Scientific Programming 16 (2008) 255–270
DOI 10.3233/SPR-2008-0247
IOS Press

Best Paper Award at SuperComputing 2007

Large-scale phylogenetic analysis on current HPC architectures

Michael Ott [a,*], Jaroslaw Zola [b], Srinivas Aluru [b], Andrew D. Johnson [c,d], Daniel Janies [c] and Alexandros Stamatakis [e]

[a] *Department of Computer Science, Technical University of Munich, Munich, Germany*
[b] *Department of Electrical and Computer Engineering, Iowa State University, IA, USA*
[c] *Department of Biomedical Informatics, The Ohio State University Medical Center, OH, USA*
[d] *Framingham Heart Study, National Heart Lung and Blood Institute, MD, USA*
[e] *The Exelixis Lab, Department of Computer Science, Ludwig-Maximilians-University Munich, Munich, Germany*

Abstract. Phylogenetic inference is considered a *grand challenge* in Bioinformatics due to its immense computational requirements. The increasing popularity and availability of large multi-gene alignments as well as comprehensive datasets of single nucleotide polymorphisms (SNPs) in current biological studies, coupled with rapid accumulation of sequence data in general, pose new challenges for high performance computing. By example of RAxML, which is currently among the fastest and most accurate programs for phylogenetic inference under the Maximum Likelihood (ML) criterion, we demonstrate how the phylogenetic ML function can be efficiently scaled to current supercomputer architectures like the IBM BlueGene/L (BG/L) and SGI Altix. This is achieved by simultaneous exploitation of coarse- and fine-grained parallelism which is inherent to every ML-based biological analysis. Performance is assessed using datasets consisting of 270 sequences and 566,470 base pairs (haplotype map dataset), and 2,182 sequences and 51,089 base pairs, respectively. To the best of our knowledge, these are the largest datasets analyzed under ML to date. Experimental results indicate that the fine-grained parallelization scales well up to 1,024 processors. Moreover, a larger number of processors can be efficiently exploited by a combination of coarse- and fine-grained parallelism. We also demonstrate that our parallelization scales equally well on an AMD Opteron cluster with a less favorable network latency to processor speed ratio. Finally, we underline the practical relevance of our approach by including a biological discussion of the results from the haplotype map dataset analysis, which revealed novel biological insights via phylogenetic inference.

Keywords: Phylogenetic inference, maximum likelihood, RAxML, IBM BlueGene/L

1. Introduction

Phylogenetic trees are used to represent the evolutionary history of a set of n organisms. A multiple alignment of DNA and/or protein sequences that represent these n organisms can be used as input for phylogenetic inference. In a phylogeny the organisms of the input dataset are located at the tips (leaves) of a binary tree and the inner nodes represent extinct common ancestors. Evolutionary events such as mutations of molecular sequences are modeled to occur along the branches of the tree, between ancestor and descendant. Phylogenetic trees have many important applications in medical and biological research (see [1] for a summary) ranging from mapping of the emergence of infectious diseases [17] to the tests of whether Caribbean frogs have a common origin or represent multiple independent invasions of the islands [16].

Due to the continuously accelerating accumulation of sequence data, caused, e.g., by the recent introduction of new massively parallel sequencing technologies such as found in the 454 pyrosequencing machine (http://www.454.com/), there is an increasing demand to compute large trees which often comprise thousands of organisms, each represented by data from several

*Corresponding author. Tel.: +49 89 28918478; Fax: +49 89 28917662; E-mail: ottmi@in.tum.de.

genes, thousands of SNPs, or whole genomes. Since alignment matrices continuously grow in both dimensions (number of organisms *and* alignment length), efficient parallel phylogeny programs are required to handle memory demands (which are primarily a function of alignment length) and growing inference times (which are primarily a function of the number of organisms).

The main algorithmic problem computational phylogeneticists face consists in the huge number of potential alternative tree topologies for a given set of n sequences. In fact, it has recently been shown that the Maximum Likelihood (ML) phylogeny problem is NP-hard [6]. The algorithmic complexity of this problem is due to the vast number of alternative tree topologies which grows exponentially with the number of organisms n, e.g. for $n = 50$ there already exist 2.84×10^{76} alternative trees. In the worst case, in order to find *the* Maximum Likelihood tree, all potential alternative trees would need to be constructed and evaluated under ML. Thus, for trees containing more than 20–25 sequences, efficient heuristic tree search algorithms are required to reduce the size of the search space, since the problem cannot be solved any more.

Significant progress in the field of heuristic ML search algorithms has been achieved over the last 5 years with the release of algorithms implemented in programs such as IQPNNI [20], PHYML [14], GARLI [32] and RAxML [25,27], to name but a few. Note that, none of these heuristic search algorithms are guaranteed to find *the* ML tree, and will hence only yield a *best-known* ML tree. In order to explore the search space more thoroughly, some algorithms such as RAxML offer the possibility to initiate multiple tree searches from distinct starting points (starting trees), i.e. to conduct multiple ML tree searches.

In addition to the algorithmic difficulty, ML-based inference of phylogenies is memory- and floating point-intensive. In fact, both memory consumption as well as inference times grow linearly with the number of alignment columns (see Section 3.1). Due to the continuous accumulation of sequence data, the application of high performance computing techniques is becoming a crucial factor for the reconstruction of ever larger phylogenies and the success of phyloinformatics.

RAxML-7.0.0 [25] (Randomized Axelerated Maximum Likelihood version 7) is an open-source code for large-scale ML-based [11] inference of evolutionary trees using multiple alignments of DNA and/or AA (Amino Acid/Protein) sequences. It is becoming increasingly popular in real-world biological studies and forms an integral component of the CIPRES (CyberInfrastructure for Phylogenetic RESearch, www.phylo.org) project and the greengenes workbench [9] (greengenes.lbl.gov).

Moreover, some of the largest published ML-based phylogenetic analyses to date have been conducted with RAxML [12,19,23]. A recent performance study [25] on real world datasets with more than 1,000 sequences reveals that RAxML is able to find better trees in less time *and* with lower memory consumption than other current ML programs (IQPNNI, PHYML, GARLI, MrBayes).

By example of RAxML we present a generally applicable parallelization strategy for ML-based phylogeny programs to current supercomputer architectures like the IBM BLueGene/L or the SGI Altix as well as common cluster architectures. We also devise an efficient mechanism to distribute data structures across nodes which removes memory requirements as a limiting factor of large-scale phylogenetic analyses. The performance of our parallelization is assessed under the GTR $+\,\Gamma$ model of evolution [31] using the two largest datasets that have been analyzed under ML to date, in terms of input matrix dimensions and memory footprint:

- A multi-gene alignment of 2,182 mammalian sequences with 51,089 nucleotide positions.
- An alignment of non-redundant SNPs (Single Nucleotide Polymorphisms) on the human chromosome 1 that consist of 270 sequences and 566,470 base pairs.

A full analysis on these datasets is currently not feasible on conventional machines.

The remainder of this paper is organized as follows: First, we review related work on parallelization of ML programs (Section 2). In Section 3 we describe a new parallelization strategy for such programs that exploits fine-grained (Section 3.1) and coarse-grained (Section 3.2) parallelism. Our experimental setup and the performance analysis of the proposed parallelization strategy are provided in Section 4. The biological results and insights we obtained from our large-scale phylogenetic analyses on HPC systems are presented in Section 5. We conclude the paper with Section 6.

2. Related work and previous parallelizations of RAxML

RAxML exploits two distinct levels of parallelism: fine-grained loop-level parallelism and coarse-grained

embarrassing parallelism. The program has been previously parallelized with OpenMP to exploit loop-level parallelism.[1] Like every ML-based program, RAxML exhibits a source of loop-level parallelism in the likelihood functions which typically consume over 95% of the overall computation time. The OpenMP implementation scales particularly well on large multi-gene alignments due to increased cache efficiency [28]. Note that loop-level parallelism can further be exploited at two levels of granularity: at a comparatively coarse-grained OpenMP level and at a fine-grained CPU level via SIMD instructions. These two layers of loop-level parallelism have been exploited in a recent RAxML porting to the IBM CELL processor [4,5]. However, the main focus of porting RAxML on Cell was on exploring programming and scheduling techniques for this architecture, using a complex bioinformatics application. In contrast to the current paper, the work on Cell represents a proof-of-concept implementation, rather than a parallelization for large-scale production runs.

