A PARALLEL APPROACH TO MULTIPLE SEQUENCES ALIGNMENT AND PHYLOGENETIC TREE LABELING

by

Jingjing Wang

Electrical Engineering
B.E., Southeast University, 2008

A Thesis
Submitted in Partial Fulfillment of the Requirements for the
Master of Science Degree

Department of Computer Science
in the Graduate School
Southern Illinois University Carbondale
December 2010

THESIS APPROVAL

A PARALLEL APPROACH TO MULTIPLE SEQUENCES ALIGNMENT AND PHYLOGENETIC TREE LABELING

by

Jingjing Wang

A Thesis Submitted in Partial

Fulfillment of the Requirements

for the Degree of

Master of Science

in the field of Computer Science

Graduate School
Southern Illinois University Carbondale
November 12, 2010

AN ABSTRACT OF THE THESIS OF

Jingjing Wang, for the Master of Science degree in Computer Science, presented on 7 September 2010,at Southern Illinois University Carbondale.

TITLE: A PARALLEL APPROACH TO MULTIPLE SEQUENCES ALIGNMENT AND PHYLOGENETIC TREE LABELING

MAJOR PROFESSOR: Dr. Mengxia Zhu

An evolutionary tree represents the relationship among a group of species, DNA or protein sequences, and play fundamental roles in biological lineage research. A high quality tree construction relies heavily on optimal multiple sequence alignment (MSA), which aligns three or more sequence simultaneously to derive the similarity. On the other hand, a good tree can also be used to guide the MSA process. Due to the high computational cost to conduct both the MSA and tree construction, parallel approaches are exploited to utilize the enormous amount of computing power and memory housed in a supercomputer or Linux cluster. In this paper, first of all, a divide and conquer based parallel algorithm is designed and implemented to perform optimal three sequence alignment using reduced memory cost. Secondly, all internal nodes of a phylogenetic tree resulting from a parallel Maximum-likelihood inference software are labeled using the parallel MSA. Such tree node labeling process is carried out from top down and is also parallelized to fully utilize the numerous cores and nodes in a high performance computing facility.

ACKNOWLEDGMENTS

I would like to thank Dr. Zhu for her constant help, guidance and suggestions throughout this thesis.

I would also like to thank Dr. Hou and Dr. Che for consenting to be a part of my thesis committee and for their guidance.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Phylogenetics refers to the evolutionary lineage among different organisms. Some evidence shows that some organisms share similar features, while evolving from the same ancestor and other features are distinct to help organisms to adapt to their specific environment conditions. Therefore, by studying the phylogeny among different organisms, it is conducive to find the similarities and differences among different species [1, 2].

A phylogenetic tree, also called evolutionary tree represents an evolutionary relationships or phylogeny among species, and play fundamental roles in biological research. The leaf nodes are represented as species and internal nodes are shown as the common ancestor of its descendant nodes. The length of a branch connecting one node to another indicates evolution time.

In the phylogenetic tree study, there are two main computational challenges. The first major challenge is to identify the optimal tree topology at the minimum tree cost. The enumeration of all of the possible tree topologies will not be feasible for big input sequences. With a given input sequence size of n, the total number of unrooted tree is $(2n-5)!!$ .It has been proved that this problem belongs to the category of NP-hard [3].Consequently, researchers have looked for alternatives such as approximation or heuristic approaches with polynomial time complexity, which have yielded suboptimal solutions. Phylogeny construction methods can be roughly categorized as either distance-based or character-based. Distance-based method computes the distance for every aligned pair of sequences to guide the tree construction. The character-based method uses multiple sequence alignment directly and is based on criteria of maximum parsimony or maximum likelihood inferences [12]. The second major computation challenge is to perform efficient sequence alignment within identified tree topology for internal node labeling at minimum tree cost.

In my thesis, both of these computational challenges are considered and addressed in our parallel program called Parallel Multiple Sequence Alignment and Phylogenetic Tree reconstruction (PMSAPT). The program exploits the parallel programming technique to shorten the execution time of tree construction as well as the multiple sequence alignment. In this proposed algorithm, a tree will be constructed with all leaf nodes from the given sequences and internal nodes labeled. The criterion is to minimize the summation of all tree branch lengths in the minimum time span. This parallel algorithm is designed to minimize the communication overhead between nodes and at the same time maximize the concurrency of independent tasks.

The thesis is organized as follows: Chapter 2 discusses the related works. Chapter 3 presents our algorithms on tree topology construction and sequence alignment. Chapter 4 shows the benchmark experiment results compared with those of existing software to illustrate the novelty and efficiency of our approaches. Conclusion is given in Chapter 5.

CHAPTER 2

RELATED WORK

Two main approaches belong to distance-based class as Neighbor Joining [7] and UPGMA (Unweighted Pair Group Method with Arithmetic Mean) [8]. UPGMA is a hierarchical clustering method to find the closest related pair of clusters from a distance matrix and combine this pair as one cluster used for next iteration. Different with UPGMA, Neighbor Joining algorithm creates an internal node rather than a cluster in each iteration. Both UPGMA and Neighbor-Joining methods are greedy and fast, however, such growing tree topology does not ever change its structure once a new node is added. Therefore, it is difficult to achieve a good tree topology. By recognizing such inflexibilities in the issue, some tree search algorithms are applied as tree refinement mechanism. The branch swapping methods such as nearest-neighbor interchanges (NNI), subtree prunning and regrafting (SPR), and tree bisection and reconnection (TBR) are widely used[9, 10]. All of these algorithms work by swapping the branches from the original position to a different position to create a new tree topology. Although these tree refinement strategies indeed can achieve better results, it is known that these programs can be stuck in local optima in the search for global optima [11].

The character-based method considers each column of the sequences independently. Maximum parsimony method produces the tree with the minimum number of changes to evolve into the current sequences. Maximum Likelihood method aims to identify the tree topology with the maximum likelihood. Such a tree is most likely to obtain the current sequences. The advantage of maximum likelihood based methods is that a large number of tree space is searched while all sequence information is utilized [12]. Therefore, the final tree topology is more reliable and accurate than that acquired from both distance-based and maximum parsimony based methods. However, some proof shows that it is a NP-hard problem to find a tree with maximum likelihood value due to a large tree space [13, 14]. In recent years, some heuristic maximum likelihood based methods integrated with other optimization strategies have been developed. IQPNNI [15] method (a method in terms of the Important Quartet Conception with the nearest neighbor interchange) has been employed to search a tree through large tree space under maximum likelihood criterion. For each iteration, some leaf nodes are deleted and re-inserted by following IQP (The Important Quartet Concept). A new reconstructed tree is further optimized using nearest-neighbor interchanges (NNI) and compared to the previous one. The tree which was computed to have a higher log-likelihood is kept. PHYML [16] (Heuristic Maximum Likelihood phylogenetic tree from multiple alignment) uses BIONJ [17] method (an improved version of the NJ algorithm) to construct an initial tree and NNI method for tree search. RAxML [12, 18] is another program for the inference of phylogeny under maximum likelihood criterion. It has some advantages over other software .First, in contrast with IQPNNI and PHYLIP which both make use of NNI method for tree search, RAxML uses a different tree search scheme named Lazy Subtree Rearrangement (LSR) which exploits a larger portion of candidate trees and thus is more likely to find a global optima. In addition, RAxML software requires less memory space and computation time than some other programs [19, 20, 21, 22]. However, RAxML only produces a tree topology and the log likelihood value without labeling each internal node and alignment.

