MT-ClustalW: Multithreading Multiple Sequence Alignment

Kridsadakorn Chaichoompu[1], Surin Kittitornkun[1], and Sissades Tongsima[2]

[1]Dept.of Computer Engineering
Faculty of Engineering
King Mongkut's Institute of Technology
Ladkrabang, Bangkok 10520 ,Thailand
{s7060809,kksurin}@kmitl.ac.th

[2]National Center for Genetic Engineering
and Biotechnology
113 Paholyothin, Klong 1, Klong Luang
Pathumtani 12120, Thailand
sissades@biotec.or.th

Abstract

ClustalW is the most widely used tool for aligning multiple protein or nucleotide sequences. The alignment is achieved via three stages: pairwise alignment, guide tree generation and progressive alignment. This paper analyzes and enhances a multithreaded implementation of ClustalW called ClustalW-SMP for higher throughput. Our goal is to maximize the degree of parallelism on multithreading ClustalW called MultiThreading-ClustalW (MT-ClustalW). As a result, bioinformatics laboratories are able to use this MT-ClustalW with much less energy consumption on multicore and SMP (Symmetric MultiProcessor) machines than that of PC clusters. The experiment results show that the MT-ClustalW framework can achieve a considerable speedup over the sequential ClustalW and original multithreaded ClustalW-SMP implementations.

1 Introduction

Multiple sequence alignment of many nucleotides or amino acids is an important tool in bioinformatics. The multiple sequence alignment technique identifies diagnostic patterns or motif to characterize protein families. It can also detect or demonstrate homology between new sequences and existing families of sequences. Thus it helps predict the secondary and tertiary structures of the new sequences which is an essential prelude to molecular evolutionary analysis. Many multiple sequence alignment tools have been proposed to reduce the high computation time of fully performing alignment of all sequences. Implementations of various multiple sequence alignment heuristics include MSA [1], PRALINE [2], T-Coffee [3], DIALIGN P [4], MAFFT [5] and ClustalW [6]. ClustalW particularly is the most popular sequential program for multiple sequence alignment, and CLUSTALX [7] is a graphical interface version of ClustalW.

Caching and parallel methods are focused to improve the computation for the better throughput. The efficient execution of multiple sequence alignment method is concerned in the data server and the cluster of workstations about the effect of data caching. [8, 9] The parallel version of ClustalW is presented by SGI [10]. The SGI parallel version shows speedup of up to 10 folds when running ClustalW on 16 CPUs [11]. pCLUSTAL presents a parallel version of CLUSTALW. In contrast to the commercial SGI parallel Clustal version, which requires an expensive SGI multiprocessor system, pCLUSTAL can run on PC clusters as well [12]. ClustalW-MPI [13] is another interesting parallel ClustalW which uses a message-passing library called MPI and runs on distributed workstation clusters. Moreover, the parallel ClustalW with Dynamic Scheduling [14] proposes the algorithm that divides a progressive alignment into subtasks and schedules them dynamically. Besides the software approach, a new approach to MSA on reconfigurable hardware platforms to gain high performance at low cost. Fine-grained parallel processing elements (PEs) are designed for the computation of pairwise distances between protein sequences. [15]

This work present a parallel ClustalW algorithm using multithreading technique, called MT-ClustalW. Our achievement is considerable and done by dual-core processor which is much less expensive workstation than the high-end SGI multiprocessor.

This paper is organized as follows: Section 1 reviews the other parallel version of multiple sequence alignment tool, ClustalW. Section 2 describes the sequential ClustalW algorithm. Section 3 analyzes the SMP ver-

1-4244-0054-6/06/$20.00

sion of ClustalW. Section 4 presents the proposed parallel ClustalW, called MT-ClustalW. Section 5 demonstrates the experimental results. Finally, Section 6 draws the conclusion of this work.

2 Sequential ClustalW

ClustalW has become the most popular algorithm for multiple sequence alignment. This program implements a progressive method for multiple sequence alignment. As a progressive algorithm, ClustalW adds sequences one by one to the existing alignment to build a new alignment. The order of the sequences to be added to the new alignment is indicated by a precomputed phylogenetic tree, which is called a guide tree. The guide tree is constructed using the similarity of all possible pairs of sequences. The algorithm consists of 3 phases that are described below:

Stage 1 —Distance Matrix: All pairs of sequences are aligned separately in order to calculate a distance matrix based on the percentage of mismatches of each pair of sequences.

