PBPI: a High Performance Implementation of Bayesian Phylogenetic Inference

Xizhou Feng[1] Kirk W. Cameron[1] Duncan A. Buell[2]

[1]Scalable Performance Laboratory
Department of Computer Science
Virginia Polytechnic Institute and State University
Blacksburg, VA 24060, USA
{fengx, Cameron}@cs.vt.edu

[2]Department of Computer Science and Engineering
University of South Carolina
Columbia, SC 29208, USA
buell@sc.edu

Abstract

This paper describes the implementation and performance of PBPI, a parallel implementation of Bayesian phylogenetic inference method for DNA sequence data. By combining the Markov Chain Monte Carlo (MCMC) method with likelihood-based assessment of phylogenies, Bayesian phylogenetic inferences can incorporate complex statistic models into the process of phylogenetic tree estimation. However, Bayesian analyses are extremely computationally expensive. PBPI uses algorithmic improvements and parallel processing to achieve significant performance improvement over comparable Bayesian phylogenetic inference programs. We evaluated the performance and accuracy of PBPI using a simulated dataset on System X, a terascale supercomputer at Virginia Tech. Our results show that PBPI identifies equivalent tree estimates 1424 times faster on 256 processors than a widely-used, best-available (albeit sequential), Bayesian phylogenetic inference program. PBPI also achieves linear speedup with the number of processors for large problem sizes. Most importantly, the PBPI framework enables Bayesian phylogenetic analysis of large datasets previously impracticable.

1. Introduction

All life on the earth, both present and past, are believed to be descended from a common ancestor. The descending pattern or evolutionary relationship among species or organisms, or the relatedness of their genes, is usually described by a phylogeny, a tree or network structure, with edge length representing the evolutionary divergence along different lineages.

Since all biological phenomena are the result of evolution, most biological studies have to be conducted in the light of evolution and require information on phylogeny to interpret data. Thus, phylogenies play important roles not only in evolutionary biology, genetics and genomics, but also in modern pharmaceutical research, drug discovery, agricultural plant improvement and disease control studies. The importance of phylogenetics has never been made more clear than by the ambitious “Tree of Life” project initiated by the US National Science Foundation, which aims to assemble a phylogeny for all forms of life on the earth [1].

The task of phylogenetic inference includes two major steps: 1) constructing a phylogenetic tree that maps the evolutionary relationship among a group of taxa, and 2) accessing the confidence on the estimated tree given the observed data. Various methods are available for building the phylogenetic tree and some of them are based on a probabilistic model of molecular evolution.

Bayesian phylogenetic inference and maximum likelihood estimation are two major methods that explicitly use a model of molecular evolution and a formalization of the likelihood function. Though there exist several debating issues [2, 3] for Bayesian phylogenetic inference , Bayesian phylogenetic inference gained wide acceptance [4, 5] after it was introduced. In a recent comparative survey [6] of widely-used state-of-the-art statistically phylogenetic programs, a Bayesian-based phylogenetic inference program, MrBayes [7] consistently outperforms other tested programs in terms of speed and tree quality.

However, Bayesian inference of large phylogeny is a computationally intensive process both in computation time and memory requirements. Consider a realistic problem: estimating the phylogeny of 200 aligned amino acid sequences with 3500 characters using a model that allows five different rates across sites. Solving such a problem requires several gigabytes of memory space and several months of computing time using current Bayesian phylogenetic inference programs. Hence a high

SC2006 November 2006, Tampa, Florida, USA
0-7695-2700-0/06 $20.00 ©2006 IEEE

performance implementation is needed to fully exploit the power of Bayesian phylogenetic inference.

There are some inherent challenges in designing a high performance implementation for Bayesian phylogenetic inference. First, Bayesian phylogenetic inference is founded on the Markov Chain Monte Carlo (MCMC) method which is essentially a sequential method where state at the current time step depends on previous time steps. Second, Bayesian phylogenetic inference requires frequent I/O operations to store the samples drawn by the MCMC method. Third, even when the multiple-chain MCMC method is used, the degree of parallelism in Bayesian phylogenetic inference is limited.

Our contributions: In this paper, we introduce a new implementation of Bayesian phylogenetic inference called **PBPI** (**P**arallel **B**ayesian **P**hylogenetic **I**nference) which improves the performance of Bayesian phylogenetic inference through the combination of algorithmic improvements and parallel processing. PBPI achieves a speedup of 19.1 over MrBayes on a single processor. Overall PBPI achieved up to 874 times speedup over the sequential version of MrBayes. PBPI also achieved a speedup of 46.1 on 64 processors of System X at Virginia Tech and larger problems yield larger speedup.

This paper is organized as follows. In the next section, we provide a brief description of Bayesian phylogenetic inference methods. The implementation of PBPI is discussed in Section 3. Validation and verification of our approach is given in Section 4. Next, we present a performance evaluation for PBPI in Section 5. We describe related work in Section 6 followed by conclusions and future work.

2. Bayesian phylogenetic inference

2.1 The posterior probabilities of phylogenetic trees

Due to the stochastic nature and complicated mechanisms in molecular evolution, phylogenetic inferences have to deal with numerous uncertainties such as unknown parameters and multiple phylogenetic trees. Bayesian phylogenetic inference provides a methodology to consider all unknown variables in a comprehensive probabilistic model and to estimate the phylogenetic tree based on a quality called posterior probability.

Given a molecular sequence alignment (i.e. an observed data) D, the objective of a Bayesian phylogenetic inference is to estimate a phylogenetic model $\Psi = (T, \tau, \theta)$ that best interprets the data. Here, the phylogenetic model $\Psi = (T, \tau, \theta)$ consists of three components: a tree structure (T) that represents the evolutionary patterns for the organism under study, a vector of branch lengths (τ) which maps the divergence time along different lineages, and a model of the molecular evolution (θ) that approximates how the characters at each site evolve over time along the tree.