Some tree construction and sequence alignment software has been developed such as TAAR (Tree Alignment and Reconstruction Application software) [4] and GESTALT (Genomic Steiner Alignment) [5], which generates sequence alignment via a tree. Yue and Tang developed a tree alignment software called MSAM to iteratively label each internal node derived from its closest three neighbors and by using TBR to optimize the current tree cost [6]. All above-mentioned software obtains an improved final tree structure starting from an initial guiding tree created from either the minimum spanning [5] or neighbor-joining methods [7].

To label each internal node of this tree for minimum total tree cost, we parallelize an optimal three sequence alignment program based on dynamic programming with the source code extracted from MSAM package[6]. In addition, a divide and conquer strategy was used to reduce the memory cost from O(n^3) to O(n^2) [23]. The limitation of original MSAM program is that it is easy to get stuck in the local optima in that N-J tree with TBR method applied for tree inference. To address this issue, the tree topology from RAxML is used to label each node from that tree. The labeling process continues iteratively until convergence point is hit. Also, a parallel algorithm for optimization is developed to speed up the program.

CHAPTER 3

PHYLOGENETIC TREE CONSTRUCTION AND MSA

We use parallel RAxML which is a better Maximum Likelihood-based Inference program to obtain a phylogenetic tree topology. Then we parallelize an optimal multiple sequence alignment (MSA) method to label each internal node from its nearest three neighbors. Such labeling process can be carried out simultaneously for internal nodes at the same level. So the labeling process is mainly determined by the height of the tree.

3.1 Maximum Likelihood-based Inference

A tree topology with highest likelihood value is selected under Maximum likelihood. However, it is a NP hard problem to find such a tree topology because of the large tree space we have to search [3]. Some heuristic tree topology inference methods were developed to generate some potentially good tree structures and find the one with the highest likelihood value.

3.1.1 Tree Likelihood Calculation

The maximum likelihood-based method in phylogenetics tree construction employs an evolutionary model which provides the probability that one sequence evolve into another along a branch. Each site of the sequences is assumed to evolve independently. The likelihood of each site is computed following the path from root node to each leaf node. If the internal node sequence is unknown, all possible nucleotides need to be considered by summing the probability of all possible states. The final tree likelihood is scored by multiplying the likelihood of each site.

An example tree structure is given in Fig. 1, and how to compute its likelihood value under a simple DNA evolution model is explained:

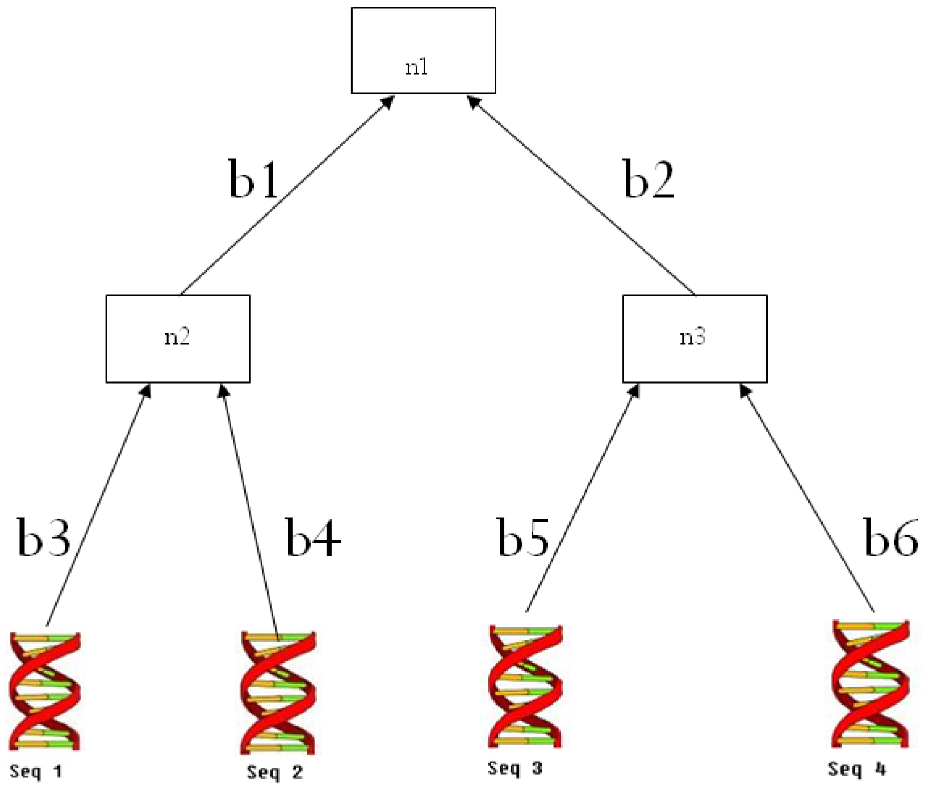

Figure 1. An example tree

In the above example, b1, b2, b3, b4, b5, b6 represents the branch length of the tree. For any position (site) x, the likelihood of this site could be computed by the formula in Eq.1,for example, P(sq1|n2,b3) gives the probability that a base in n2 evolves into seq1 along with a branch b3 [12]:

L(x)

=

$$\sum_{n1=A}^{T}\sum_{n2=A}^{T}\sum_{n3=A}^{T} P(n1)P(n2 \mid n1, b1)P(n3 \mid n1, b2)P(sq1 \mid n2, b3)P(sq2 \mid n2, b4)P(sq3 \mid n3, b5)P(sq4 \mid n3, b6)$$

Equation 1. Likelihood calculation of a specific site x [12]

After the likelihood of each site is computed, then likelihood of this tree could be computed as the product of the likelihood of each site:

$$L_{tree} = \prod_{i} L(i)$$

Equation 2. Tree likelihood formula [12]

Each potential tree topology is generated, with its likelihood computed. The tree with the highest likelihood value will be chosen.