Stage 2 —Neighbor joining: The guide tree is calculated from the distance matrix using a neighbor joining algorithm [16]. The guide tree defines the order which the sequences are aligned in the next stage.

Stage 3 —Progressive alignment: The sequences are progressively aligned following the guide tree.

Now we discuss the complexity for all stages. Give N sequences and sequence length L, calculating the distance matrix in stage 1 takes $\mathcal{O}(N^2L^2)$ time. A neighbor joining algorithm is $\mathcal{O}(N^4)$ time for constructing the guide tree in stage 2. And for the last stage, progressive alignment is $\mathcal{O}(N^3+NL^2)$ time. The summary is shown in Table 1 [17].

3 Analysis : ClustalW-SMP

ClustalW-SMP (version 0.99–9) is the SMP version of ClustalW version 1.82, was created by O. Duzlevski [18]. We have analyzed ClustalW-SMP to find the bottleneck for improving the throughput and the speedup of the SMP version. Hence we used the Intel thread profiler tool [19] to analyze the SMP version. The profiler shows that the SMP version is not yet fully parallelized in Figure 1. This figure shows that only Distance matrix and Progressive alignment stages are forced into the threads, but not in Neighbor joining stage. Therefore, we modified the code of both the sequential ClustalW and the SMP versions and measure the execution time of three major stages. The two criteria to be analyzed are: first, the elapsed time of the three stages which we want to find the time ratio among the three stages and second, the number of threads which we want to find the relation between the thread number and the PC efficiency.

The protein sequence data sets from the National Center for Biotechnology Information (NCBI) are used as our test data whose sequence lengths and sequence numbers are shown in table 2. The profiles are done on 3GHz Intel Pentium IV processor with Hyper thread technology and 1 GB of memory. The elapsed times of ClustalW and ClustalW-SMP are compared which in this case ClustalW-SMP is faster. Although the executions are done in 2, 4 and 8 threads, the execution times of ClustalW-SMP still nearly, that shown in Table 3. Table 4 exhibits the elapsed times of each stage. In neighbor joining stage, the execution times are similar considering the same sequence numbers while the sequence lengths are fixed.

Figures 2 and 3 show the time ratios of the three stages of ClustalW-SMP with varying the number of sequences and the sequence lengths are fixed at 200 and 800 amino acids. The time ratios of neighbor joining stage with the fewer sequence numbers are more than the time ratios of neighbor joining stage with the more sequence numbers while the sequence lengths are fixed. That means if we can parallel the tasks in the neighbor joining stage, the execute time should be reduced more. This helps the alignment which needs to align the too many sequences to spend the faster time.

4 MT-ClustalW

We have modified the neighbor joining part of ClustalW-SMP. The implementation was done by using `pthread` library with `MUTEX` object as a synchronization object. Figure 4 shows the code for guide tree stage, the 4 nested loops has $\mathcal{O}(N^4)$ time complexity. We then break the 2 inner loops to be executed in parallel.

The flowchart of the thread synchronization is described in Figure 5. After the 2 inner loops were moved to be the parallel function, the parameters are set in the main thread. The main thread starts the parallel function by calling `signal(start)` and waits for the reply signal to execute next loop. The `thread_num` variable is the maximum number of thread that we can use. In parallel function, it executes parallel tasks after waiting for the start signal. Then the parallel