In the Bayesian framework, both the observed data D and the phylogenetic model Ψ are treated as random variables. Next, the joint distribution of the data can be modeled as

$$P(D, \Psi) = P(D \mid \Psi) P(\Psi) \quad (1)$$

Once the data is known, Bayesian theory can be used to compute the posterior probability for a specific phylogenetic model Ψ_i using

$$P(\Psi_i \mid D) = \frac{P(D \mid \Psi_i) P(\Psi_i)}{\sum_j \left(P(D \mid \Psi_j) P(\Psi_j) \right)} \quad (2)$$

Here, $P(D \mid \Psi_i)$ is called the likelihood (the probability of the data under the model Ψ_i), $P(\Psi_i)$ is called the prior probability of the model (the unconditional probability of the model without any knowledge of the observed data), and $P(D) = \sum_j \left(P(D \mid \Psi_j) P(\Psi_j) \right)$ is the unconditional probability of the data. Since phylogenetic models are hypotheses about how the data will evolve, both the prior and the posterior of a specific phylogenetic model should be interpreted as a confidence interval for the model instead of explained as frequencies.

This distribution of $P(D \mid \Psi)$ is the basis of Bayesian phylogenetic inference; a lot of useful information can be obtained from this distribution. For example, we can estimate the posterior probability of a specific phylogenetic tree topology by calculating the marginal distribution as

$$P(T_i \mid D) = \iint P(T_i, \tau, \theta \mid D) d\tau d\theta . \quad (3)$$

2.2 Approximation of the posterior probability using MCMC

The posterior distribution can be approximated by a class of methods called Markov Chain Monte Carlo

Initialization phase: $t \leftarrow 0$; $\Psi^{(0)} \leftarrow \Psi_0$
While $t \leq \max imum - generation$
  Draw a proposal Ψ from $q(\bullet \mid \Psi^{(t)})$
  Compute the acceptance $\alpha(\Psi^{(t)}, \Psi)$
  Draw a random variable u from $U(0,1)$
  If $u \leq \alpha(\Psi^{(t)}, \Psi)$
    Then $\Psi^{(t+1)} \leftarrow \Psi$
    Else $\Psi^{(t+1)} \leftarrow \Psi^{(t)}$
  $t \leftarrow t+1$

Figure 1: The Metropolis-Hasting algorithm for Bayesian phylogenetic inference

(usually abbreviated as MCMC) which simulate random variables from a target distribution, known up to a normalizing constant. The basic idea of the MCMC methods is first to construct a Markov chain that has the space of the parameters to be estimated as its state space and the posterior probability distribution of the parameters as its stationary distribution. Next, we simulate the chain and treat the realization as a representative sample from the posterior probability of the parameters of interests.

Two major strategies can be used to construct the Markov chains for exploring posterior distribution: the Metropolis-Hasting algorithm [8, 9] and the Gibbs sampler [10]. When applied to Bayesian phylogenetic inference, the Metropolis-Hasting algorithm can be described as Figure 1.

In the Metropolis-Hasting algorithm, $\alpha\left(\Psi^{(t)}, \Psi\right)$ is called the acceptance probability, and is defined as

$$\alpha(\Psi^{(t)}, \Psi) = \min\left(1, \frac{\pi\left(\Psi \mid D\right)}{\pi\left(\Psi^{(t)} \mid D\right)} \cdot \frac{q\left(\Psi^{(t)} \mid \Psi\right)}{q\left(\Psi \mid \Psi^{(t)}\right)}\right) \quad (4)$$

Here, $q(\Psi \mid \Psi^{(t)})$ is the proposal probability which can be in any form that satisfies

$$q(\cdot \mid \Psi) > 0. \quad (5)$$

In theory, a Markov chain constructed with the Metropolis-Hasting algorithm will converge to a stationary distribution if the chain is irreducible, aperiodic, and possesses a stationary distribution given the chain runs long enough [11]. Once we design a Markov chain that satisfies these requirements in Bayesian phylogenetic inference, we can approximate the posterior distribution of phylogenetic models.

Several practical implementation issues exist. One is to overcome the stickiness at local optima and maintain a desired acceptance ratio (the percentage of the proposed states that are accepted). We know that the irreducible property requires chains have a positive probability to move from one state Ψ_i into any other state Ψ_j in a finite number of time steps (s), i.e.

$$prob\left(\Psi\left(t+s\right) = \Psi_j \middle| \Psi\left(t\right) = \Psi_i\right). \quad (6)$$

Therefore, if we can design MCMC strategies that connect any two phylogenetic models in as few steps as possible while maintaining the quality of the proposals, we can improve the efficiency and accuracy of the posterior probability approximations.

3. The implementation of PBPI

As explained earlier, the computation time of a Bayesian phylogenetic inference is determined by two factors: the length of the Markov chains for approximating the posterior probability of the phylogenetic trees and the computation time needed for evaluating the likelihood values at each generation. The length of the Markov chains can be reduced by developing improved MCMC strategies to propose high quality candidate states and to make better acceptance/rejection decisions; the computation time per generations can be speed up by optimizing the likelihood evaluation and exploiting parallelism. PBPI implemented both techniques, but this paper will focus on the second factor.

3.1 Likelihood evaluation

The likelihood for a phylogenetic model Ψ is proportional to the probability of observing the dataset $X = \{x_{ij}\}$ under Ψ. Evaluating the likelihood values and estimating the maximum likelihood are the most computationally expensive parts of Bayesian phylogenetic inference. We now discuss how PBPI improves the performance of likelihood evaluation in Bayesian phylogenetic inference.