3.1.2 RAxML software

RAxML [12] (Randomized Axelerated Maximum Likelihood), developed by A.Stamatakis, is a maximum likelihood-based tree construction software. In sequential RAxML, an initial parsimony tree which has a better likelihood than N-J tree is built as a starting tree. After that, a computing intensive tree optimization process starts by performing a standard subtree rearrangement. In detail, all possible subtrees within the current best tree are subsequently moved from their original positions to new ones by the lower up to the upper rearrangement distance setting. Instead of optimizing the branch length of the entire tree to calculate its likelihood for each insertion, which definitely increases the computation cost, only the three branches neighboring to insertion point is optimized before the likelihood is calculated during each rearrangement step. 20 best tree candidates are stored after each rearrangement step, and only these 20 trees rather than all possible ones are performed global branch length optimization based on a pulley principle to improve likelihood value. The tree with the highest likelihood will be selected for the next rearrangement step. Also during each rearrangement step, the current best tree is updated if the tree likelihood is improved after a movement of a specific subtree, and the remaining subtree rearrangements are performed on this topology. The program ends until the upper rearrangement distance setting reaches the given maximum one. To pursue a better performance, a parallel RAxML was also developed. In phase one, the master sends the input sequence file to each slave. Each worker generates random permutation and builds trees with parsimony values. The tree with the highest likelihood value stands out and is sent back to the master. In phase two (a), the master distributes a specific subtree ID and current best tree topology to each slave. Each slave rearranges and integrates the specified subtree and returns a tree with a better likelihood value. The master compares this tree with the current best tree and updates the tree with higher likelihood as the best one, followed by sending a new subtree ID with current best tree topology to the slave. After all subtrees are rearranged, the program goes to phase two (b). The master requests a tree list from each worker, which stores 20 best trees computed by each worker during this rearrangement step .After that, the master merges lists and distributes 20 best trees to workers for global branch length optimization. The tree with the highest likelihood value is selected and compared with the old one. The master node starts the next rearrangement step if the likelihood is improved. Otherwise, the rearrangement setting increases by one if the maximum rearrangement setting is not reached and phase two as mentioned above is performed again. The termination condition in parallel program is the same with that in sequential one [12].

3.2 Multiple Sequence Alignment along Phylogenetic Tree

RAxML software produces a good phylogenetic tree topology. The next step is to label internal nodes in this tree topology. At first, an initial sequence should be assigned to each internal node. This procedure is called tree initialization .The tree refinement follows up by computing a new sequence for each internal node by conducting MSA with its three nearest neighbors. This original idea comes from F.Yue and J.Tang in their previous work [6]. The novelty of their algorithm is memory usage reduction from O (n^3) to O (n^2), however, the cost time is still high. Therefore, we proposed parallel approaches to address both the MSA and tree node labeling. The details are given as below.

3.2.1 Internal Nodes Initialization

There are various methods to initialize each internal node. One method is to randomly assign a sequence to each internal node. Another method proposed by F.Yue and J.Tang is to generate a median sequence for each internal node with its three nearest leaf nodes [6]. Although a better result is produced, the computation is slow for long sequences .Since it is not very necessary to spend so much time on the initialization of internal nodes, we use a fast pairwise alignment method to initialize each node with its two child nodes. At first, we align sequences to determine the gap position if the lengths of two sequences are different. To generate the median sequence for the internal node, a consensus vote is operated based on a mutation matrix at every site of the alignment.

3.2.2 Serial Tree Optimization

After the initialization step, each internal node is assigned a sequence as the starting point for later adjustment. However, this assignment still has much space to be improved, and thus the multiple sequence alignment of input sequences will be still far away from the optimal one. The tree optimization mechanism implemented in MSAM software is adapted and enhanced here [6, 23].The optimization steps are the following: Starting from the root node to next-to-the last level of the tree generated from RAxML , a new sequence , called median, is computed with alignment from its three neighbors to label node sequences. This new sequence is allowed to replace the old one if the sum of branch distances to three neighbors is smaller than previous labeling. Once there is a replacement during an iteration, the next iteration is about to execute. This optimization procedure terminates if there is no improvement during the previous iteration. To generate each median sequence, a divide-and-conquer strategy from MSAM [6, 23] is adopted to find a midpoint to split a 3-D sequence cube into two smaller cubes recursively, as showed in Fig. 2. The central part of this approach determines a midpoint where the optimal alignment passes through to split 3D cube. Similar with a divide and conquer approach for two sequences [24],

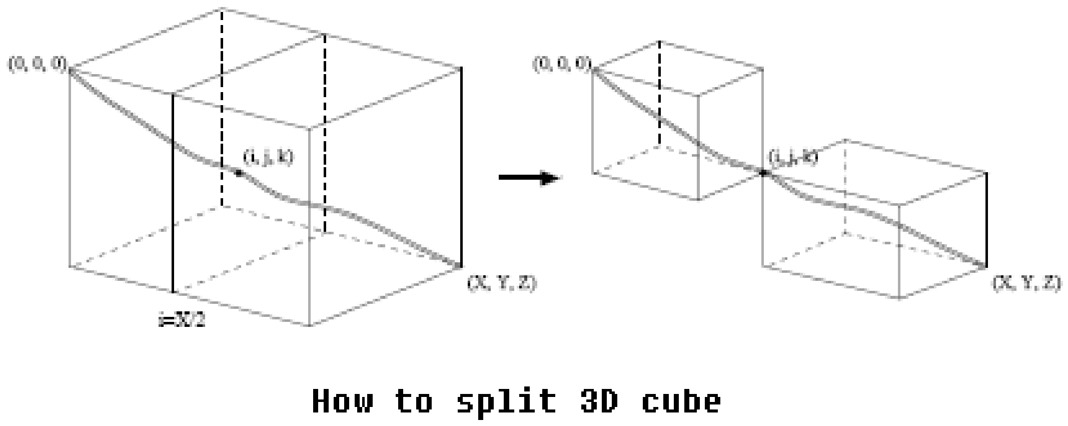

Figure 2. How to split 3D cube [6]

one input sequence with the length X is selected at first, and i=X/2 is defined as the first coordinate of midpoint, so the problem is to find another two coordinates of such midpoint. To find these two optimal coordinates (j and k), a score function is implemented to record both the forward and backward scores of three sequences by respectively using the parameter (C, B, A, Z, Y, i) and (rev(C), rev(B),rev(A)[i…X],Z,Y,X-i), where rev() stands for a reverse string. These two coordinates are determined when the sum of forward and backward scores is minimized [23, 25].This score function calculates the best score of alignment ending in each state format. Since each coordinate has three possible states: I(insert), D(delete), and M(match/mismatch), there are 27 possible state combinations that are required to calculate the score . The minimum cost among these is picked, which is represented as a formula showed in Eq.3. To make the correctness of the global alignment, the final state information obtained from forward scoring needs to be sent to the next recursive call in upper cube as the end. The state information generated from backward scoring will be passed through to the next recursive call in bottom cube as the start. This divide and conquer method is recursively executed until the 3-D cube can't be divided any more (i=1). Then a conquer procedure starts from the smallest cube by aligning three sequences kept in this cube and generating a median one in terms of the selected state combination, which yields the smallest cost from Eq.3.