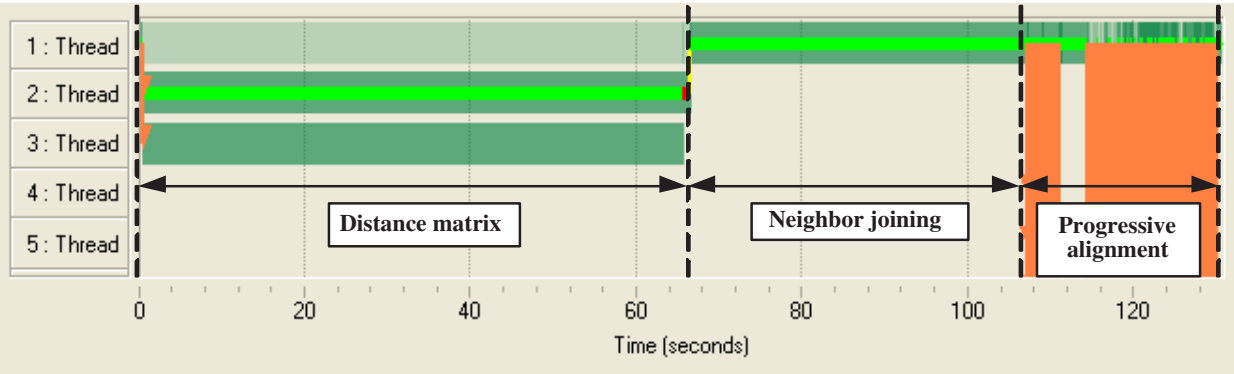

Figure 1. Intel Thread Profiler shows the time-line of the experiment with 400 protein sequences and about 200 amino acids in length .

Table 1. Complexity of the sequential ClustalW. The big-O asymptotic complexity of the elements of ClustalW as a function of L, the sequence length, and N, the number of sequences, retaining the highest-order terms in N with L fixed and vice versa.

Stage	Time complexity
Distance Matrix	$\mathcal{O}(N^2L^2)$
Neighbor joining	$\mathcal{O}(N^4)$
Progressive alignment	$\mathcal{O}(N^3 + NL^2)$
Total	$\mathcal{O}(N^4 + L^2)$

Table 2. Sequence length and Sequence number of the protein sequence as test data

Variables	Values
Average sequence lengths	200, 400, 600, 800
Sequence number	200, 400, 600, 800
Maximum thread number	2, 4, 8

Table 3. Elapsed times of ClustalW compare with ClustalW-SMP running in 2,4,8 threads

#seq-#len	ClustalW (*sec*)	ClustalW-SMP (*sec*)		
		2 Threads	4 Threads	8 Threads
200-200	26	22	22	22
200-800	494	420	414	415
400-200	130	118	116	116
400-800	1,932	1,667	1,665	1,676
600-200	496	469	462	462
600-800	4,544	3,943	3,939	3,959
800-200	1,395	1,347	1,341	1,341
800-800	8,539	7,500	7,480	7,520

Table 4. Elapsed times of ClustalW-SMP in each stage

# seq-# len	ClustalW-SMP (*sec*)		
	Distance Matrix	Neighbor Joining	Progressive Alignment
200-200	17	1	3
200-800	379	1	34
400-200	65	41	10
400-800	1,538	41	86
600-200	149	293	21
600-800	3,481	293	166
800-200	259	1,044	38
800-800	6,185	1,050	265

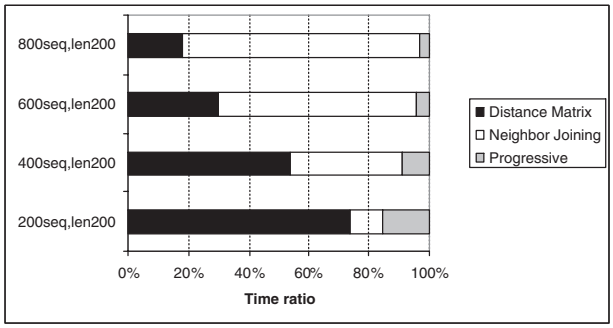

Figure 2. Time ratio of the three stages of ClustalW-SMP. The number of sequences are varied but the sequences lengths are fixed at about 200 amino acid