The MPI version of RAxML exploits the embarrassing parallelism that is inherent to every real-world phylogenetic analysis. In order to conduct such an analysis (see [12] for an example), about 20–200 distinct tree searches (multiple inferences) to find a best-scoring ML tree on the original alignment as well as a large number of (100–1,000) bootstrap (BS) analyses have to be conducted. *Bootstrap Analyses* are required to assign confidence values ranging between 0.0 and 1.0 to the inner nodes of the best-known/best-found ML tree. This allows one to determine how well-supported certain parts of the tree (the hypothesis) are and is important for the respective biological conclusions. Bootstrapping is analogous to multiple inferences. The only difference being, that inferences are conducted on a randomly re-sampled alignment or BS replicate, i.e., a certain number of alignment columns are re-weighted for every bootstrap run. Phylogenetic bootstrapping can be regarded as assessment of the topological stability of the tree under slight alterations of the input data.

All those individual tree searches, be it bootstrap or multiple inferences, are completely independent from each other and can thus be exploited by a simple master–worker scheme. If there is a sufficient amount of memory per CPU and every CPU can be assigned an inference job, this represents the most efficient approach to exploit HPC platforms for production runs.

[1] The latest release of RAxML uses Pthreads instead of OpenMP.

Most other parallel implementations of ML programs [10,20,26,29,32] have mainly focused on the intermediate level of parallelism (inference/search algorithm parallelism) which is situated between the loop-level parallelism and coarse-grained parallelism currently exploited in RAxML. The work on the exploitation of inference parallelism mainly deals with highly algorithm-specific and mostly MPI-based parallelization of various hill-climbing, genetic, as well as divide-and-conquer search algorithms. Typically, such parallelizations yield a lower parallel efficiency compared to the embarrassing and loop-level types due to hard-to-resolve dependencies in the respective search algorithms. Moreover, these parallelizations are much more program-specific and thus not generally applicable. Minh et al. [21] recently implemented a hybrid OpenMP/MPI version of IQPNNI which exploits loop-level and inference parallelism.

3. A parallelization scheme for RAxML

Though there exist SMP-based supercomputers like the SGI Altix where communication between several threads or processes can be achieved via shared memory, we use MPI as programming paradigm for both levels of parallelism (coarse- and fine-grained) for the following reasons:

- Most parallel architectures (for example, Linux clusters or the IBM BlueGene/L) are non-SMP systems, a generally applicable approach cannot rely on the availability of a pure SMP system.
- Using message-passing on SMP systems is technically feasible and exhibits good performance since messages between processes are exchanged via shared memory.
- Parallelizations for shared memory systems based on OpenMP frequently exhibit suboptimal scalability because the required data locality – which is particularly important on NUMA systems – is hard to achieve and control [22]. Given the fact that MPI-processes do not share common memory regions, data locality for each process is induced by the programming paradigm.

Thus, a hybrid MPI/MPI-based coarse-/fine-grained parallelization provides a sufficient degree of flexibility to either simultaneously compute many jobs on a relatively short alignment or to orchestrate a large number of processors for the joint computation of the likelihood function on very long and memory-intensive alignments on a broad variety of HPC architectures.

3.1. Fine-grained parallelism

As already mentioned the computation of the likelihood function consumes over 90–95% of total execution time in all current ML and Bayesian implementations. Due to their intrinsic fine-grained parallelism coupled with a low number of dependencies, the ML functions thus represent ideal candidates for parallelization at a low level of granularity.

To compute the likelihood of a fixed *unrooted* tree topology with given branch lengths one needs to compute the entries for all likelihood vectors, which are located at the inner nodes of the tree, bottom-up towards a virtual root that can be located at any branch of the tree. For DNA data each of the m entries, where m is the alignment length, of an inner likelihood vector consists of 4 double values (20 double values for amino acids). These double values contain the probabilities of observing an `A, C, G` or `T`, the 4 DNA-bases adenine, guanine, cytosine and thymine, at this specific inner node. If in addition, as in the present case, the discrete Γ model of rate heterogeneity [31] with 4 discrete rates is used, each entry of the likelihood vector consists of $4 \cdot 4 = 16$ ($4 \cdot 20 = 80$ for AA) double values, 4 (20) for each discrete rate. The sequences of the alignment are located at the tips of the tree topology and are represented by tip vectors which consist of simple `char*` arrays of length m to which the AA or DNA alphabet is mapped (see [2] for more implementation details).

The data-structures required to store these n sequences at the tips and the $n - 2$ likelihood vectors at the inner nodes account for more than 90% of the total memory footprint of RAxML. In fact, the memory consumption of all ML and Bayesian implementations is largely dominated by these data structures.

Once the likelihood vectors have been computed, the log likelihood value of the tree topology can be evaluated by summing up over the likelihood vector values to the left and right of the virtual root. Moreover, in order to just obtain *the* Maximum Likelihood value for a single *fixed* tree topology, all individual branch lengths must be optimized with respect to the overall likelihood score. For a more detailed description please refer to [11] and [24]. Note that, most current search algorithms such as GARLI, RAxML, or PHYML, do not re-optimize *all* branch lengths after a change in the tree topology but carry out local optimizations in that neighborhood of the tree which is most affected by the change. The main bulk of these computations consists of `for`-loops over the length m of the alignment. The individual iterations of the `for`-loops over tip and likelihood vectors are independent from each other. This property is due to one of the fundamental assumptions of the ML model which states that individual columns evolve independently from each other [11].

We now summarize the three basic operations at an abstract level and provide their approximate contributions to overall run-time. All operations essentially consist in combining the values of two or three likelihood and/or tip vectors via a relatively large number of floating point operations:

1. *Computation of partial likelihood vectors (approximately 55–60% of run-time)*: This operation computes the entries of a likelihood vector located at an inner node p by combining the values of the likelihood or tip vectors and branch lengths of its two descendants. Thus, this function operates on 3 likelihood/tip vectors but does not require any reduction operations.
2. *Log likelihood value computation (approximately 5% of run-time)*: This function just combines the values of two likelihood/tip vectors at the nodes located at either end of the branch where the virtual root has been placed and computes the log likelihood score. It requires a global reduction operation.
3. *Branch length optimization (approximately 30–35% of run-time)*: This operation optimizes a specific branch between two nodes of the tree (two likelihood/tip vectors) via a Newton–Raphson procedure. In order to perform this operation, synchronization between each individual iteration of the Newton–Raphson method is required as well as respective reduction operations to compute the derivatives of the likelihood function.

In the following we describe how the fine-grained parallelism which is typically exploited with OpenMP (with Pthreads in the most recent RAxML release) on SMP systems [21,28] can be mapped to appropriate MPI collective communication operations. We have implemented a master–worker approach where the master process maintains the only copy of the tree topology and steers the actual tree search as outlined in [27] by issuing the three distinct types of likelihood vector combination instructions to the worker processes.

At initialization each of the p worker processes allocates a fraction m/p space for the n tip and $n - 2$ inner likelihood vectors, i.e. he memory space for tip/likelihood vectors is equally distributed among the processes. These vectors are consistently enumerated

in all workers and the master, despite the fact that no memory is actually allocated at the master. The workers are light-weight processes because they only implement the actual mathematical operations on the tip and likelihood vectors. Thus, the master process simply has to broadcast commands such as optimize the branch length between vectors number x and y given the current branch length z. Global reduction operations, which in both cases (log likelihood computation and branch length optimization) are simply an addition over m double values, are performed via the respective MPI collective reduction operation. In contrast to the aforementioned operations (branch length optimization and likelihood computation), the computation of inner likelihood vectors frequently consists of a series of recursive calls, depending on how many vectors must be updated due to changes in the tree topology or model parameters. In order to reduce the communication frequency such series of recursive calls are transformed into an iterative sequence of operations by the master. The master then sends the whole iterative sequence of inner likelihood vector updates to each worker via one single broadcast.