$$T_{\delta_1\delta_2\delta_3}(i,j,k)=\min\begin{cases}T_{MMM}(i',j',k')+Cost(A[i],B[j],C[k],\delta_1\delta_2\delta_3)-g\times Transition(\delta_1\delta_2\delta_3,MMM)\\ T_{MMD}(i',j',k')+Cost(A[i],B[j],C[k],\delta_1\delta_2\delta_3)-g\times Transition(\delta_1\delta_2\delta_3,MMD)\\ T_{MDM}(i',j',k')+Cost(A[i],B[j],C[k],\delta_1\delta_2\delta_3)-g\times Transition(\delta_1\delta_2\delta_3,MDM)\\ \cdots\cdots\cdots\cdots\\ T_{DDM}(i',j',k')+Cost(A[i],B[j],C[k],\delta_1\delta_2\delta_3)-g\times Transition(\delta_1\delta_2\delta_3,DDM)\end{cases}$$

Equation 3. The minimum cost [23]

When searching the midpoint, only the cells at level i-1 determine ones at level i. Instead of keeping all of the state information at each level, which requires memory space O (n^3), the algorithm developed by F.Yue and J.Tang can reduce memory usage from O(n^3) to O(n^2) in that only the score is delivered to find a midpoint . Started from level i=0 to level i= X, two arrays U[Y,Z] and D[Y,Z] are enough to deliver as well as store the score for each different pair of y and z. After the forward and backward calculation ends, the minimum of the combing score can be found and the related coordinates will be known to represent the midpoint.

3.2.3 Parallel Tree Optimization

MSAM software, which uses the above mentioned algorithm to compute each internal node sequentially, is unrealistic to handle large number of sequences due to its high computational cost. Our parallel approaches can be used to minimize the execution time by utilizing a message-passing interface to run the program with multiple computer nodes.

First, the critical part of MSAM algorithm is a three sequence alignment method that finds the midpoint to divide a big 3-D cube into two smaller ones recursively. The subdivision of each smaller 3-D cubes split from a common bigger cube is independent from the other. Therefore, a parallel strategy can be developed by subdividing each smaller cube simultaneously by two independent computer nodes. For example, in the first level, given three sequences with length X, Y, and Z, computer node 0 uses forward and backward scoring to find a midpoint on the plane x=x/2, followed by keeping the upper left sub-matrix and sending the lower right part to computer node 1. Then, in the second level, computer node 0 and node 1 simultaneously divides its sub-matrix into two

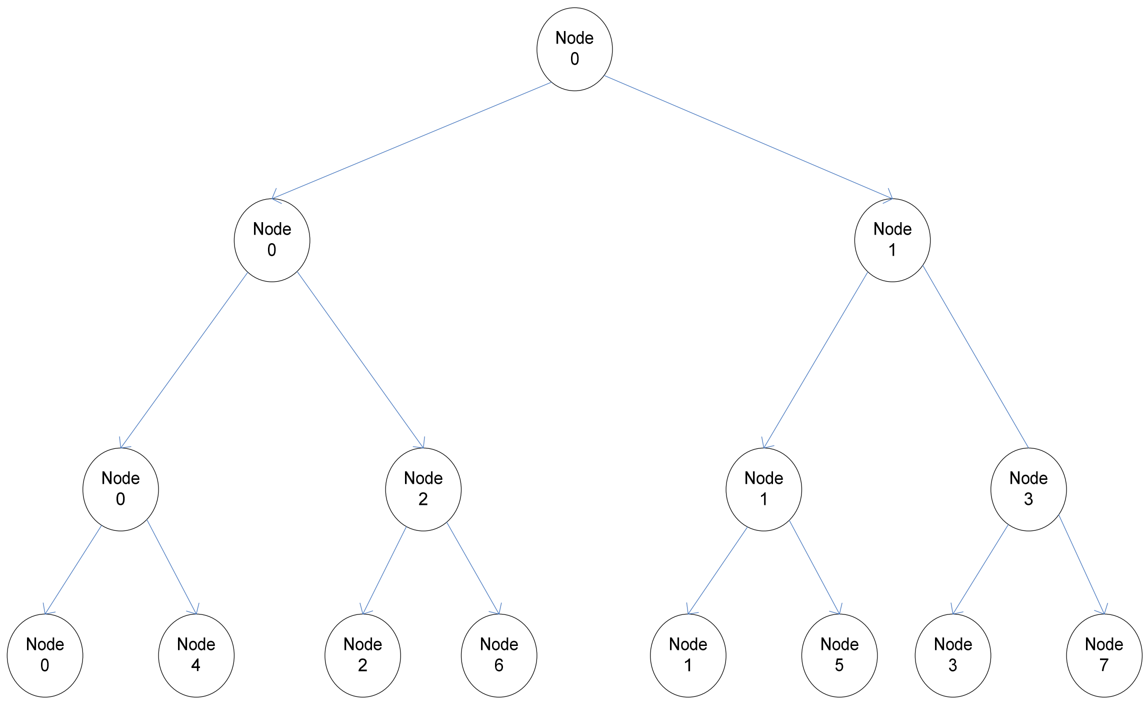

Figure 3. The simultaneous subdivisions by 8 Computer nodes

smaller ones on the plane x= X/2, where this X stands for the subsequence length rather than the entire one and sends a sub-sub-matrix to node 2 and node 3 respectively. Such data partition continues until a minimum size of the sub-matrix is reached or all available computer nodes are assigned tasks. Consequently, for each node assigned a sub-matrix larger than unit (x>1||y>1||z>1), the serial three sequence alignment method is invoked. During this division, each parent node must send both midpoint and ending point coordinates to its child nodes, which provides information of the matrix. Each child node sends the alignment and median sequence back to its upstream parent node from which its sub-matrix comes for results concatenation. Fig. 3

exhibits how this approach works with 8 computer nodes. In general, the relationship of destination and source computer node id's can be represented as the formula in Eq. 4, where n stands for the times that matrix has been divided.

$$P_d = P_s + 2^{(n-1)}$$

Equation 4.

It is observed that the internal nodes in the same tree level can be computed simultaneously using the parallel MSA without interfering with each other before the next level starts. We dynamically create MPI workgroups and assign nodes from each workgroup to perform the parallel MSA for a single internal node. The number of workgroups is determined by the number of internal nodes at each level .The master node of a supercomputer distributes tasks to each workgroup by sending the sequence information to the leading node in each workgroup. Upon receiving the task, each leading node in a workgroup multicasts the information to all workgroup members. After that, each workgroup can start to carry out the parallel MSA. The master node waits for the results returned from the leading node in each workgroup and compares the new sequence with the old one to decide whether to update or not before traversing to the next tree level. Only the sequence that generates lower total branch length of its three edges will be kept and updated. The flowchart is shown in Fig.4.