Given a phylogenetic model and an alignment of DNA sequences, the corresponding likelihood can be calculated using the algorithm proposed by Felsenstein [12] and described elsewhere [13]. This algorithm traverses the phylogenetic tree in post order and computes the conditional probabilities for each internal node from the conditional probabilities of its children nodes. We denote the length of the alignment M, the number of taxa N, and the number of possible states at each site S, each likelihood evaluation requires $o\left(N \cdot M \cdot S^2\right)$ multiplications.

A desirable property of the Felsenstein algorithm is the likelihood local update which is shown in Figure 2. Once the conditional probabilities for all nodes in the current phylogenetic tree (T) are preserved in evaluating its likelihood value, and the next phylogenetic tree T^* is proposed from current phylogenetic tree with partial changes, then only the conditional probabilities for those nodes appearing in the back tracing path to the root nodes needs to be recomputed. For example, as shown in Figure 1, once the likelihood of the tree on the left is evaluated, we only need to compute the conditional probabilities for node 10, 15, and 16 to obtain the likelihood of the tree on the right.

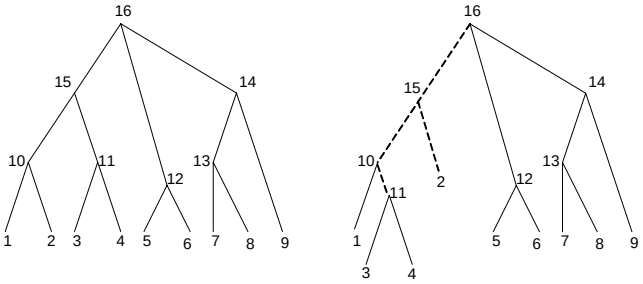

Figure 2: Illustration of the likelihood local update

Likelihood local update only requires $o\left(\log N \cdot M \cdot S^2\right)$ multiplications and introduces significant performance improvement at the cost of larger

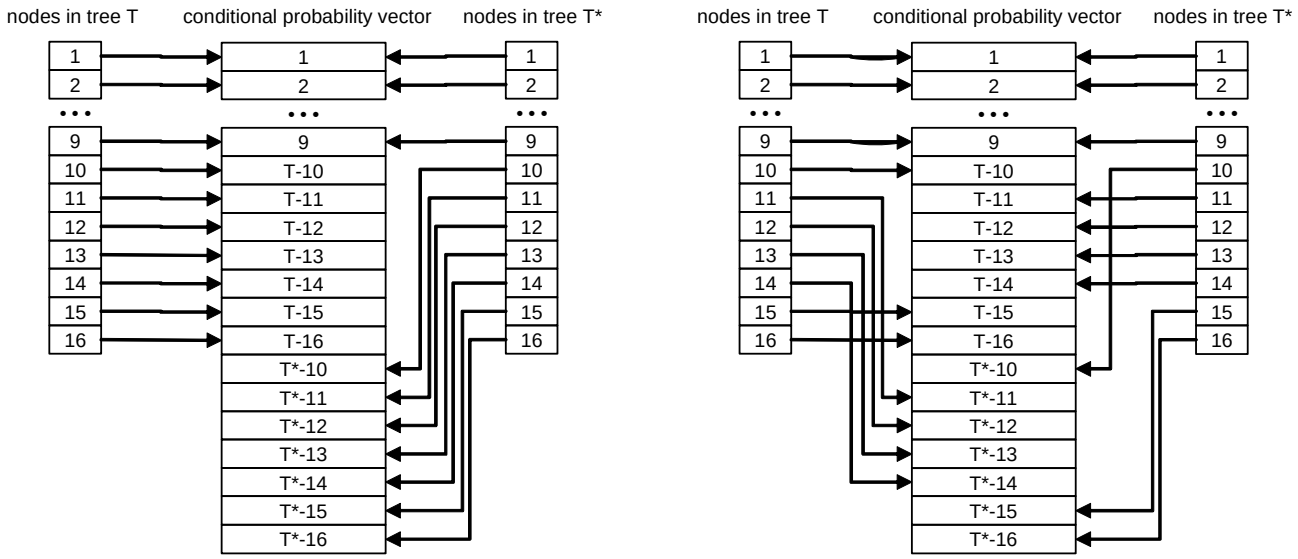

Figure 3: Illustration of vector re-linking

memory requirements. However, because in MCMC-based Bayesian phylogenetic inference the proposed tree candidate may be either accepted or rejected, we need an efficient way to keep the likelihood local update properties without frequent memory copies to save and restore the conditional probabilities for the affected nodes.

In PBPI, we use vector re-linking to implement likelihood local update in MCMC. Figure 3 shows how vector re-linking works:

(1) In the initialization step, we allocate a conditional probability vector for each internal node in the current tree and the proposal tree. Since the conditional probabilities of the leaf nodes stay constant, both trees share the same set of conditional probability vectors for their leaf nodes.

(2) When the new proposal tree is obtained from current tree and waiting to be evaluated, we swap the node-conditional probability vector links between the corresponding unaffected nodes in the current tree and proposal tree and keep the remaining links untouched.

(3) When the proposal is accepted, we swap the current tree and the proposal tree with all links unchanged; when the proposal is rejected, we swap back all links for unaffected nodes.

Additional optimizations for likelihood evaluations implemented by PBPI include sequence pattern compression under $i.i.d.$ (independent identically-distributed) assumption and tree balancing. If the number of unique patterns in the alignment is $\alpha \cdot M$, then pattern compression will reduce the required multiplications by $1-\alpha$. When likelihood local update is used, the number of multiplications is proportional to the depth of the tree. Tree balancing keeps the depth of the children nodes of the root roughly equal and results in a near smallest tree depth.

3.2 Parallel implementation

We identify the levels of parallelism in Bayesian phylogenetic inference.

(a) **Dataset level parallelism**: breaking a large dataset into a number of partially overlapping sub datasets, analyzing each sub dataset and merging the estimated results into a larger tree (supertree).