In Fig. 1 we provide a simplified view of the parallel implementation for an alignment with 4 sequences and 100 distinct alignment columns ($m = 100$). The two inner likelihood ($V1$, $V2$, large rectangles) and four tip vectors ($S1$–$S4$, thick black lines) are split equally among both worker processes. The master only maintains the tree data structure and executes the RAxML search algorithm, i.e., steers the tree search and coordinates the likelihood computations. In this example the master broadcasts a request for branch length optimization of branch $z5$ which is performed by executing computations on the likelihood vectors $V1$ and $V2$ in the workers. Note that, the master only needs to send the vector reference numbers Ref($V1$), Ref($V2$) to the workers.

In order to improve the efficiency in cases where the master only needs to orchestrate a small number of workers, e.g., due to more performant individual CPUs or a comparatively short dataset, one can conduct an appropriate fraction of the likelihood computations at the master (see Section 4.2 for respective results of this modification).

3.2. Coarse-grained parallelism

As outlined in Section 2, RAxML also exploits the embarrassing parallelism inherent to every ML-based production run on real biological data via a simple master–worker scheme. A centralized master distributes tree inference jobs on distinct starting trees or distinct bootstrap replicates to the worker processes.

We modified the above scheme to exploit hybrid parallelism with RAxML using MPI for both layers:

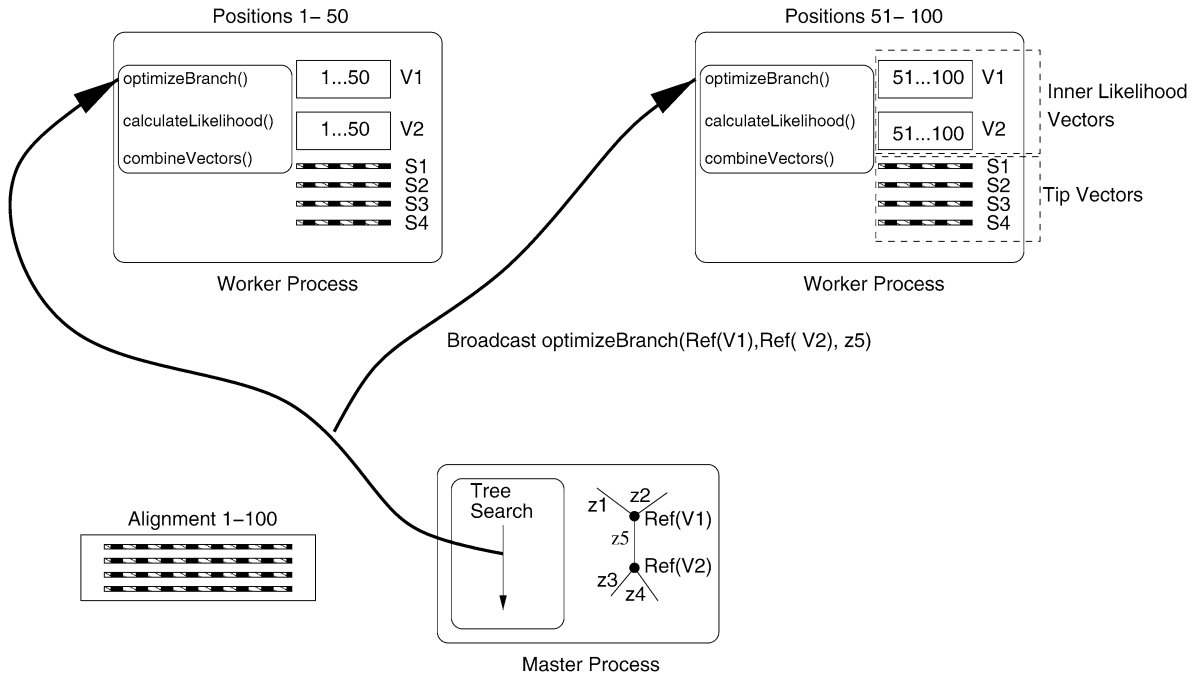

Fig. 1. Simplified representation of the fine-grained parallelization strategy.

coarse-grained work distribution and fine-grained parallelism as described in the preceding section. In a small example scenario a set of 4 individual master processes might be working on fine-grained individual ML searches on 20 distinct starting trees or bootstrap replicates. Those 4 masters can, e.g., coordinate the computations on 3, 7 or 15 individual worker processes each – depending on the dataset size – to carry out likelihood computations. In order to distribute coarse-grained work at the tree search level one of those 4 masters has to assume the role of a super-master. Apart from scheduling fine-grained work to its respective private set of workers the super-master also distributes coarse-grained work to the remaining regular masters. For this purpose we appropriately modified the straight-forward master–worker scheme of the standard RAxML distribution which uses a work queue. These modifications are required to avoid frequent perturbations of fine-grained work scheduling at the super-master by coarse-grained work distribution to other master processes. Note that, the execution time for a tree or bootstrap search typically takes at least several minutes – if not hours – i.e. master/super-master communication is relatively scarce.

Initially, we divide the `MPI_COMM_WORLD` communicator into the respective subgroups (4 subgroups in our example) by using the `MPI_Comm_split` command. On BG/L we apply the following scheme: Each resulting sub-communicator is built as a 3D mesh with dimensions x, y, z, such that $x \approx y \approx z$. The master node has coordinates $(0, 0, 0)$. This mechanism to create partitions and place the master node is well adapted to the collective operations in the current IBM MPI implementation. When `MPI_COMM_WORLD` is used `MPI_Bcast` and `MPI_Reduce` utilize the specialized low-latency network for collective communication. However, when custom communicators are used collective communication is conducted via the point-to-point network which has a higher latency. The exact algorithm deployed in this case depends on the message size and the shape of the communicator, e.g., whether it is rectangular. In our case the shape of the communicator guarantees that optimal algorithms, which, e.g., use deposit bits, are executed in the point-to-point network. Consequently, the latency is only slightly higher compared to the latency of the `MPI_COMM_WORLD` communicator (3.35 µs, plus 90 ns per hop versus 2.5 µs) on the faster network.

Once the communicators have been set up the master of subgroup 0 becomes the super-master. At program initialization, each master process immediately starts computations on bootstrap replicates or ML searches without communicating with the super-master. Every time a master has completed the computations on a tree it sends a message to the super-master and locally stores the tree in a list. This message contains the number of trees that have been computed so far by this specific master. Every time the super-master receives such a message it checks if the total number of trees specified by the user (20 in our example) has already been computed. If that is the case, the super-master sends a termination message to all other master processes. When a master receives the termination message it sends all locally stored trees to the super-master, which prints them to file. Thereafter, each master terminates along with the respective worker processes. When all tree topologies have been written to file, the super-master exits as well. The above modification avoids the perturbation of fine-grained work scheduling at the super-master, since the actual tree topologies are only sent at the end of the inference process.

Figure 2 outlines the example setup with 4 masters that use 3 worker processes each for ML computations.

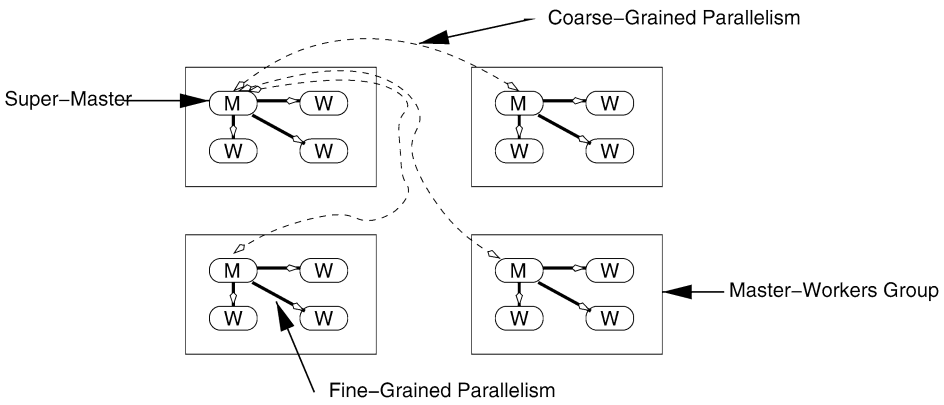

Fig. 2. Hybrid MPI/MPI parallelization of RAxML on BlueGene/L.