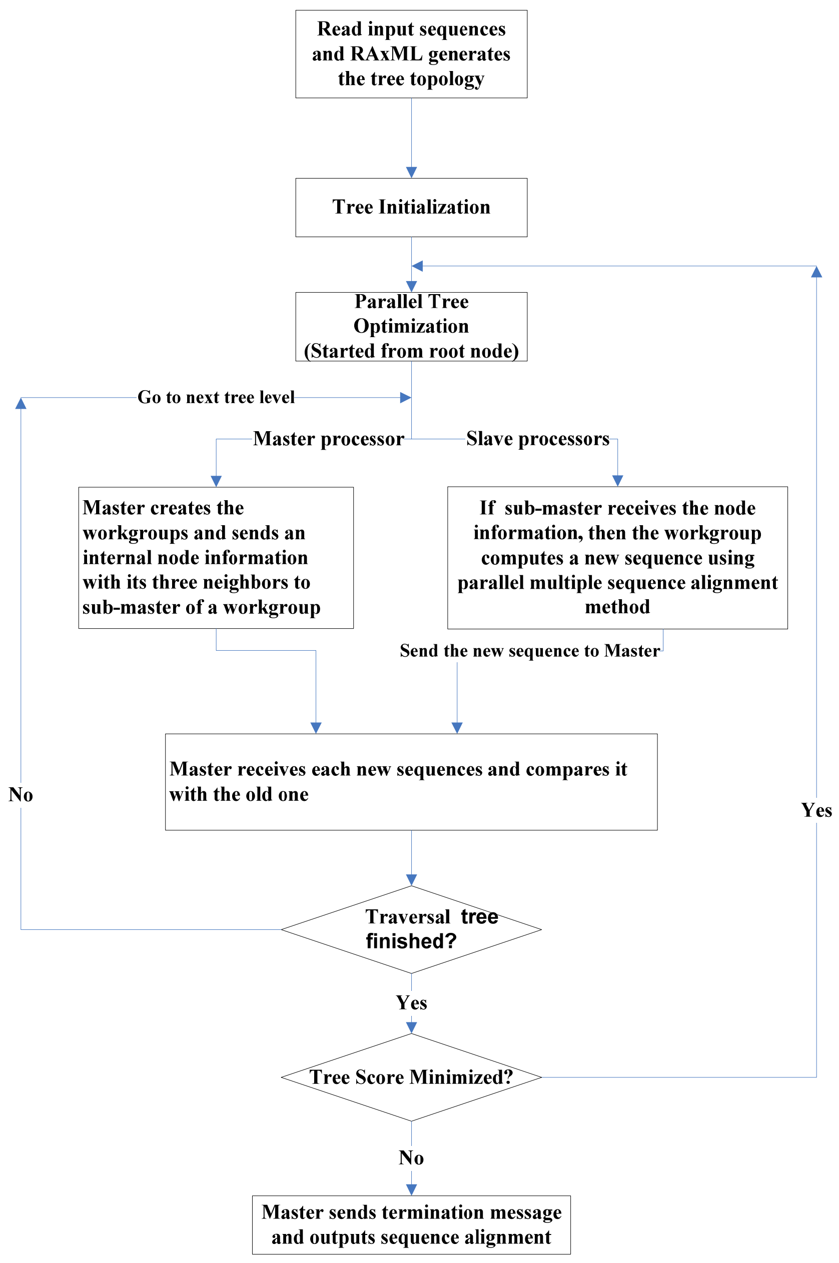
Read input sequences and RAxML generates the tree topology
Tree Initialization
Parallel Tree Optimization (Started from root node)
Go to next tree level
Master processor
Slave processors
Master creates the workgroups and sends an internal node information with its three neighbors to sub-master of a workgroup
If sub-master receives the node information, then the workgroup computes a new sequence using parallel multiple sequence alignment method
Send the new sequence to Master
Master receives each new sequences and compares it with the old one
No
Yes
Traversal tree finished?
Yes
Tree Score Minimized?
No
Master sends termination message and outputs sequence alignment

Figure 4. The program Flowchart

3.3 Pseudo code of the algorithm

Fig. 5 describes the pseudo code of our algorithm, in which we give the pseudo code for each phase. In the optimization phase, the master node dynamically creates workgroups when traversing a new level. There are two key parameters: WorkgroupSize and WorkgroupNumber. WorkgroupSize is an input parameter which defines the node number in a workgroup. WorkgroupNumber parameter, calculated by the master node in BFS_label_tree () function (Fig. 6), provides the information on how many workgroups need to be created. In each level of the tree, this parameter is computed as the total number of internal nodes in this level.

```
Starting Tree:
    Starting Tree topology is generated by parallel RAxML software

Tree Initialization (add input sequences to leaf nodes)
if (myrank==0)
    for level i from bottom to top
        for all internal nodes in level i do
                Label each internal node by pair-wise alignment of
                its two children node sequences
        end for
    end for
end if

Tree Optimization
if (myrank==0)
     while tree score can be minimized do
                Set queue={root}
                BFS_label_tree( queue )
     end while
     Broadcast (termination_tag, all processors)

else
     while message_tag ! = termination_tag do
          if ( myrank <= WorkgroupSize* WorkgroupNumber)
                if ( groupID==0)
                     Receive(node, $P_{master}$)
                     Broadcast(node, workgroup)
                end if

               /*Each workgroup computes a new sequence using */
               /*parallel multiple sequence alignment method */
                   Parallel_MSA(node)

                if (groupID==0)
                   /* Send this new sequence to the master node*/
                       Send (sequence, $P_{master}$)
                end if
          end if
     end while
end if
Output the sequence alignment using MSF format
```

Figure 5. The algorithm of the program

In the BFS_label_tree () function, the master node calculates the internal nodes number, which determines the workgroup number, and sends this parameter to each slave node to dynamically create the workgroup. Also the nodes at the next level of the tree are stored into a queue in order to keep track of them when traversing to the next level. After creating the workgroups, the master node distributes internal nodes with three neighbor’s information to workgroups. The master node waits for new computed sequences and compares them with old ones before traversing to the next tree level.

```
BFS_label_tree(queue)
{
       Set n as the total number of internal nodes at the next level
       Set WorkgroupNumber as the total number of internal nodes at this
       level

       if (n!=0)
           /*  Store the internal nodes at the next level into queue2  */
               Set queue2={node_1,node_2,…,node_i}, where nodes are internal
                                                  nodes at the next level
       end if
       /* Dynamically create the workgroup */
                      Broadcast (WorkgroupNumber, all processors )
                      Call CreateGroup ()

       for i from 1 to  WorkgroupNumber
               /* Send the information of internal node i to  */
               /* the sub-master of workgroup i , called W_i  */
                   Send (queue[i-1],  W_i )
       end for
       for i from 1 to WorkgroupNumber
              /*  receive the median sequence from workgroup */
                  Receive (new_seqeunce, W_any )

              /*  compare this sequence with the old one  */
                 Call Compare (new_sequence, old_sequence)
       end for
       if  (n!=0)
                BFS_label_tree (queue2)
       end if

       Return
}
```

Figure 6. BFS_label_tree function

3.4 The time complexity

In this section, we will compare the time complexity of both sequential and parallel three sequence alignment methods and then give a complexity of the whole parallel program.