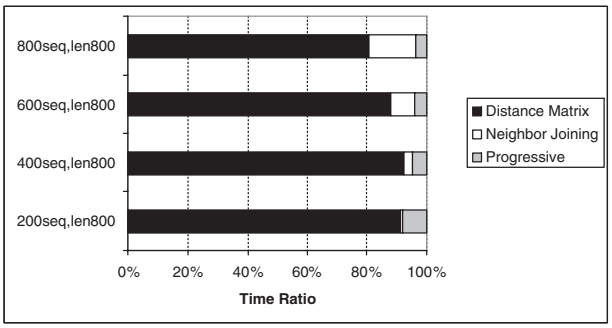

Figure 3. Time ratio of the three stages of ClustalW-SMP. The number of sequences are varied but the sequences lengths are fixed at about 800 amino acid

```
for(nc=1; nc<=(last_seq-first_seq+1-3); ++nc) {
        sumd = 0.0;
        for(j=2; j<=last_seq-first_seq+1; ++j)
        for(i=1; i<j; ++i) {
                tmat[j][i] = tmat[i][j];
                sumd = sumd + tmat[i][j];
                }
        tmin = 99999.0;
        for(jj=2; jj<=last_seq-first_seq+1; ++jj)
                if(tkill[jj] != 1)
                        for(ii=1; ii<jj; ++ii)
                                if(tkill[ii] != 1) {
                                        diq = djq = 0.0;
                                        for(i=1; i<=last_seq-first_seq+1; ++i) {
                                                diq = diq + tmat[i][ii];
                                                djq = djq + tmat[i][jj];
                                        }
                                        dij = tmat[ii][jj];
                                        d2r = diq + djq - (2.0*dij);
                                        dr  = sumd - dij -d2r;
                                        fnseqs2 = fnseqs - 2.0;
                                        total= d2r+ fnseqs2*dij +dr*2.0;
                                        total= total / (2.0*fnseqs2);
                                        if(total < tmin) {
                                                tmin = total;
                                                mini = ii;
                                                minj = jj;
                                        }
                                }
                        }
                                ...
```

THREAD BODY

Figure 4. The source code of the guide tree stage

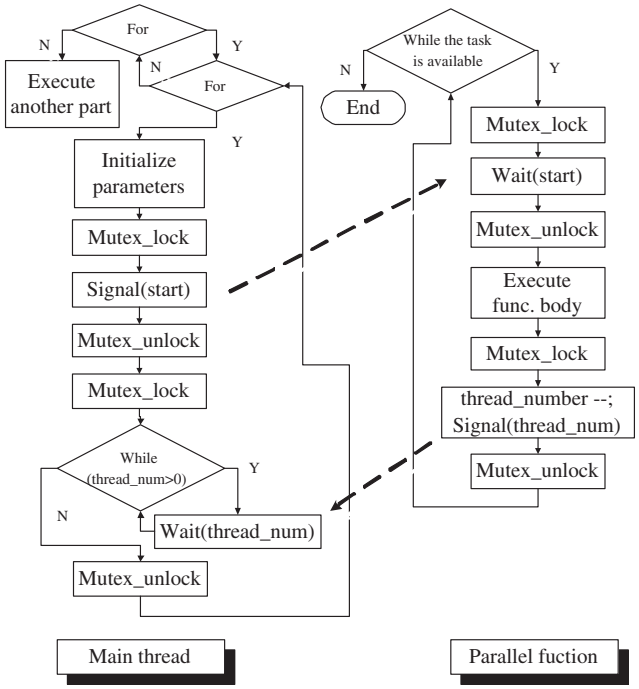

Figure 5. Flowchart of the thread synchronization

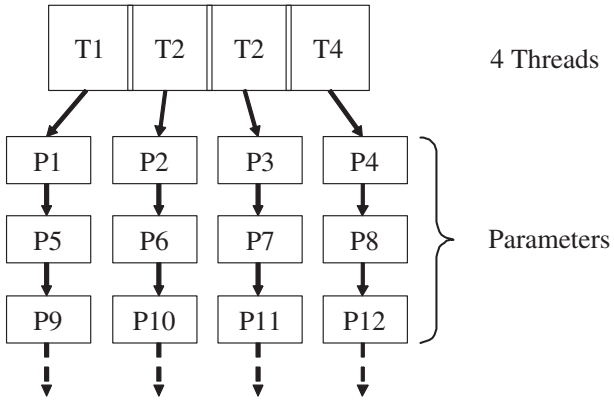

Figure 6. All loads are distributed to all thread functions equally as the parameters

function decreases the `thread_num` variable by one and send the `thread_num` signal to the main thread. After calling `signal(thread_num)`, the parallel function reaches the while condition and determine if there are more tasks to be executed. If so, the parallel function continues waiting for the start signal but if not, the function will cease. Back to the main thread, the main thread proceeds to the next loop after all parallel threads complete. When all loops are done, the main thread continues in the another part.