(b) **Run level parallelism**: performing several independent analyses on the same dataset to detect the convergence of the chains and remove odd results.

(c) **Partition level parallelism**: dividing the dataset into a number of partitions for improved evolutionary models or combined data from different genes.

(d) **Chain level parallelism**: running several chains to increase the mixing rate of the chains (for example Metropolis-coupled MCMC implemented in MrBayes).

(e) **Sequence level parallelism**: evaluating site (or pattern) likelihood across the sequence concurrently and summing up to obtain the global likelihood (the likelihood for the trees).

Chain level parallelism and sequence level parallelism are common to almost all Bayesian phylogenetic inference. Hence PBPI integrates chain and sequence level parallelism into its kernel module.

Details of the parallel algorithm implemented by PBPI can be found in our previous work [1] [14, 15]. We summarize this algorithm as follows. The processor pool is

[1] Section 3.1 and 3.2 detail the improvements to the original algorithm we proposed. We emphasize that the contribution of this work includes implementation of improved versions of our algorithm (MCMC strategies and sample algorithms), design and validation of a framework for Bayesian phylogenetic inference, direct comparisons to other approaches (e.g. MrBayes), and performance results for large data sets on a terascale system.

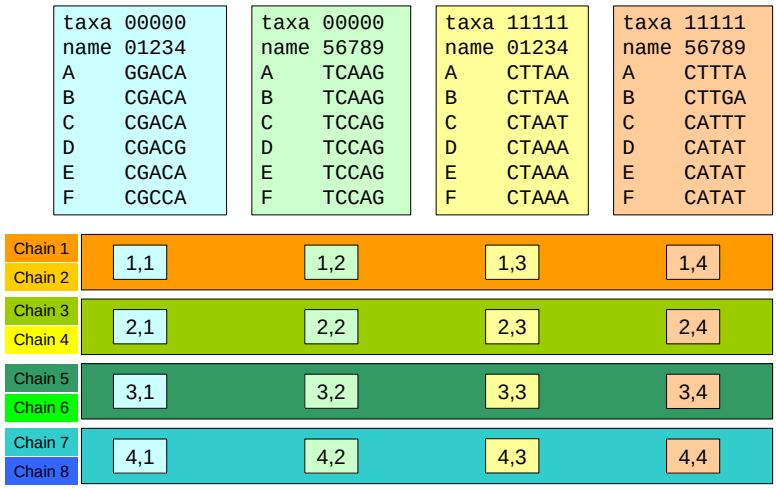

Figure 4: Illustration of processors organization and task assignment

arranged as a $c \times r$, two-dimensional Cartesian grid. The sequence is split into c segments and each column is assigned one segment. The chains are divided into r groups and each row is assigned one group of chains. Figure 4 illustrates how to map 8 chains onto a 4×4 grid where the number of characters (or patterns) is 20.

We use context-aware synchronizations to eliminate the need of frequent communications in synchronizing all processors. Context-aware synchronizations are based on the fact that if two nodes share some common knowledge, then both nodes know the next move of their peers.

We implement this common knowledge by assigning a row-wise random number generator (RNG-1) and a grid-wise random number generator (RNG-2) on each processor. RNG-1 has the same seed on all processors in each row and is used to generate new proposals and decide the acceptance of the proposals for all chains on that processor. RNG-2 has the same seed on all processors in the grid and is used to choose chains for swapping and to decide the acceptance of chain swapping.

During each chain update step, all processors in the same row always generate the same proposal. A collective communication is used to sum up the partial likelihood into global likelihood.

During each chain exchange step, all processors in the grid select the same pair of chains for swapping. Point-to-point communications between processors on the same column but different rows (only for selected chains) are needed to exchange the state information for the two selected chains.

According to the discussion in section 3.1, load imbalance among processors in different rows may occur due to different numbers of computations for the likelihood local update. Such imbalance can be reduced by increasing the intervals between two adjacent chain exchange steps.

The implementation of PBPI is based on MPI (message passing interface) and is portable to most platforms which support MPI.

4. Validation and verification

We validate the correctness of the PBPI implementation using a simulation study, a common technique to assess the performance of a phylogenetic method. We choose several phylogenetic trees published by RDP-II projects [16] and simulate 120 datasets under the JC69 and K2P models (transition/tranversion parameter = 2.0) using SEQ-GEN [17]. We ran PBPI in parallel to analyze these datasets under the JC69 model. The tree samples are summarized as 50% majority consensus tree, MPP tree (the tree with the highest frequency of occurrences in the tree samples) and 95% credible tree set (the set of trees whose cumulative occurrence frequency is larger than 95%).

Table 1 shows the topological accuracies represented by the average number of false negative branches in the estimations for the 24-taxa model tree (shown as Figure 5-a). The results show that PBPI can estimate the model tree with high accuracy either when the assumptions are correct (simulating data and estimating trees under JC69 model) or partially violated (simulating data under K2P model and estimating trees under JC69 model).

Furthermore, we examined the model tree and the estimated trees in detail. We found that nearly all the false negative branches shown in Table 1 are in fact caused by artifacts of the topological distance metrics. In other words, the topological distance metric does not consider zero-length branches or multi-furcated trees. For example, the MPP tree shown as Figure 5-b estimates the subtree in the model tree (fus.necph2:0.0088,(fus.necph3:0.0063,af044948:0.0000):0.0005) as ((fus.necph2:0.0090,fus.necph3:0.0061):0.0000,af044948:0.0000). These two subtrees are in fact topologically equivalent, if considering zero branch lengths.

For model trees with 50, 107 and 218 taxa, we obtained similar accuracy and results. Hence, the correctness of the PBPI implementation is confirmed.