The critical part of the three sequence alignment method is to recursively call the score function to find a midpoint. In the sequential version, this score function will be called twice for each matrix, since both the forward and backward scoring are necessary to search the coordinates of the midpoint. So the time complexity of a serial version $t_{sequential}$ is $O(n^3)$ [6] with n representing the length of each sequence. In the parallel version, a binary tree structure is constructed with root starting the division with the original sequence data. The communication costs in the dividing and combining phases need to be considered, the communication time t_{divide} in the dividing phase is $O(\log_2 p \times t_{startup} + t_{comm} \times (3 \times \log_2 p))$ with p denoting number of processors, $t_{startup}$ as the message delay and t_{comm} as the transfer time on a single unit of data. $t_{startup}$ is usually much larger than t_{comm}. In each message, three coordinates of the midpoint are sent. And there are $\log_2 p$ messages in total. The size of the sequence gets reduced by half every time a mid-point of the matrix is found until all *p* processors are occupied. In the combing phase the time cost $t_{combine}$ is similar to t_{divide} in a reverse message sending order, and the median sequence needs to be concatenated and sent back from the lower level to the upper one. The time cost $t_{combine}$ is $O(\log_2 p \times t_{startup} + t_{comm} \times (\frac{n}{2} + \frac{n}{4} + ... + \frac{n}{p}))$.The computing time t_{comp} for each processor at the leaf level is $O((\frac{n}{p})^3)$. The total time $t_{parallel}$ is the summation of t_{comp}, t_{divide} and $t_{combine}$, which is approximately $O(5 \times \log_2 p + n + \frac{n^3}{p^3})$.

For the whole program, each internal node will be computed sequentially in the serial version so that the time complexity $T_{sequential}$ is $O(k \times t_{sequential})$, where k is the total amount of internal nodes. Our parallel program will achieve a much better performance since internal nodes in the same tree level can be calculated simultaneously that the time complexity $T_{parallel}$ is $O((m-1) \times t_{parallel})$, where m is the height of the phylogenetic tree and $t_{parallel}$ is the execution time of the parallel three sequence alignment method.

CHAPTER 4

EXPERIMENTAL RESULTS

R. Gharegozlou [26] wrote a report to compare different MSA software in terms of three aspects: alignment accuracy, computational time, as well as, memory usage .The reports shows that some programs such as TCoffee and ProbCons can provide a highly correct sequence alignment, but cannot handle large sequence datasets because of high computational time and memory cost, especially with the ever-increasing size of sequence datasets scientists are facing[26].Our PMSAPT program parallelizes a dynamic programming-based MSA program with effective memory usage to produce the optimal alignment, and can label every internal node's sequence of the tree generated from a parallel maximum likelihood inference method to reduce the tree cost. Most tree generating programs, such as MUSCLE [31, 32], do not label the internal nodes to enhance the tree information.

Our program combines the advantages of two popular software in phylogenetic field: RAxML and MSAM. However, it is also integrated with some algorithms and strategies to overcome their defects. Our program can handle both protein and DNA sequences with different lengths and output a sequence alignment, which is a major advantage over RAxML that only provides phylogenetic tree structure. To give a sequence alignment, our program utilizes an optimal dynamic programming algorithm implemented in MSAM program to reduce memory usage. However, compared with MSAM program, we use maximum likelihood based method implemented in RAxML software rather than distance-based method to build up a phylogenetics tree structure. Also, we speed up our program using some parallel algorithms, which is much faster than original MSAM software.

In this Chapter, we present three different types of experimental results. First of all, the sequence alignment results in terms of the sum of pairs from different software are compared. Secondly, a tree score table is provided to compare the performance of different tree construction programs. Finally, we compare the execution time of both our parallel program and the serial one. Our parallel program has been executed on Kraken Cray XT5 supercomputer, which is the fastest supercomputer in academia, located at National Institute for Computational Science (NICS), featuring Cray Linux Environment (CLE) 2.2 with 129 TB memory and 8,256 compute nodes. Each compute node has 12 cores with 16 GB of memory. To run the program on Kraken, we need to create a job script, which should specify the resources we want to use and a statement to run the executable. This script should be saved under a directory: /luster/scratch/mzhu. We submit our job via the "qsub" command (See the Kraken instruction : http://www.nics.tennessee.edu/computing-resources/kraken). The Blosum62 [27] substitution matrix is used and the gap open penalty is set to be 16, and the gap extension penalty with value of 3.

4.1 Sequence Alignment Performance Comparison

The BALIBASE benchmark database is used in our experiments. The sequences kept in this database come from FSSP and HOMSTRAD structural database or from some literatures [28, 29, 30]. Since there are a large number of sequence datasets in this database, it is not practical to use each one to test the programs. We focus on two families of sequence datasets, namely RV12 and RV913. RV12 has 44 datasets which contains 20--40% sequences identity, while RV913 contains 27 datasets with 40--80% sequence residue identity. Three sequence datasets in each group are chosen with different sequence length, namely short with length less than 200; medium with length between 200 and 600, and long with length over 600.Furthermore, BALIBASE benchmark provides a reference alignment with MSF format for each sequence dataset and contains a software to calculate the SP (Sum of Pair) scores for test alignment compared with respective reference one. SP scores is calculated in terms of the correctly aligned residue pairs. Higher SP score indicates a better performance of sequence alignment [28, 29, 30].

We compare our program with other popular sequence alignment packages. For example, MUSCLE (multiple sequence comparison by log-expectation) [31, 32] is a progressive /iterative alignment software that is often utilized as a replacement for Clustal. POA [33], developed by Christopher Lee is a partial order alignment software using partial order/hidden Markov model. Both can generate quite high SP scores on BALIBASE benchmark database. We also compare our program with MSAM and MSAM-H programs. MSAM-H utilizes a faster but less accurate MSA method than MSAM software. When aligning three sequences, MSAM-H aligns two closest ones and then aligns with the third one[6]. For every sequence dataset, we run our program three times and calculate the mean SP scores since the RAxML software may construct different tree topologies, but with the same likelihood value.

Table 1 gives the SP scores of six sequence datasets from five programs. For the short group, PMSAPT can achieve the highest SP score in *BOX017* dataset, while MSAM can get the best result in *BB12003* dataset. For the

Table 1. SP scores from five programs

Sequence No. / program	RV12			RV913		
	BB12003 (Short)	BB12005 (Medium)	BB12030 (Long)	BOX017 (Short)	BOX142 (Medium)	BOX082 (Long)
POA	0.911	0.846	0.854	0.695	0.971	**0.974**
MUSCLE	0.903	**0.893**	**0.888**	0.716	**0.979**	0.97
MSAM-H	0.852	0.88	0.372	0.684	0.953	0.696
MSAM	**0.931**	0.88	0.61	0.714	**0.979**	0.695
PMSAPT	0.905	0.786	0.681	**0.722**	0.972	0.691

medium and long groups, MUSCLE program seems to achieve better results than others.