Thick black arrows indicate frequent fine-grained ML communications within each master–worker group. Dotted thin lines show the infrequent and less time-critical communications between the super-master and the remaining masters.

4. Experimental setup and results

In this section, we describe the experimental setup and platforms used (Section 4.1). We also provide performance data for the fine-grained parallelization in Section 4.2 as well as for the hybrid parallelization (Section 4.3). Finally, in Section 4.4 we report execution times for production-level analyses on the complete biological datasets.

4.1. Experimental setup

To test the scalability of our parallelization we used two large and challenging real-world datasets as well as subsets thereof:

1. A multi-gene alignment of 2,182 mammalian sequences with 51,089 base-pairs that comprises data from 67 different genes. Despite the fact that a few ML trees could already be computed with RAxML on a 4-way AMD Opteron, the execution times do not allow for a full bootstrap analysis. Large-scale analyses of mammalian phylogenies have recently received considerable attention (including the popular press), since they can be used, e.g., to date the rise of present-day mammals [3].
2. A matrix of 566,470 polymorphic data points for 270 taxa.[2] This dataset contains genotype data for 283,235 non-redundant SNPs on the human chromosome 1 in sorted order for 270 individuals in the HapMap project [8]. Each column in the matrix is occupied by nucleotides A, C, G, T. For each SNP each individual contributes a pair of consecutive nucleotide columns (1 each from their paternal and maternal chromosome, sorted alphabetically within each pair). Thus, for 283,235 SNPs the total sequence alignment length is 566,470 per taxa. A phylogenetic analysis of this dataset represents a novel approach where individual humans are represented as leaves on a phylogenetic tree that is reflective of their ancestral history. The combination of genotypic and phenotypic data on phylogenetic trees makes new types of correlative inference possible that differ from standard linear approaches [15]. Phylogenetic trees inferred on HPC systems can be combined and correlated with publicly available biological phenotype data from the same human samples [30].

[2] A taxon is a name designating an organism or group of organisms.

In order to test scalability on various dataset-sizes we extracted appropriate sub-alignments from the above datasets. From the mammalian alignment we extracted sub-alignments containing 50 sequences with 5,000 and 50,000 base-pairs each. In addition, we extracted alignment samples with 50 as well as 250 sequences and a length of 500,000 base-pairs from the 270 sequence HapMap dataset. Note, that usually not all columns in such alignments are unique. Therefore, most programs for phylogenetic inference compress the input datasets and discard redundant columns by assigning a respective higher weight to column patterns that appear more than once. Consequently, the number of distinct patterns m' in our test datasets was lower than the numbers stated above which refer to the number of columns m. The number of distinct patterns (actual length of the compute-intensive `for`-loops) are 3,066, 23,385 and 216,025 for the 50 sequences sub-alignments, and 403,581 for the 250 sequences dataset respectively.

The scalability of our approach was assessed on the following systems:

- The InfiniBand Cluster at Technische Universität München: A Linux cluster consisting of 32 4-way AMD 2.4 GHz Opteron 850 processors with 8 GB of main memory per node which are interconnected by Mellanox Technologies MT23108 Infiniband host channel adapters and an MTEK43132 Infiniband switch.
- The BlueGene/L system at Iowa State University: A one-rack machine with 1,024 nodes (2,048 CPUs) and a peak performance of 5.734 teraflops.
- The HLRB2 (Bavarian Supercomputer System) at Leibniz Rechenzentrum: An SGI Altix 4700 system with a total of 9,728 Intel Itanium2 Montecito cores, an aggregated peak performance of 62.3 teraflops and 39 terabyte of main memory.

Since RAxML uses randomized algorithms for the creation of starting trees and for bootstrapping, the run-times as well as the results of the tree search differ slightly among individual program runs. In order to obtain reproducible results and run-times, we used a fixed

seed for both random number generators (bootstrapping and computation of starting trees).

4.2. Scalability of fine-grained parallelism

We provide speedup graphs for the fine-grained parallelization on all three HPC systems for increasing dataset sizes in Figs 3–7. Absolute run-times for all three systems are given in Tables 1–3, respectively.

Plots 3 and 4 depict speedups for 50-sequence subsets of the mammalian alignment consisting of 3,066 and 23,385 alignment patterns, respectively. The comparatively poor performance for more than 15 workers in Fig. 3 can be explained by the relatively small prob-

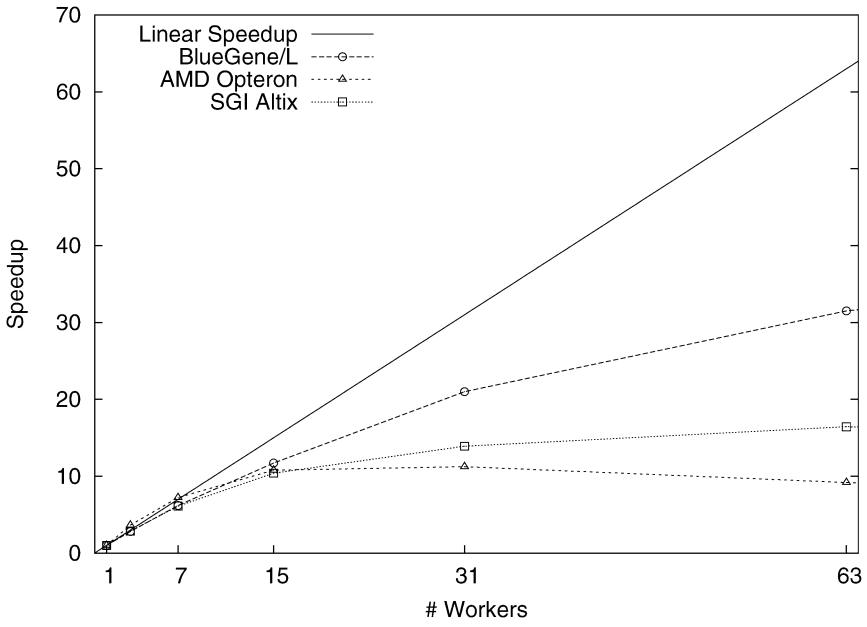

Fig. 3. Speedup on 50 sequences with 3,066 distinct patterns.

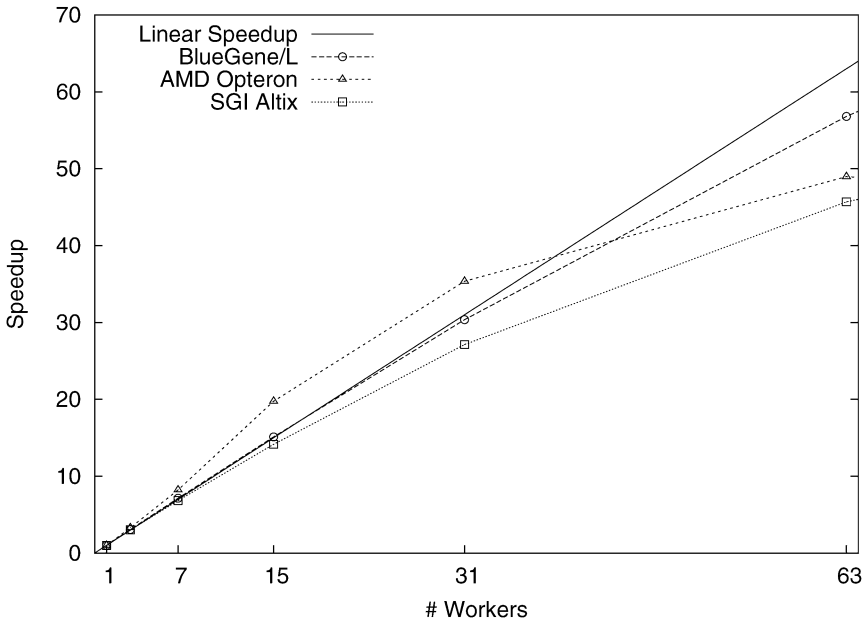

Fig. 4. Speedup on 50 sequences with 23,385 distinct patterns.