Table 1: Topological accuracy for PBPI for simulated datasets
(MPP: Maximum Posterior Probability Tree; CTS: 95% credible tree set; CON: 95% majority consensus tree)

	JC69 Model			K2P Model		
# of characters	MPP	CTS	CON	MPP	CTS	CON
1000	1.6	0	1.8	2.6	0.6	2.6
5000	1.4	0.4	1.4	1	0.2	1.2
10000	0.4	0	0.6	0.8	0.2	0.8

Table 2: Sequential execution time for PBPI and MrBayes

Dataset	# of taxa	# of sites / patterns	PBPI	MrBayes	Speedup
fuso024_L1000	24	1000 / 402	1.7 min	10.1 min	5.8
fuso024_L10000	24	10000 / 1972	9.4 min	46.1 min	4.7
burk050_L1000	50	1000 / 432	2.8 min	23.4 min	8.0
burk050_L10000	50	10000 / 2429	15.1 min	2.0 hour	7.5
arch107_L1000	107	1000 / 1000	10.7 min	1.9 hour	10.0
arch107_L10000	107	10000 / 9996	1.8 hour	18.4 hour	9.8
back218_L1000	218	1000 / 1000	13.9 min	3.9 hour	15.8
back218_L10000	218	10000/10000	2.2 hour	43.4 hour	19.1

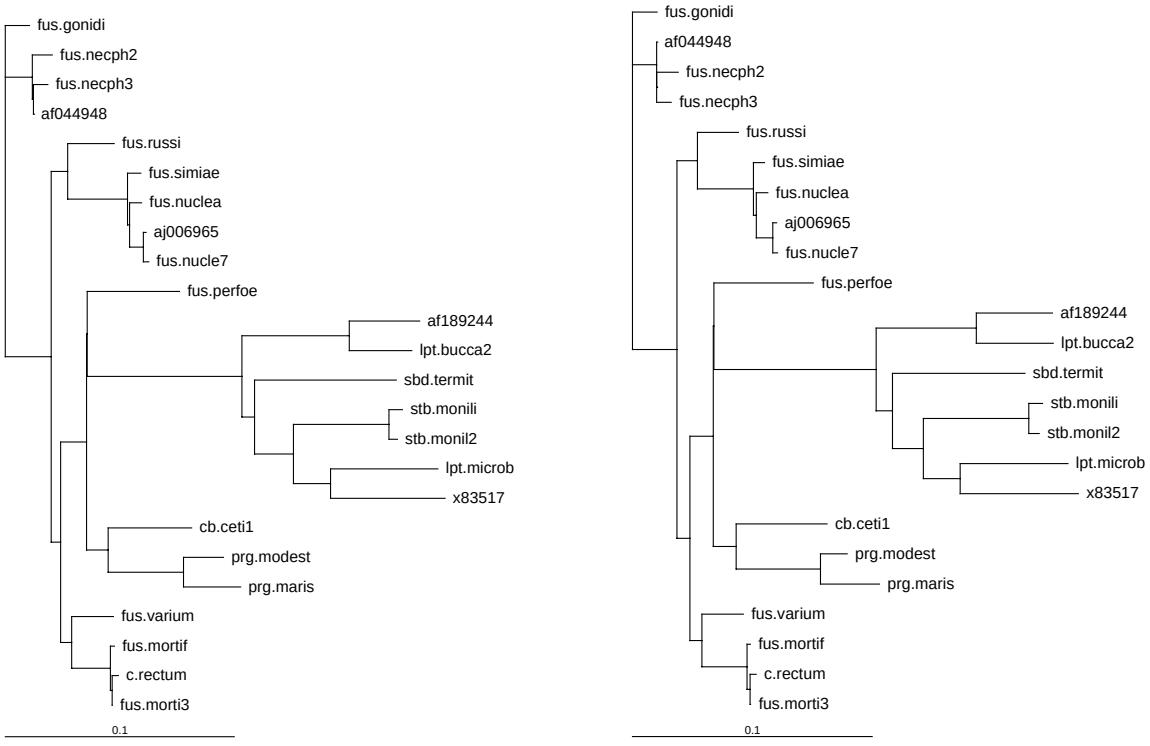

Figure 5: the 24-taxa model tree and an MPP trees for one of the simulated dataset

5. Performance evaluation

5.1 Experimental methodology

We evaluated the implementation of PBPI using System X in the Terascale Computing Facility at Virginia Tech. This system has 1100 Apple XServer G5 cluster nodes, each node having one Dual 2.3 GHz PowerPC 970FX processors, and 4 GB ECC DDR400 (PC3200) RAM. The cluster nodes were connected with SilverStorm's 91200 InfiniBand-based switches with a bidirectional port speed of 10 Gbps (billions of bits per second).

We chose a subset of from the datasets generated in validation and verification part as the benchmark datasets. For each model tree, we chose two datasets: one has 1000 characters, the other has 10,000 characters. We analyzed the benchmark datasets under the JC69 model. For each case tested, we run 4 MCMC chains with parallel tempering schema. We set the length of the chains as 200,000 generations, the sampling intervals as 100 generations, and the chain swapping intervals as 10 generations. We also ran MrBayes (version 3.1) with the same configurations for comparisons. For the last two datasets, the execution time values for MrBayes were estimated from MCMC runs with 5000 generations.

For each case, we executed 5 runs and used the average execution time over the 5 runs in performance evaluation. All timing values were wall clock time.

5.2 Sequential execution time

Table 2 provides the average execution times for PBPI and MrBayes on the benchmark datasets when running in a single processor on System X. The measured results showed that when both programs run in sequential mode, PBPI runs 5-19 times faster than MrBayes, depending on the problem size. For larger problems, PBPI yields larger performance improvement. The deviations for the measurements are less than 6%.