We also observe that our program and MSAM both can achieve better results than MSAM-H in average. This is, to some extent, because of the optimization mechanism selected. MSAM-H uses a simplified algorithm to compute median sequence, which is faster but less accurate. Nevertheless, as we parallelize the optimization phase on three sequence alignment, the optimal approach is used to produce the best median sequence out of the three using dynamic programming approach.

4.2 Tree Score

We also compare the tree score generated by PMSAPT with that of MSAM and MSAM-H. All of these three programs make use of the same tree score calculation strategy as used in MSAM package. In other words, the tree score is defined as the sum of all edge lengths of the tree. Each edge length is calculated by the pairwise distance between two sequences in the edge .The objective of the optimization phase in each program is to minimize the tree score [6].So the tree score is able to be considered as an important factor to measure a program's performance.

Table 2 illustrates the absolute tree scores of six sequence datasets from three programs: MSAM-H, MSAM and PMSAPT. The absolute values are used for consistence. The smaller the tree score is, the better the quality of the tree is. There are some influential factors on tree score. The first factor is node labeling algorithm. Both MSAM and MSAM-H programs make use of Neighbor Joining Method to build up the tree topology. However, MSAM uses a dynamic programming technique to compute the more accurate median sequence. Table 2 shows that MSAM can generate a smaller tree score in average than MSAM-H. Another important factor is the layout of the tree topology. With the same input sequence file and substitution matrix, different tree topologies can produce different tree scores. Our program exploits parallel RAxML software to construct the tree topology that excels over the progressive and greedy N-J method. Table

2 shows the tree scores generated by our program are smaller than those from MSAM in most cases.

Table 2. Absolute score from three programs

Sequence No. / Program	RV12			RV913		
	BB12003 (Short)	BB12005 (Medium)	BB12030 (Long)	BOX017 (Short)	BOX142 (Medium)	BOX082 (Long)
MSAM-H	3474	11352	18078	4316	13759	30493
MSAM	3365	10841	18902	4264	13754	29970
PMSAPT	3038	10402	14993	4271	13510	29613

From Table 1 and Table 2, we can find that a better tree score value does not mean a better SP score. This may be because of different measurement methods applied or because of the reference alignment itself provided by BALIBASE benchmark software.

4.3 Serial and Parallel Comparison

In MSAM program, the execution time of optimization phase is a bottleneck so that MSAM is not practical or time-efficient to deal with large sequences datasets .In Chapter 3, we introduced some parallel strategies in our program to parallelize the optimization phase. In this section, with the same tree topology and initialization phase, we make some experiments to compare the execution time of the serial optimization version with parallel one. The serial optimization version is the same with that implemented in MSAM program. To measure the performance, one parameter is introduced here: speedup factor. The speedup factor defined in Eq.5 indicates how much a parallel algorithm is faster than the serial one.

$$SpeedupFactor = \frac{T_{sequential}}{T_{parallel}} = \frac{\mathrm{k} \times \mathrm{t}_{sequential}}{(m-1) \times t_{parallel}} = \frac{k \times p^3 \times n^3}{(m-1) \times (p^3 \times 5 \times \log_2 p + n^3 + p^3 \times n)}$$

Equation 5. Speedup factor formula

We first select BB12030 [30] sequences dataset as our test data. In this dataset, there are six sequences with average length over 790. Our parallel PMSAPT is run on 8, 16, and 32 processors in each workgroup respectively. Table 3 shows the different execution time and speed up values. It is obvious that the parallel program is much faster than the serial version due to the parallel update of the internal nodes at the same level.

It is conjectured that if there are significant number of tree levels that contains multiple internal nodes, we would be able to achieve higher time saving due to the parallel labeling of all internal nodes at the same level. We select another sequence dataset, BB12004 [30] also from BALIBASE benchmark database to test the effect of program performance with larger problem size. The BB12004 dataset contains 15 sequences with average length around 249. The tree topology built for this sequence dataset contains more tree levels with multiple internal nodes than BB12030. We observe a higher speed up from this dataset. Table 4 demonstrates the time difference.

Table 3 *BB12030* dataset comparison

Version	Execution Time	Speedup Factor
Serial	14.193 *hours*	N/A
Parallel(8 nodes)	7.782 *hours*	1.8238
Parallel(16 nodes)	7.780 *hours*	1.8243
Parallel(32 nodes)	7.776 *hours*	1.8252

Table 4 *BB12004* dataset comparison

Version	Execution Time	Speedup Factor
Serial	2.257 *hours*	N/A
Parallel(8 nodes)	0.951 *hours*	2.3733
Parallel(16 nodes)	0.945 *hours*	2.3884
Parallel(32 nodes)	0.950 *hours*	2.3758

$$ratio = \frac{t_{comp}}{t_{comm}} = \frac{(\frac{n}{p})^3}{n + 5 \times \log_2 p} = \frac{n^3}{p^3 \times (n + 5 \times \log_2 p)}$$

Equation 6. Computation/communication ratio

Compare with serial program, our parallel program can achieve better performance on execution time saving. However, there are still many things we can do to improve the efficiency of our parallel program. One thing we can observe is that when we keep adding more processors, no more speed up can be achieved. Such bottleneck problem is mainly caused by the high communication cost in the parallel MSA function. To prove this, we can calculate the computation/communication ratio using Eq.6 and t_{comp}, t_{divide}, $t_{combine}$ from Section 3.4. The computation/communication ratio defined in Eq.6 highlights effect of communication with increasing problem size and system size. The ratio is $\frac{n^3}{p^3 \times (n + 5 \times \log_2 p)}$, with p denoting number of processors and n representing the length of each sequence. From Eq.5 and Eq.6, we can find that with same sequences, the more processors usage does not change much for speed up value, but results in the lower ratio value. That means the overheads of synchronization and communication is greater. To address this issue, it is planned that OpenMP and multi-core technologies will be exploited to resolve these issues. Also, we can optimize our program's structure to reduce communication cost. For example, instead of sending each type of data one by one within parallel three sequence alignment method, we can create a new data type to save these data at first and then send it via one message, in order to reduce the number of communication message.

CHAPTER 5
CONCLUSION

The proposed work is to develop a parallel approach to conduct multiple sequence alignment, as well as, tree node labeling. The parallel RAxML software is integrated to generate a phylogenetic tree under the maximum likelihood criteria, which performs better than distance based methods in terms of the tree score. Then, pairwise sequence alignment is used to initialize the internal node labeling. We utilize and adapt the optimization phase implemented in MSAM to iteratively label the tree nodes in order to minimize the total tree score and achieve a better final sequence alignment. The most time consuming function for the optimization phase is three sequence alignment and is parallelized in a divide and conquer strategy. The labeling of internal nodes at the same level is conducted at the same time by different workgroups that are dynamically created. The two levels of parallelization on both the MSA and tree traversal enable us to fully utilize the abundant computing resources available in a supercomputer. The experiments results show that our PMSAPT can significantly reduce the computing time. In addition, the tree quality of PMSAPT is comparative to or even better than some popular software using BALIBASE benchmark experiments.