We reduce `pthread` creation overhead by creating the array of parallel functions. The set is equal to the maximum thread number that is set the user. All loads are distributed to all thread functions equally and taken by the parallel functions to execute the tasks. The logic diagram of the algorithm shows in Figure 6.

5 Experiment Results

We implement our algorithm in C programming language and utilize the `pthread` library. The experiment was conducted using a 2.8GHz Intel Pentium D (Dual-core) processor with 2 GB of memory. This computer runs MS Windows XP pro service pack 2 with no other applications installed. The elapsed times are measured as well as the wall clock time between the distance matrix stage and the progressive alignment stage. All measured times include the times that were taken to read the input data from a file and write the solution into a file. Furthermore, all measured times were measured when there is no one using the system except this experiment.

Our experiments measured the following: (1) The elapsed times as a function of the number of sequences in the guide tree stage. (2) Relative speedup (ratio of the elapsed time of ClustalW and the elapsed time of MT-ClustalW) as a function of the number of sequences.With the same test data set, Figure 7 shows the elapsed times for the ClustalW and MT-ClustalW results of the Neighbor joining stage as a function of number of sequences when running 8 threads. The MT-ClustalW successfully decreases the execution time. Figure 8 shows the speedup for the ClustalW-SMP and MT-ClustalW results of 8 threads as a function of number of sequences as compared to ClustalW. It can be observed that higher speedup can be achieved with the shorter sequence length to be aligned. When the sequence length is small, the speedup increase. When the sequence length is large, the speedup will decrease because the elapsed time of the guide tree stage depend on the number of sequences but the other stages depend on both the number of sequences and the sequence length. If the number of sequences is large and the sequence length is long, the ratio of the elapsed

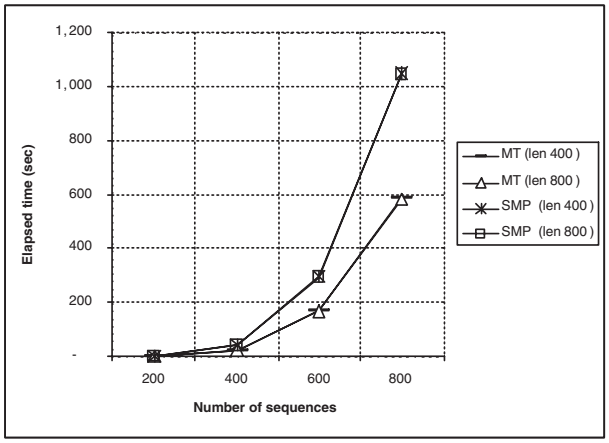

Figure 7. Elapsed times for the ClustalW and MT-ClustalW results of Neighbor joining stage as a function of number of sequences (8 threads)

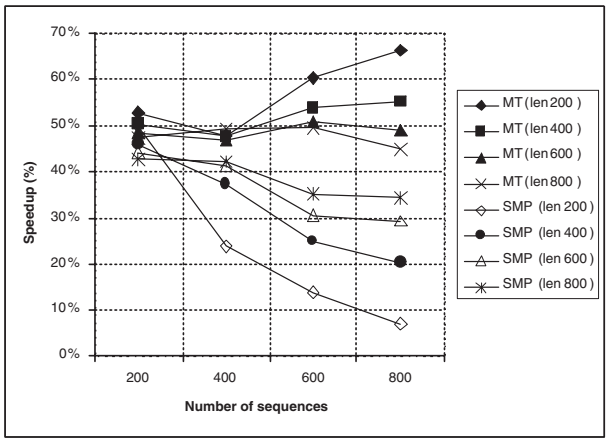

Figure 8. Speedup for the ClustalW-SMP and MT-ClustalW results of 8 threads as a function of number of sequences. Both speedup are compared with ClustalW

time of the guide tree stage and the elapsed time of the other stages is lower so the speedup decrease and vice versa.

Now, we will focus on the speedup of MT-ClustalW as a function of number of threads. In Figure 9, the speedup increases from 2 threads to 4 threads then decreases after 4 threads. Also in Figure 10, the speedup is almost the same as the speedup of Figure 9 except the 800 sequence data. The speedup seems to be lower then steady when the thread number is raised because the limitation of the machine resources.