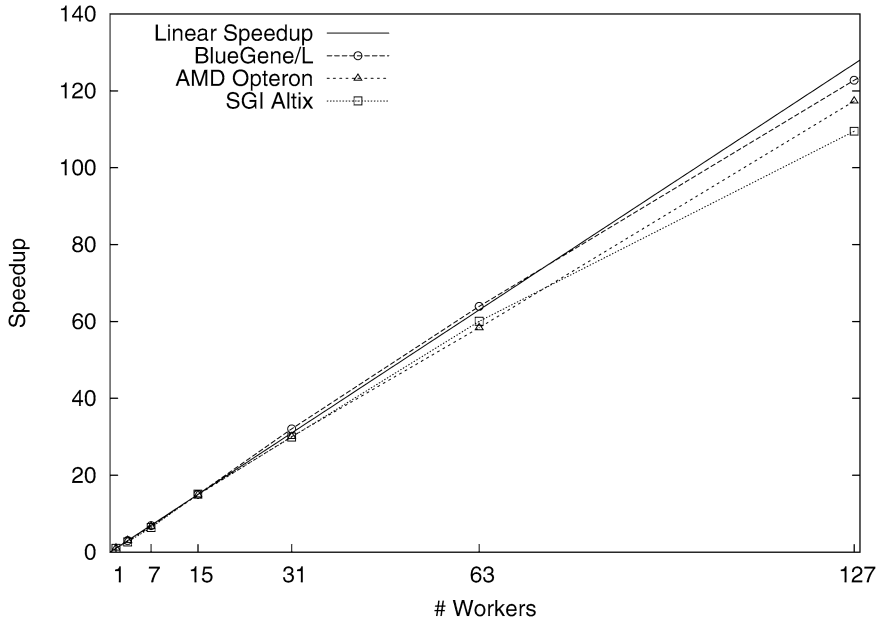

Fig. 5. Speedup on 50 sequences with 216,025 distinct patterns.

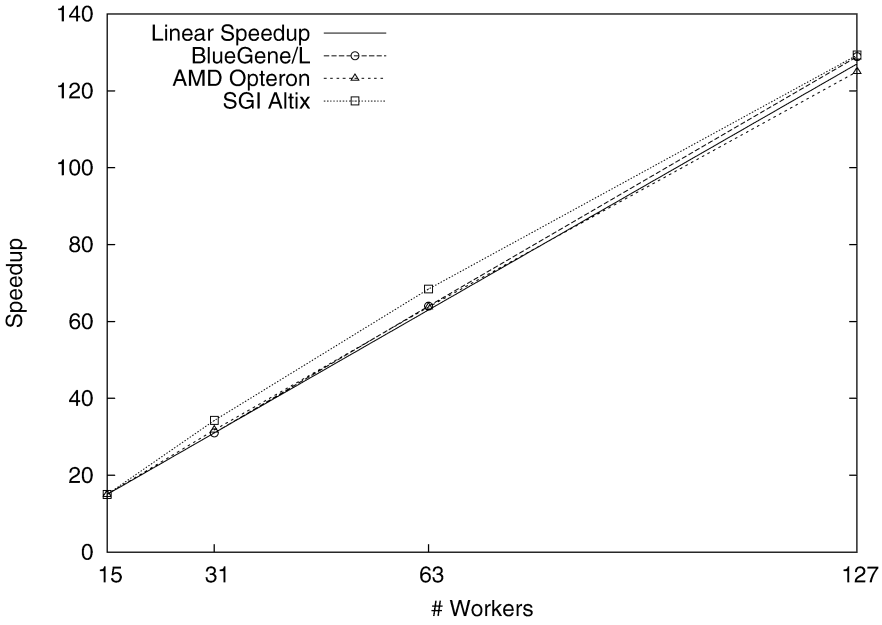

Fig. 6. Speedup on 250 sequences with 403,581 distinct patterns on 15–127 workers.

lem size. Note that, performance on the Opteron cluster is slightly super-linear up to 31 worker processes in Fig. 4 due to increased cache efficiency. Another observation is that BG/L scales significantly better for this setting with more than 63 workers. However, this was expected as the BlueGene system provides a better communication to computation ratio due to the low latency network coupled with moderate per CPU computing power. Therefore, BG/L scales considerably better, even on small problem sizes.

In Fig. 5 we depict speedup values for a 50-taxa haplotype subset with 216,025 distinct patterns. Since the `for`-loops for this alignment are longer by one order of magnitude, the communication to computation ra-

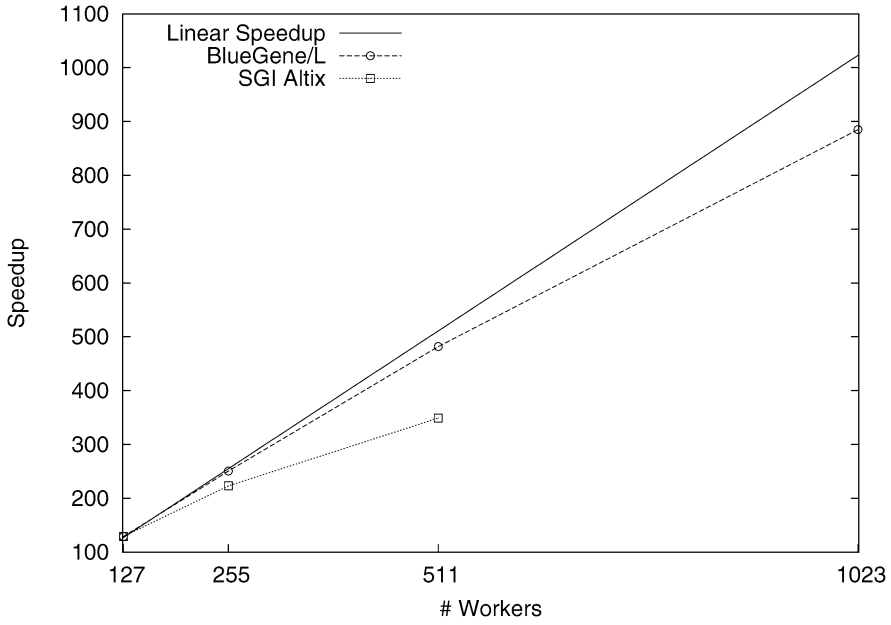

Fig. 7. Speedup on 250 sequences with 403,581 distinct patterns on 127–1,023 workers.

Table 1
Absolute run-times on BlueGene/L (in seconds)

# SEQS	# BP	# Workers									
		1	3	7	15	31	63	127	255	511	1,023
50	3,066	1,400	498	226	120	67	44	33	31	29	29
50	23,385	14,326	4,653	2,008	948	472	252	142	87	60	48
50	216,025		32,659	14,187	6,531	3,055	1,533	798	436	256	169
250	403,581					145,739	70,617	35,056	18,025	9,375	5,105

Table 2
Absolute run-times on SGI Altix 4700 (in seconds)

# SEQS	# BP	# Workers								
		1	3	7	15	31	63	127	255	511
50	3,066	672	235	110	65	49	41	46		
50	23,385	6,663	2,205	975	471	246	146	101		
50	216,025	46,353	17,589	7,228	3,074	1,547	772	423		
250	403,581				135,544	59,340	29,716	15,725	9,079	5,815

Table 3
Absolute run-times on AMD Opteron (in seconds)

# SEQS	# BP	# Workers						
		1	3	7	15	31	63	127
50	3,066	664	182	92	62	59	72	120
50	23,385	7,706	2,332	937	390	218	157	181
50	216,025	51,569	17,084	7,766	3,462	1,717	884	440
250	403,581				192,591	90,901	45,306	23,111

tio improves roughly by a factor of 10. As a result, the scalability on all three platforms is almost linear up to 63 workers. For 127 workers, the speedup slows down for all architectures. However, the BlueGene still shows a speedup of 123 which corresponds to an efficiency of approx. 97%. In Figs 6 and 7 we show how the program scales on 250 haplotype sequences with 403,581 distinct patterns, up to 1,024 CPUs. Note that, plots 6 and 7 provide relative scalability compared to a run with 15 workers. This is due to the fact that we were not able to execute the program with a smaller number of workers because of memory shortage on the Opteron and BG/L systems. However, as shown in Figs 3 and 4, the program scales linearly for up to 15 workers even for datasets with a smaller number of distinct patterns. Interestingly, the absolute run-times for the 250 sequences dataset on the SGI Altix are almost two times lower than on the Opteron system while for the 50 sequence datasets the absolute run-times are almost identical. Furthermore, the Altix shows a slight super-linear speedup up to 127 workers. Evidently, the effect of the approximately 9 times larger caches on the Itanium only becomes apparent after a certain alignment size threshold.