Both programs obtained statistically equivalent tree samples and the same majority consensus trees. However, we also noticed different posterior probability values for some branches in the summarized consensus trees. Further investigations are needed to explain the meaning of the posterior probability values of branches (or clade) for the phylogenetic trees and design some experimental methodology to verify that a Bayesian-based phylogenetic inference program estimates the posterior probability accurately. This is beyond the scope of our current work.

During the measurements, we used the same running parameters for both PBPI and MrBayes without using the improved MCMC strategies such as variable proposal step MCMC and multipoint MCMC implemented in PBPI. We can expect larger performance improvement for PBPI when these improvements are used since we can use a shorter chain by reducing the sampling intervals.

We credit this performance improvement of PBPI to its proposal algorithms, its likelihood local update and improved memory management.

5.3 Parallel performance and scalability

We chose benchmark datasets FUSO024_10000, ARCH107_1000, and BACK218_10000 as fixed problems to investigate the parallel speedup of PBPI. These three datasets represent three kinds of problems:

- FUSO024_10000: problems with a small number of taxa and long sequences of characters;
- ARCH107_1000: problems with a medium number of taxa and short sequences of characters;
- BACK218_10000: problems with a large number of taxa and long sequences of characters.

For each dataset, we ran 4 chains with parallel tempering schema, where each chain lasts 200,000 generations. We scale the number of processors from 4 to 64 processors and calculate the speedup using the mean values of the execution time over 5 runs. The results are shown in Figures 6.

The results indicate that for all three benchmark datasets, PBPI achieves roughly linear speedup. For FUSO024_L10000 dataset, the average speedup on 64

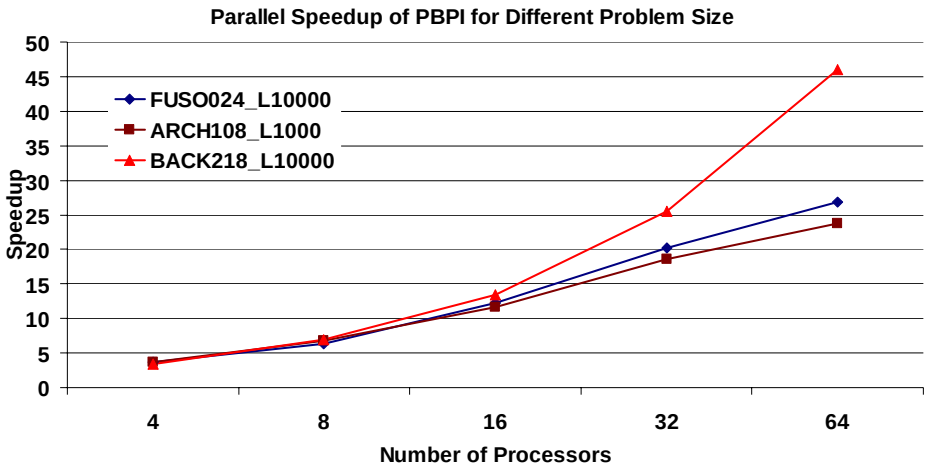

Figure 6: Parallel speedup of PBPI for three benchmark datasets

processors is 26.8; for ARCH107_L1000 dataset, the average speedup was 23.7; for Back218_L10000 dataset, the average speedup was 46.1.

We also measured the performance of PBPI on more than 64 processors for Back218_L10000 dataset and provided the results in Figure 7. For the first 64 processors, we used one processor per node in the computation; for 128 and 256 processors, we used two processors per node. Since each chain swapping step needs one point-to-point communication between two corresponding processors on two different rows, there are message contentions between two processors on the same node. This is the reason for the obvious slow down for the speedup when PBPI ran on 128 processors. The average speedup was 52.3 on 128 processors, or 74.6 on 256 processors.

Combining the performance improvement for both sequential optimizations and parallel speedup, for the same dataset, PBPI ran up to 874 times faster using 64 processors than MrBayes using a single processor on a system similar to SystemX, or 1424 times faster using 256 processors.

5.4 Discussion

Figures 6 and 7 show the parallel speedup of PBPI for a fixed-size problem (Amdahl's law [18]). From both figures, we observe near linear speedup for small problems up to 16 processors and for large problems on 64 processors. We credit this performance improvement to our sequence level parallelism and due to the context-aware synchronizations that ensure processors operate autonomously and avoid message communications.

We noticed recently that the MrBayes developers implemented a parallel version of their framework that uses chain level parallelism. Since we haven't observed any significant performance changes when we ran the parallel version of MrBayes using 1, 2, 4, and 8 processors on System X, this study does not include a direct comparison. However, since the current implementation of parallel MrBayes only exploits chain level parallelism, its scalability of parallel MrBayes will be limited by the number of chains used in the analysis. Based on this assumption and primarily due to the other forms of parallelism we have identified and our current results, we estimate PBPI will run up to 219 times faster (on 64 processors, or 375 times on 256 processors) than the parallel version of MrBayes for the tested dataset.

Following the scalability analysis provided in [14] and according to Gustafson's Law for fixed-time speedup [19], we can scale the problem size to achieve even large speedups for PBPI. Here, the problem can be scaled in three dimensions: 1) the number of taxa; 2) the number of characters (patterns); and 3) the number of MCMC chains. The trend of parallel speedup of PBPI for scaled problem is shown in Figure 6.

6. Related work

We classify related work into two categories: Bayesian phylogenetic inference and parallel phylogenetic inference.

MCMC-based Bayesian analysis was introduced into phylogenetic inference by three independent research groups (Yang et al at UC Berkeley[20], Mau et al at University of Wisconsin [21], and Li et al at Ohio State [22]) in 1996 and became widely used after the development of BABME [23] and MrBayes [7]. Reviews of Bayesian phylogenetic inference and its differences with "traditional" approaches are available by Huelsenbeck el al [24] and Holder et al [25]. MrBayes also implemented the metropolis-coupled MCMC method (also called MCMCMC or MC^3) to improve the mixing rate of the sampling chains. Since most of these implementations are sequential, they have inherent performance limitations when solving large phylogenetic problems.

