiments were conducted on a quad-core Intel i7-2600 with hyperthreading enabled, that is, using 8 SIMT cores, and for BEAGLE OpenCL 1.2 (Build 82248) was used.

TreeBASE Matrix ID

DNA

M10243	M10245	M10434	M10467	M10933	M1110	M11113	M11745	M11762
M12051	M12098	M1224	M12388	M13718	M14164	M14165	M14678	M1838
M214	M2307	M2534	M2902	M2931	M3031	M3198	M3514	M3605
M3777	M4170	M4324	M4326	M4399	M4720	M4727	M4794	M4806
M4850	M4900	M4927	M4938	M5078	M5235	M5381	M5731	M5931
M6134	M6414	M6625	M667	**M7024**	M7054	M7165	M7210	M7211
M7292	M7929	M8012	M8385	M8619	M8692	M8703	M8975	M8982
M8984	M9033	M9142	M9143	M980	M9915			

Protein

M10236	M10866	M11012	M11013	M1118	M11335	M11336	M11338	M11341
M11342	M11344	M11595	M11596	M11740	M12103	M12104	M12376	M13804
M1726	M2358	M2593	M2926	M3113	M3807	M3810	M4249	**M4292**
M4318	M4325	M4539	M4780	M4860	M4884	M510	M5379	M6416
M7729	M8175	M8461	M8630	M9973				

IDs shown in bold font denote the datasets used for the evaluation of PThreads performance.

*

References

Abascal, F., D. Posada, and R. Zardoya. 2007. MtArt: a new model of amino acid replacement for Arthropoda. Mol. Biol. Evol. 24:1–5.

Adachi, J. and M. Hasegawa. 1996. Model of amino acid substitution in proteins encoded by mitochondrial DNA. J. Mol. Evol. 42:459–468.

Adachi, J., P. J. Waddell, W. Martin, and M. Hasegawa. 2000. Plastid genome phylogeny and a model of amino acid substitution for proteins encoded by chloroplast DNA. J. Mol. Evol. 50:348–358.

Dang, C., Q. Le, O. Gascuel, and V. Le. 2010. FLU, an amino acid substitution model for influenza proteins. BMC Evol. Biol. 10:99.

Darriba, D., A. J. Aberer, T. Flouri, T. A. Heath, F. Izquierdo-Carrasco, and A. Stamatakis. 2013. Boosting the performance of bayesian divergence time estimation with the phylogenetic likelihood library. *in* IPDPS Workshops IEEE Computer Society.

Dayhoff, M., R. Schwartz, and B. Orcutt. 1978. A model for evolutionary change in proteins. Pages 345–352 *in* Atlas of Protein Sequence and Structure (Dayhoff, ed.). Nat'l Biomedical Research Foundation.

Dimmic, M. W., J. S. Rest, D. P. Mindell, and R. A. Goldstein. 2002. rtREV: An amino acid substitution matrix for inference of retrovirus and reverse transcriptase phylogeny. J. Mol. Evol. 55:65–73.

Henikoff, S. and J. G. Henikoff. 1992. Amino acid substitution matrices from protein blocks. Proc. Natl. Acad. Sci. 89:10915–10919.

Jones, D. T., W. R. Taylor, and J. M. Thornton. 1992. The rapid generation of mutation data matrices from protein sequences. Comp. Appl. Biosci. 8:275–282.

Klug, T., M. Ott, J. Weidendorfer, and C. Trinitis. 2011. Autopin–automated optimization of thread-to-core pinning on multicore systems. Pages 219–235 *in* Transactions on high-performance embedded architectures and compilers III. Springer.

Kosiol, C. and N. Goldman. 2005. Different versions of the dayhoff rate matrix. Mol. Biol. Evol. 22:193–199.

Le, S. Q., C. C. Dang, and O. Gascuel. 2012. Modeling Protein Evolution with Several Amino Acid Replacement Matrices Depending on Site Rates. Mol. Biol. Evol. 29:2921–2936.

Le, S. Q. and O. Gascuel. 2008. An improved general amino acid replacement matrix. Mol. Biol. Evol. 25:1307–1320.

Müller, T. and M. Vingron. 2000. Modeling Amino Acid Replacement. J. Comput. Biol. 7:761–776.

Nickle, D. C., L. Heath, M. A. Jensen, P. B. Gilbert, J. I. Mullins, and S. L. Kosakovsky Pond. 2007. HIV-Specific Probabilistic Models of Protein Evolution. PLoS ONE 2:e503.

Rota-Stabelli, O., Z. Yang, and M. J. Telford. 2009. MtZoa: A general mitochondrial amino acid substitutions model for animal evolutionary studies. Mol. Phylogenet. Evol 52:268 – 272.

Veerassamy, S., A. Smith, and E. R. M. Tillier. 2003. A transition probability model for amino acid substitutions from blocks. J. Comput. Biol. Pages 10–997.

Whelan, S. and N. Goldman. 2001. A general empirical model of protein evolution derived from multiple protein families using a maximum-likelihood approach. Mol. Biol. Evol. 18:691–699.

Yang, Z. and R. Nielsen. 1998. Synonymous and nonsynonymous rate variation in nuclear genes of mammals. J. Mol. Evol. 46:409–418.