In general, one can conclude that the scalability of fine-grained parallelism directly depends on the length of the alignment or more precisely on the number of distinct patterns m', because the computation/communication ratio increases with the length of the likelihood vectors. The number of sequences and thus the number of nodes in the tree show a less prevalent impact on performance. One reason for this is that the computation of partial likelihood vectors benefits from the increased number of internal nodes, as more recursive likelihood vector updates can be aggregated and communicated via one single broadcast and thus in turn improve upon the computation/communication ratio. However, the total number of branches whose lengths need to be optimized also increases with the number of sequences. This function is comparatively costly with respect to communication because of the frequent reduction operations which need to be performed at every iteration of the Newton–Raphson procedure. Furthermore, the cost of these reduction operations also increases with the number of processes involved. In the final analysis, the positive and negative performance impact caused by increasing the number of sequences is balanced.

As already mentioned in Section 3.1, in some situations it makes sense to soften the strict master/worker paradigm and to perform likelihood-computations at the master as well. Therefore, we modified the program such that the master can also execute the worker-code. We tested this modification on the AMD Opteron system with the 50 sequences datasets. Table 4 shows how much workload can be handled by the master without slowing down the other worker processes. The values represent the master workload compared to the workload at each worker process, i.e., simply the ratio of the number of likelihood vector entries computed at the master and the workers. For example, 2 processors and 100% indicate that the workload (fraction of likelihood vector entries computed) was equally split between the master and the single worker. 90% and 16 processes means that the master performed 5.63% of the likelihood computations while the worker processes performed 6.29% each.

Table 4
Maximum workload at master on AMD Opteron

# SEQS	# BP	# Processors						
		2	4	8	16	32	64	128
50	3,066	100%	100%	95%	90%	90%	80%	60%
50	23,385	100%	95%	95%	90%	90%	90%	70%
50	216,025	100%	100%	95%	90%	90%	95%	90%

The results show that up to 8 processors, 95% of worker computations can easily be handled by the master process. Up to 90% can be performed by the master if he has to orchestrate up to 31 workers. Even with 127 workers over 60% can be offloaded to the master. However, this corresponds to only 0.5% of the total workload and would consequently result in a run-time reduction of only 0.5%. Hence, it only makes sense to perform likelihood computations at the master if the worker count is low.

4.3. Scalability of hybrid parallelism

We assessed the performance of the coarse-grained parallelization using the mammalian dataset with 50 sequences and 23,385 distinct alignment patterns. The experiments were conducted on BlueGene/L partitions of 32, 128 and 512 nodes.

Figure 8 shows execution times for individual tree inferences using groups of 8, 16, 32, 64 and 128 nodes. The straight black line shows the time for a single master–worker group (see Table 1). The remaining three graphs show execution times for multiple master–worker groups on the aforementioned BG/L partitions which have been split into 4, 8, 16, 32 and 64 groups (where applicable) using `MPI_Comm_split` as described in Section 3.2.

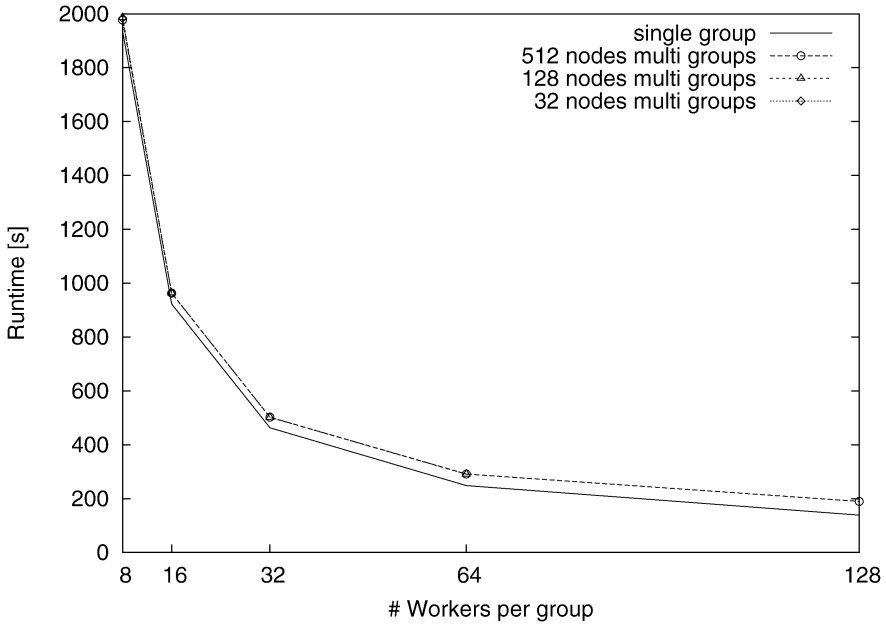

Fig. 8. Execution times of multiple groups setup with 50 sequences and 23,385 distinct patterns.

As expected, the run-times observed for the multiple group configuration are slightly higher than the corresponding run-times of one single master–worker group. This is due to the fact that, in multi-group configurations all messages are sent over the higher latency peer-to-peer network while a single group can utilize the faster specialized collective network (see Section 3.2). This is a limitation of the MPI implementation on BG/L which will probably be corrected in future versions.

Figure 8 shows that the total number of distinct groups does not influence the run-time of the individual tree searches. This underlines that communication between masters and super-master is infrequent and does thus not affect the fine-grained parallelism within each group.

Figure 9 provides the total run-times for 32 distinct tree searches on 32, 128 and 512 nodes. The nodes have been split into groups of 16 and 32 nodes. For example in the case of 512 nodes with 32 nodes per group, 16 masters with a private set of 31 workers each, perform 32 distinct tree searches in parallel, i.e., two per group. The plot shows that the total execution time decreases linearly with an increasing number of total nodes used for computation. Furthermore, one can see that, as expected, by considering the absolute run-times for single groups in Table 1, groups of 16 nodes perform slightly better than groups of 32 nodes.

4.4. Full haplotype data analysis

We used the BlueGene/L system to perform a full phylogenetic analysis of the haplotype alignment (212 sequences, 566,470 base pairs). To the best of our knowledge, this represents the largest dataset with respect to input matrix size, analyzed under ML to date. Initially, we conducted ML-searches for the best-scoring tree with 1,024 CPUs (1,024 nodes in co-processor mode) in single-group configuration. We were able to complete 7 distinct tree searches within 14 hours. Afterwards we performed a bootstrap analysis on 2,048 CPUs in virtual node mode which were divided into 8 individual master–worker groups (256 CPUs per group). It took 26 hours to infer 64 bootstrapped trees.

In addition, we used the comprehensive haplotype alignment to test the scalability of our approach in virtual node mode. The average run-time of the aforementioned 7 tree searches in co-processor mode is 6,838 seconds. The average execution time on 512 nodes in virtual node mode, i.e. 1,024 CPUs, amounts to 7,000 seconds (efficiency: 97.69%).