The first parallel implementation of Bayesian phylogenetic inference was described by Feng et al [14] and served as the starting point of PBPI described in this work. As described in section 3, we made several important changes for PBPI which brought dramatic performance improvement. MrBayes also provides a parallel implementation (see Altekar et al [26]). However, based on

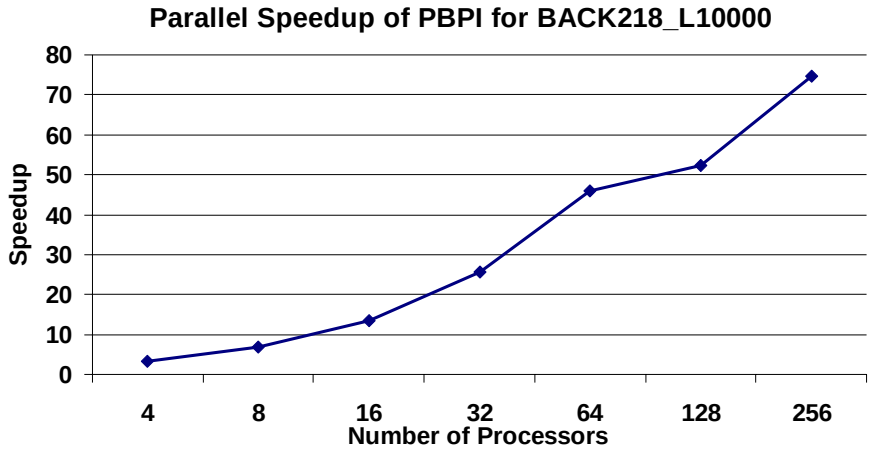

Figure 7: Parallel speedup of PBPI for benchmark datasets BACK218_L10000

the literature description, parallel MrBayes only provides chain level parallelism and its scalability is limited by the number of chains used in the analysis. This implementation of MrBayes increases the number of chains to effectively scale the work load while fixing the problem size. PBPI overcomes this limitation by combining sequence level parallelism and chain level parallelism. Another key difference between PBPI and MrBayes is that PBPI implements several improved MCMC strategies which can speedup the convergence rate and avoid local optima.

Several scalable parallel programs for maximum likelihood-based phylogenetic methods, have been developed recently, among them are parallel fastDNAml [27, 28], PAxML [29], parallel genetic algorithm for ML [30], and parallel TREE-PUZZLE [31]. Bader et al discussed GRAPPA, a highly scalable implementation for breakpoint phylogenetic analysis using gene order data [32]. Since Bayesian phylogenetic inference uses different criterion to define the "optimal" trees for given datasets, PBPI is quite different from these programs both in its underlying inference method and parallel implementations. We believe the application of high performance computing in phylogenetic inference will bring significant changes in the field of phylogenetics both in the scale of inferred phylogenies and the in the reduction of time to solution.

7. Conclusions and future work

There are many advantages of Bayesian phylogenetic inference [25]. However, expensive computations have limited the application of Bayesian phylogenetic inference to extremely large problems. The dramatic reduction in time to solution achieved by PBPI enables new opportunities for Bayesian phylogenetic inference. For example, using the popular MrBayes framework, the analysis of Back218_L10000 may take 18 days using 4 MCMC chains with the length of 1,000,000 generations; using PBPI, the execution time to generate the equivalent samples takes 30 minutes.

Currently, PBPI implements the GTR (general time reversible) model and its simplifications for DNA sequences. As a next step, we will implement more realistic models and support additional data types. More realistic models will provide further insight into evolution with higher accuracy but will also consume more computation time.

One important aspect for Bayesian phylogenetic inference is that it provides an assessment of the confidence level for the estimated trees using the quantity of posterior probability. In our work, we identified the risk of inaccurate posterior probability estimations for large complicated datasets using MCMC methods and developed numerous improvements. We will further investigate the effectiveness of these improvements and their impact on performance.

The Bayesian approach has the potential to resolve issues in building very large trees. But current Bayesian phylogenetic inference can not applied to more than 1,000 taxa due to the resulting computational complexity. We plan to incorporate divide and conquer techniques such as the supertree method [33] or DCM method [34] into future versions of PBPI to solve phylogenetic problems with tens of thousands of taxa.

References

[1] NSF, "Assembling the Tree of Life (ATOL)," 2003.

[2] Pickett, K. M., Simmons, M. P., and Randle, C. P., "Do Bayesian support values reflect probability of the truth?," *Cladistics-the International Journal of the Willi Hennig Society*, vol. 20, pp. 92-93, 2004.

[3] Huelsenbeck, J. P., Larget, B., Miller, R. E., et al., "Potential applications and pitfalls of Bayesian inference of phylogeny," *Syst. Biol.*, vol. 51, pp. 673-688, 2002.

[4] Murphy, W. J., Eizirik, E., O'brien, S. J., et al., "Resolution of the Early Placental Mammal Radiation Using Bayesian Phylogenetics," *Science*, vol. 294, pp. 2348-2351, 2001.

[5] Glenner, H., Hansen, A. J., Sorensen, M. V., et al., "Bayesian inference of the metazoan phylogeny: A combined molecular and morphological approach," *Current Biology*, vol. 14, pp. 1644-1649, 2004.

[6] Williams, T. L. and Moret, B. M. E., "An Investigation of Phylogenetic Likelihood Methods," in *Proceedings of. Proceeding of 3rd IEEE Symposium on Bioinformatics and Bioengineering*, 2003.

[7] Huelsenbeck, J. P. and Ronquist, F., "MRBAYES: Bayesian inference of phylogenetic trees," *Bioinformatics*, vol. 17, pp. 754-755, 2001.