REFERENCES

[1] Tree of Life Project. What is Phylogeny? *Tree of Life Web project*, 6 May 2008 <http://tolweb.org>

[2] R. Potter. Constructing Phylogenetic Trees using Multiple Sequence Alignment, *Master Thesis*, 2008

[3] J. Felsenstein. Inferring Phylogenies, *Sinauer Associates, Sunderland*, MA,2004

[4] T. Jiang, and F.Liu .Tree alignment and reconstruction application software, 1996

[5] G. Lancia , and R.Ravi . GESTALT: Genomic Steiner Alignments. *10th Annual Symposium on Combinatorial Pattern Matching*, pp.101-114, 1999

[6] F.Yue, and J. Tang. A New Approach for Tree Alignment Based on Local Re-Optimization, *2008 International Conference on BioMedical Engineering and Informatics*, vol. 1, pp.34-38, 2008

[7] N.Saitou, and M.Nei. The neighbor-joining method: A new method for reconstructing phylogenetic trees, *Mol. Biol. Evol.*, Vol.4, pp. 406-425 ,1987

[8] P.Sneath, and R.Sokal. Numerical Taxonomy, *W.H. Freeman and Company*, pp. 230-234, June 1973.

[9] D.Swofford. and G.Olsen. Phylogeny reconstruction, *Molecular systematic*, pp. 411-501,1990

[10] D.Swofford, G.Olsen, P.Waddell, and D.Hillis. Phylogenetic inference, *Molecular Systematics ,second edition*, pp.407-514,1996

[11]G. Giribet. Efficient Tree Searches With Available Algorithms, *Evolutionary Bioinformatics 2007*, pp. 341-356, 2007

[12] A. Stamatakis. Distributed and Parallel Algorithms and Systems for Inference of Huge Phylogenetic Trees based on the Maximum Likelihood Method, *PH.D. thesis*, 2004

[13] S. Roch. A Short Proof that Phylogenetic Tree Reconstruction by Maximum Likelihood is Hard, *IEEE/ACM Transactions on Computational Biology and Bioinformatics(TCBB),* pp.92 , Vol. 3, Issue 1, Jan. 2006

[14] B.Chor, and T.Tuller. Maximum likelihood of evolutionary trees: hardness and approximation*, Bioinformatics*, pp. 97-106, Vol.21, No.1 2005

[15] L.Vinh ,and A.Haeseler. IQPNNI: Moving Fast Through Tree Space and Stopping in Time, *Mol.Biol.Evol* , pp.1565-1571,Vol. 21,No.8 , 2004

[16] S. Guindon, and O.Gascuel. A simple ,fast and accurate algorithm to estimate large phylogenies by maximum likelihood, *Systematic Biology*, pp.696-704,Vol.52, No.5, 2003

[17] O.Gascuel . BIONJ: An improved version of the NJ algorithm based on a simple model of sequence data, *Mol.Biol.Evol* , pp.685-695,Vol.14, No.7,1997.

[18] A.Stamatakis, F.Blagojevic, C.Antonopoulos, D.Nikolopoulos . Exploring new Search Algorithms and Hardware for Phylogenetics: RAxML meets the IBM Cell, *In Journal of VLSI Signal Processing Systems*, pp. 271-286, Vol.48, No.3, 2007

[19] A.Stamatakis, T.Ludwig, and H.Meier. RAxML-III: A Fast Program for Maximum Likelihood-based Inference of Large Phylogenetic Trees, *Bioinformatics*, Vol.21, No.4, pp.456-463, 2005

[20] A.Stamatakis. RAxML-VI-HPC: Maximum Likelihood-based Phylogenetic Analyses with Thousands of Taxa and Mixed Models . *In Bioinformatics,* pp.2688-2690, Vol.22, No.21, 2006

[21] M.Ott, J.Zola, S.Aluru, A.Stamatakis. Large-scale Maximum Likelihood-based Phylogenetic Analysis on the IBM BlueGene /L, *In Proceedings of IEEE/ACM Supercomputing (SC 2007) conference, Reno, Nevada*, November 2007.

[22] A.Stamatakis . Phylogenetic Models of Rate Heterogeneity: A High Performance Computing Perspective, *In Proceedings of 20th IEEE/ACM International Parallel and Distributed Processing Symposium (IPDPS2006),High Performance Computational Biology Workshop, Rhodos ,Greece*, April 2006

[23] F. Yue and J. Tang. A Divide-and-Conquer Implementation of Three Sequence Alignment and Ancestor Inference*, BIBM 2007*, 2007

[24] E.Myers and W.Miller . Optimal Alignments in Linear Space , *Computer Applications in the Biosciences*, pp.11-17,Vol.4, No.1 , 1988

[25] D.Hirschberg. A linear space algorithm for computing maximal common subsequences*, Communications of the ACM*, Vol.18, No.6, pp.341-343,June 1975

[26] R. Gharegozlou. Protein Multiple Sequence Alignment:Benchmarks and Comparison, *Biochemistry 218 Final Project*,March,2009

[27] S.Henikoff and J.Henikoff. Amino acid substitution matrices from protein blocks, *Proc.Natl.Acad.Sci.USA,*Vol.89,pp.10915-10919, November,1992

[28] A.Bahr,J.Thompson,J.Thierry and O.Poch. BAliBASE(Benchmark Alignment dataBASE):enhancements for repeats,transmembrane sequences and circular permutations,*Nucleic Acids Research,*Vol.29,No.1,pp.323-326,2001

[29]J.Thompson, F.Plewniak and O.Poch. A comprehensive comparison of multiple sequence alignment programs, *Nucleic Acids Research,*Vol.27,No.13,pp.2682-2690,1999

[30] J.Thompson, F.Plewniak and O.Poch. BAliBASE: A benchmark alignment database for the evaluation of multiple alignment programs, *Bioinformatics*, Vol.15,No.1,pp 87-88,1999

[31] R.Edgar. MUSCLE: multiple sequence alignment with high accuracy and high throughput,*Nucleic Acids Research,*Vol.32,No.5, pp. 1792-1797,2004

[32] R.Edgar. MUSCLE: a multiple sequence alignment method with reduced time and space complexity,*BMC Bioinformatics*,Vol.5,No.1,pp.113,2004

[33] C.Lee.Generating consensus sequences from partial order multiple sequence alignment graphs,*Bioinformatics*,Vol.19,No.8,pp.999-1008,2003

VITA

Graduate School
Southern Illinois University

Jingjing Wang Date of Birth: February 2, 1986

2316 Coach Road, Long Grove, IL 60047

jeffy.wang1986@gmail.com

Southeast University
Bachelor of Engineering, Electrical Science and Technology, June 2008

Thesis Title:
A Parallel Approach to Multiple Sequences Alignment and Phylogenetic Tree Labeling

Major Professor: Mengxia Zhu