The above results demonstrate that a full real-world analysis of challenging datasets is feasible on current supercomputer architectures like the IBM BlueGene/L or the SGI Altix. As we have shown in our experiments, fine-grained parallelism can be used to efficiently exploit hundreds of CPUs. Given the fact that

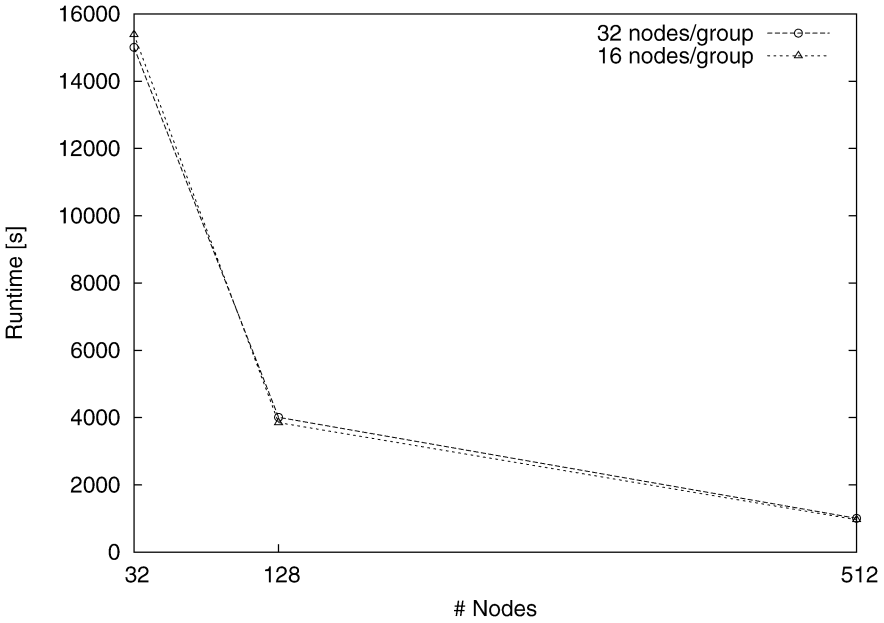

Fig. 9. Execution times for 32 distinct tree searches with 50 sequences and 23,385 distinct patterns.

alignments used for phylogenetic analyses continue to grow and that real-life problems require hundreds if not thousands of individual tree searches and bootstrap runs, our approach has the potential for efficient exploitation of forthcoming petascale systems.

5. Biological results

In the current section we briefly describe the biological results and insights that have been attained from the large-scale phylogenetic inference of the haplotype map dataset on the BG/L.

As already mentioned, there is a rapid growth in available sequence data as well as improvement of sequencing techniques, which will create a vast wealth of genome sequences aimed at personalized medical care. These sequences afford opportunities for analyses not only across biological species, i.e., "classic" phylogenetic inference, but also within populations potentially yielding important clues about genetic variation that accounts for inter-individual differences, for instance, between geographically distinct populations or between healthy and afflicted individuals.

As outlined in the preceding section we analyzed a huge phylogenetic tree based on SNP data from hundreds of individual humans from around the world. The utility of a phylogenetic approach is that it provides a mechanism for inferring genotype–phenotype correlation that explicitly accounts for the natural interrelatedness of species and individuals in a population. In contrast, most other current studies of phenotype–genotype correlation use statistical approaches (i.e., regression) that incorrectly assume that the individuals are entirely independent samples. We previously demonstrated the potential of a phylogenetic approach in finding meaningful genotype–phenotype correlations, e.g., loci (certain regions of the genome) associated with susceptibility to infection by the bacteria that causes Anthrax using comparative SNP data from strains of mice [15].

Here we extend this approach to combine genetic information from hundreds of individuals worldwide enabling possible future analysis of continuous phenotypic data such as microarray gene expression analysis, blood chemistry, longevity and disease susceptibility.

5.1. Data acquisition and assembly details

Genotype data for a set of 283,235 non-redundant SNPs located on the human chromosome 1 for 270 human individuals in the HapMap project [8] were downloaded from www.hapmap.org/genotypes/2007-03/fwd_strand/non-redundant and assembled into a phylogenetic data matrix in the order of their physical position on the chromosome. 60 taxa that represented children were removed from the alignment because of their level of genetic redundancy. Analogous

data for the chimpanzee and macaque for each corresponding SNP position was obtained from the UCSC dbSNP Build 126 data [18]. Their data was aligned with the human HapMap data as outgroups via Perl scripts based on the respective SNP identifiers. Outgroups, i.e., sequences of organisms that are not located within the group of species that shall be analyzed, are used to root phylogenetic trees. This large dataset was then used for the analyses described in Section 4.4.

5.2. *Biological insights*

The optimization of the character describing the geographic locale of the individuals onto the tree using the Mesquite software package (Maddison and Maddison, www.mesquiteproject.org) confirmed the anticipated geographic clustering of various individuals as depicted in Fig. 10.

The mixture noted between Chinese and Japanese branches may reflect the close geographic as well as migratory history of these populations. Our approach yields a biologically reasonable and meaningful phylogenetic tree for geographically distinct human samples that supports an African origin of *Homo Sapiens* [7].

6. Conclusion and future work

We have presented a generally applicable parallelization strategy for the phylogenetic ML function. In addition, we have demonstrated that our approach scales well up to 1,024 processors on the IBM BlueGene/L, up to 256 processors on the SGI Altix 4700, and up to 128 CPUs on a common Linux cluster architecture. Due to the ability to handle and scale on large datasets, the presented parallelization scheme opens up new possibilities with respect to the inference of large-scale phylogenies and provides a viable solution for future computational needs in phylogenetics.

To emphasize the practical relevance of our work and to show that datasets which require supercomputing power already exist, we inferred a phylogeny for hundreds of human samples using genetic information for chromosome 1, the largest human chromosome.

In addition, the current chromosome 1 analysis on the BG/L shows that it is computationally and methodologically feasible as well as insightful to compute phylogenetic intra-population trees for large samples of the human population. These trees are of interest to study human origins and migration. Moreover,

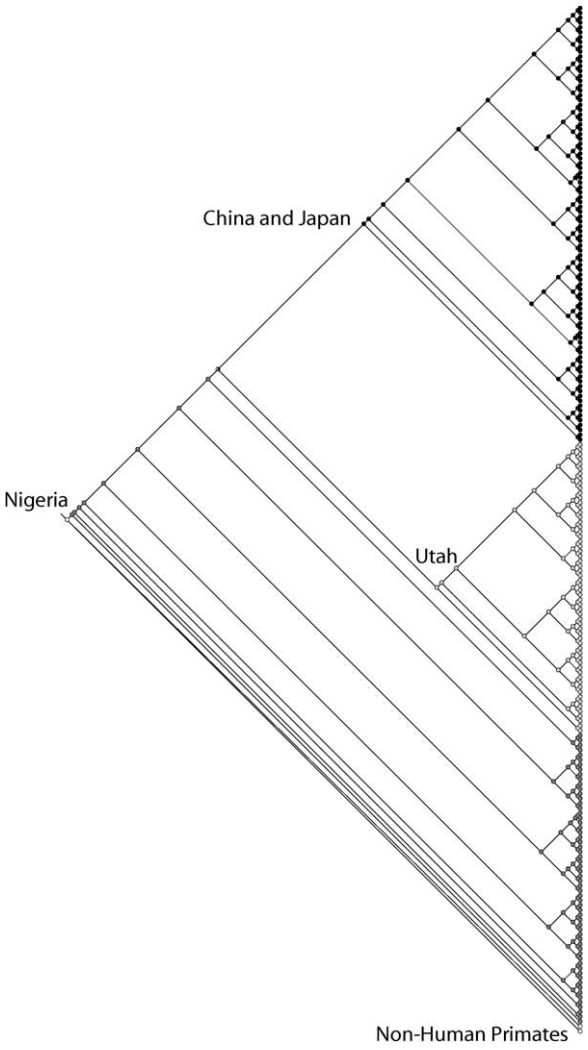

Fig. 10. Phylogenetic tree inferred from haplotype dataset. White circles – non-human primates (chimpanzee, macaque); grey circles – humans of Yoruban descent from Ibadan, Nigeria; light gray circles – humans of Northern and Western European descent from Utah; black circles – humans of Han Chinese descent from Beijing, China and humans of Japanese descent from Tokyo, Japan.

in conjunction with measurements of human biomedical science (e.g. cholesterol levels) for large sets of individuals, these trees can enable new methodologies in searching for genes that contribute to important human differences. Publicly available measurements (e.g., microarray analysis [30]) for the samples analyzed here exist, suggesting potential future analyses. Due to its phylogenetic nature, this type of analysis acknowledges interrelationships among the individuals and thus differs from standard statistical approaches [15] that ignore this inherent property of biological data [13].

Current efforts are underway to generate genome-wide SNP datasets by sampling a diverse ethnogeographic dataset, as well as to sequence more than 1,000 human genomes for public release. Thus, analytical approaches such as the one we present here are increasingly relevant. The next step, in the context of large-scale phylogenetic analyses, is to approach whole genome comparisons, and to examine issues of recombination by separate analysis of autosomes, sex-linked chromosomes, or mitochondrial genomes.