[8] Metropolis, N., Rosenbluth, A. N., Rosenbluth, M. N., et al., "Equations of state calculations by fast computing machine," *J. Chem. Phys.*, vol. 21, pp. 1087-1091, 1953.

[9] Hastings, W. K., "Monte Carlo sampling methods using Markov chains and their application," *Biometrika*, vol. 57, pp. 97-109, 1970.

[10] Geman, S. and Geman, D., "Stochastic Relaxation, Gibbs Distributions, and the Bayesian Restoration of Images," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 6, pp. 721-741, 1984.

[11] Tierney, L., "Markov-Chains for Exploring Posterior Distributions," *Annals of Statistics*, vol. 22, pp. 1701-1728, 1994.

[12] Felsenstein, J., "Evolutionary trees from DNA sequences: a maximum likelihood approach," *Journal of Molecular Evolution*, vol. 17, pp. 368-76, 1981.

[13] Durbin, R., Eddy, S., Krogh, A., et al., *Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids*: Cambridge University Press, 1998.

[14] Feng, X., Buell, D. A., Rose, J. R., et al., "Parallel algorithms for Bayesian phylogenetic inference," *Journal of Parallel and Distributed Computing*, vol. 63, pp. 707-718, 2003.

[15] Feng, X., "High Performance, Bayesian-based Phylogenetic Inference Framework," PhD

Dissertation, Department of Computer Science and Engineering, University of South Carolina, 2006.
[16] Cole, J. R., Chai, B., Farris, R. J., et al., "The Ribosomal Database Project (RDP-II): sequences and tools for high-throughput rRNA analysis," *Nucl. Acids Res.*, vol. 33, pp. D294-296, 2005.
[17] Rambaut, A. and Grassly, N. C., "Seq-Gen: An application for the Monte Carlo simulation of DNA sequence evolution along phylogenetic frees," *Computer Applications in the Biosciences*, vol. 13, pp. 235-238, 1997.
[18] Amdahl, G. M., "Validity of the Single Processor Approach to Achieving Large-Scale Computing Capabilities," in *Proceedings of. AFIPS Spring Joint Computer Conference*, Reston, VA, 1967.
[19] Gustafson, J., "Reevaluating Amdahl's Law," *Communications of the Acm*, vol. 31, pp. 532-533, 1988.
[20] Yang, Z. H. and Rannala, B., "Bayesian phylogenetic inference using DNA sequences: A Markov Chain Monte Carlo method," *Molecular Biology and Evolution*, vol. 14, pp. 717-724, 1997.
[21] Mau, B., Newton, M. A., and Larget, B., "Bayesian phylogenetic inference via Markov chain Monte Carlo methods," *Biometrics*, vol. 55, pp. 1-12, 1999.
[22] Li, S. Y., Pearl, D. K., and Doss, H., "Phylogenetic tree construction using Markov chain Monte Carlo," *Journal of the American Statistical Association*, vol. 95, pp. 493-508, 2000.
[23] Simon, D. L. and Larget, B., "Bayesian analysis in molecular biology and evolution (BAMBE)," Department of Mathematics and Computer Science, Dequesne University, Pittsburgh 1998.
[24] Huelsenbeck, J., Ronquist, F., Nielsen, R., et al., "Bayesian inference of phylogeny and its impact on evolutionary biology," *Science*, vol. 294, pp. 2310-2314, 2001.
[25] Holder, M. and Lewis, P. O., "Phylogeny estimation: traditional and Bayesian approaches," *Nature Rev. Genet.*, vol. 4, pp. 275-284, 2003.
[26] Altekar, G., Dwarkadas, S., Huelsenbeck, J. P., et al., "Parallel metropolis coupled Markov chain Monte Carlo for Bayesian phylogenetic inference," *Bioinformatics*, vol. 20, pp. 407-415, 2004.
[27] Stewart, C. A., D. Hart, D. K. Berry, G. J. Olsen, E. Wernert, W. Fischer, "Parallel implementation and performance of fastDNAml - a program for maximum likelihood phylogenetic inference," in *Proceedings of. Supercomputing 2001*, 2001.
[28] Olsen, G., Matsuda, H., Hagstrom, R., et al., "fastDNAmL: a tool for construction of phylogenetic trees of DNA sequences using maximum likelihood," *Comput. Appl. Biosci.*, vol. 10, pp. 41-48, 1994.
[29] Stamatakis, A., Ludwig, T., and Meier, H., "RAxML-III: a fast program for maximum likelihood-based inference of large phylogenetic trees," *Bioinformatics*, vol. 21, pp. 456-463, 2005.
[30] Brauer, M. J., Holder, M. T., Dries, L. A., et al., "Genetic algorithms and parallel processing in maximum-likelihood phylogeny inference," *Molecular Biology and Evolution*, vol. 19, pp. 1717-1726, 2002.
[31] Schmidt, H. A., Strimmer, K., Vingron, M., et al., "TREE-PUZZLE: maximum likelihood phylogenetic analysis using quartets and parallel computing," *Bioinformatics*, vol. 18, pp. 502-504, 2002.
[32] Moret, B. M. E., Bader, D. A., and Warnow, T., "High-performance algorithm engineering for computational phylogenetics," *Journal of Supercomputing*, vol. 22, pp. 99-110, 2002.
[33] Sanderson, M. J., Purvis, A., and Henze, C., "Phylogenetic supertrees: assembling the trees of life," *Trends in Ecology & Evolution*, vol. 13, pp. 105-109, 1998.
[34] Huson, D. H., Nettles, S. M., and Warnow, T. J., "Disk-covering, a fast-converging method for phylogenetic tree reconstruction," *Journal of Computational Biology*, vol. 6, pp. 369-386, 1999.