Finally, current technical work covers the integration of the parallelization scheme developed in this paper into the most recent standard release of RAxML, which features a large variety of nucleotide substitution models and a novel rapid bootstrapping algorithm that further accelerates the phylogenetic analysis process by one order of magnitude via algorithmic means.

Acknowledgments

We would like to thank Olaf Bininda-Emonds at the University of Jena for providing us his sequence data to assess RAxML performance.

This research is supported in part by the National Science Foundation under CNS-0521568 and CCF-0431140. D.J. and A.D.J. would like to acknowledge the support of the Department of Biomedical Informatics in the School of Biomedical Sciences at the Ohio State University College of Medicine. The Exelixis Lab (AS) is funded under the auspices of the Emmy-Noether program by the German Science Foundation (DFG).

References

[1] D. Bader, B. Moret and L. Vawter, Industrial applications of high-performance computing for phylogeny reconstruction, in: *Proc. of SPIE ITCom*, Denver, CO, Vol. 4528, 2001, pp. 159–168.

[2] D. Bader, U. Roshan and A. Stamatakis, Computational grand challenges in assembling the tree of life: Problems and solutions, in: *Advances in Computers*, Vol. 68, Elsevier, Seattle, WA, 2006.

[3] O.R.P. Bininda-Emonds, M. Cardillo, K.E. Jones, R.D.E. MacPhee, R.M.D. Beck, R. Grenyer, S.A. Price, R.A. Vos, J.L. Gittleman and A. Purvis, The delayed rise of present-day mammals, *Nature* **446** (2007), 507–512.

[4] 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.

[5] F. Blagojevic, D.S. Nikolopoulos, A. Stamatakis and C.D. Antonopoulos, RAxML-cell: Parallel phylogenetic tree inference on the cell broadband engine, in: *Proc. of IPDPS 2007*, Long Beach, CA, 2007.

[6] B. Chor and T. Tuller, Maximum likelihood of evolutionary trees: hardness and approximation, *Bioinformatics* **21**(1) (2005), 97–106.

[7] J.D. Clark, Y. Beyene, G. WoldeGabriel, W.K. Hart, P.R. Renne, H. Gilbert, A. Defleur, G. Suwa, S. Katoh, K.R. Ludwig, J.-R. Boisserie, B. Asfaw and T.D. White, Stratigraphic, chronological and behavioural contexts of Pleistocene Homo sapiens from Middle Awash, Ethiopia, *Nature* **423** (2003), 747–752.

[8] T.I.H. Consortium, The International HapMap Project, *Nature* **426** (2003), 789–796.

[9] T.Z. DeSantis, P. Hugenholtz, N. Larsen, M. Rojas, E.L. Brodie, K. Keller, T. Huber, D. Dalevi, P. Hu and G.L. Andersen, Greengenes, a chimera-checked 16S rRNA gene database and workbench compatible with ARB, *Appl. Environ. Microbiol.* **72**(7) (2006), 5069–5072.

[10] Z. Du, F. Lin and U. Roshan, Reconstruction of large phylogenetic trees: a parallel approach, *Comput. Biol. Chem.* **29**(4) (2005), 273–280.

[11] J. Felsenstein, Evolutionary trees from DNA sequences: a maximum likelihood approach, *J. Mol. Evol.* **17** (1981), 368–376.

[12] G.W. Grimm, S.S. Renner, A. Stamatakis and V. Hemleben, A nuclear ribosomal DNA phylogeny of acer inferred with maximum likelihood, splits graphs, and motif analyses of 606 sequences, *Evol. Bioinf. Online* **2** (2006), 279–294.

[13] A. Grupe, S. Germer, J. Usuka, D. Aud, J. Belknap, R. Klein, M. Ahluwalia, R. Higuchi and G. Peltz, *In silico* mapping of complex disease-related traits in mice, *Science* **292**(5523) (2001), 1915–1918.

[14] S. Guindon and O. Gascuel, A simple, fast, and accurate algorithm to estimate large phylogenies by maximum likelihood, *Syst. Biol.* **52**(5) (2003), 696–704.

[15] F. Habib, A.D. Johnson, R. Bundschuh and D. Janies, Large scale genotype–phenotype correlation analysis based on phylogenetic trees, *Bioinformatics* **23** (2007), 785–788.

[16] M. Heinicke, W. Duellman and S. Hedges, From the cover: Major Caribbean and Central American frog faunas originated by ancient oceanic dispersal, *Proc. Natl. Acad. Sci.* **104**(24) (2007), 10092.

[17] D. Janies, A.W. Hill, R. Guralnick, F. Habib, E. Waltari and W.C. Wheeler, Genomic analysis and geographic visualization of the spread of avian influenza, *Syst. Biol.* **56**(2) (2007), 321–329.

[18] D. Karolchik, R. Baertsch, M. Diekhans, T. Furey, A. Hinrichs, Y. Lu, K. Roskin, M. Schwartz, C. Sugnet, D. Thomas et al., The UCSC genome browser database, *Nucleic Acids Res.* **31**(1) (2003), 51–54.

[19] R.E. Ley, J.K. Harris, J. Wilcox, J.R. Spear, S.R. Miller, B.M. Bebout, J.A. Maresca, D.A. Bryant, M.L. Sogin and N.R. Pace, Unexpected diversity and complexity of the guerrero negro hypersaline microbial mat, *Appl. Envir. Microbiol.* **72**(5) (2006), 3685–3695.

[20] B. Minh, L. Vinh, A. Haeseler and H. Schmidt, pIQPNNI: parallel reconstruction of large maximum likelihood phylogenies, *Bioinformatics* **21**(19) (2005), 3794–3796.

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

[22] D.S. Nikolopoulos, T.S. Papatheodorou, C.D. Polychronopoulos, J. Labarta and E. Ayguad, Is data distribution necessary in openmp?, in: *Proc. Supercomputing*, Dallas, TX, 2000.

[23] C. Robertson, J. Harris, J.R. Spear and N. Pace, Phylogenetic diversity and ecology of environmental Archaea, *Curr. Opin. Microbiol.* **8** (2005), 638–642.

[24] A. Stamatakis, Distributed and parallel algorithms and systems for inference of huge phylogenetic trees based on the maximum likelihood method, PhD thesis, Technische Universität München, Germany, October 2004.

[25] A. Stamatakis, RAxML-VI-HPC: maximum likelihood-based phylogenetic analyses with thousands of taxa and mixed models, *Bioinformatics* **22**(21) (2006), 2688–2690.

[26] A. Stamatakis, T. Ludwig and H. Meier, Parallel inference of a 10,000-taxon phylogeny with maximum likelihood, in: *Proc. of Euro-Par 2004*, Pisa, Italy, September 2004, pp. 997–1004.

[27] A. Stamatakis, T. Ludwig and H. Meier, RAxML-III: A fast program for maximum likelihood-based inference of large phylogenetic trees, *Bioinformatics* **21**(4) (2005), 456–463.

[28] A. Stamatakis, M. Ott and T. Ludwig, RAxML-OMP: An efficient program for phylogenetic inference on SMPs, in: *Proc. PaCT*, Krasnoyarsk, Russia, 2005, pp. 288–302.

[29] C. Stewart, D. Hart, D. Berry, G. Olsen, E. Wernert and W. Fischer, Parallel implementation and performance of fastDNAml – a program for maximum likelihood phylogenetic inference, in: *Proc. of SC 2001*, Denver, CO, November 2001.

[30] B.E. Stranger, M.S. Forrest, A.G. Clark, M.J. Minichiello, S. Deutsch, R. Lyle, S. Hunt, B. Kahl, S.E. Antonarakis, S. Tavare, P. Deloukas and E.T. Dermitzakis, Genome-wide associations of gene expression variation in humans, *PLoS Gene* **1**(6) (2005), 695–704.

[31] Z. Yang, Maximum likelihood phylogenetic estimation from DNA sequences with variable rates over sites, *J. Mol. Evol.* **39** (1994), 306–314.

[32] D. Zwickl, Genetic algorithm approaches for the phylogenetic analysis of large biological sequence datasets under the maximum likelihood criterion, PhD thesis, University of Texas at Austin, April 2006.