The proposed method fully utilizes the multithreading feature in ClutalW and achieves the better speedup of ClustalW. MT-ClustalW is faster than ClustalW-SMP especially when the number of sequences is large. This will be compatible with the massive work for the alignment and gains more throughput.

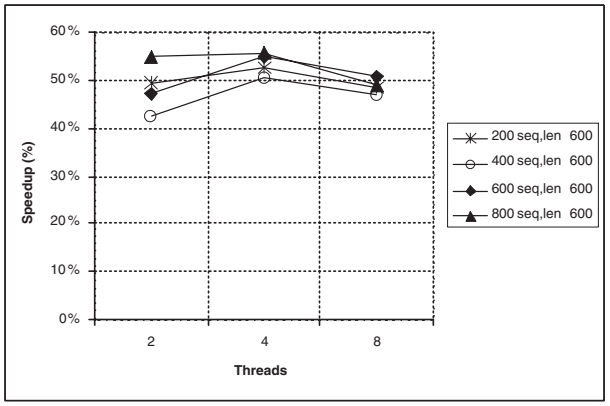

Figure 9. Speedup of the MT-ClustalW results as a function of number of threads that compared to ClustalW. The sequence lengths are fixed at 600 amino acids

6 Conclusion

In this paper, we presented a fully multithreading version of ClustalW called MT-ClustalW which significantly improves the performance of the traditional ClustalW. In the proposed MT-ClustalW, all stages: the distance matrix, the guide tree and the progressive alignment; are divided into a number of threads and executed on a Pentium-D machine. The proposed algo-

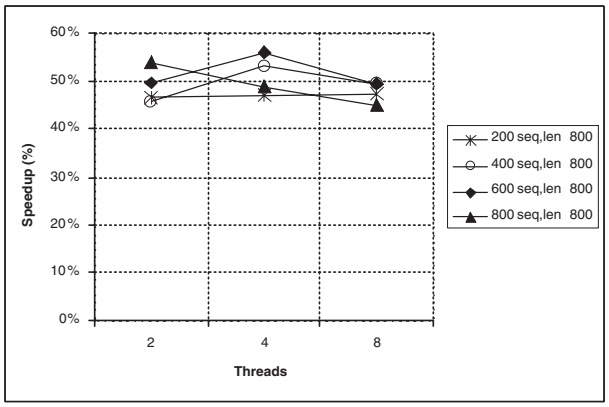

Figure 10. Speedups of the MT-ClustalW results as a function of number of threads that compared to ClustalW. The sequence lengths are fixed at 800 amino acids

rithm shows better speedups than the ClustalW-SMP and much better than that of sequential ClustalW.

References

[1] D. J. Lipman, S. F. Altschul, and J. D. Kececioglu, "A tool for multiple sequence alignment," *Proc. Nail. Acad. Sci. USA*, vol. 86, pp. 4412–4415, 1989.

[2] V. A. Simossis and J. Heringa, "Praline: a multiple sequence alignment toolbox that integrates homology-extended and secondary structure information," *Nucleic Acids Research, 2005*, vol. 33, pp. 289–294, 2005.

[3] C. Notredame, D. G. Higgins, and J. Heringa, "T-coffee: A novel method for fast and accurate multiple sequence alignment," *IDEAL J. Mol. Biol. (2000)*, vol. 302, pp. 205–217, 2000.

[4] M. Schmollinger, K. Nieselt, M. Kaufmann, and B. Morgenstern, "Dialign p: Fast pair-wise and multiple sequence alignment using parallel processors," *BMC Bioinformatics 2004*, vol. 5, no. 128, 2004.

[5] K. Katoh, K. Misawa, K. Kuma, and T. Miyata, "Mafft: a novel method for rapid multiple sequence alignment based on fast fourier transform," *Nucleic Acids Research, 2002*, vol. 30, no. 14, pp. 3059–3066, 2002.

[6] J.D. Thompson, D.G. Higgins, and T.J. Gibson, "Clustal w: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, positions-specific gap penalties and weight matrix choice," *Nucleic Acids Research 22*, vol. 22, no. 22, pp. 4673–4680, 1994.

[7] J. D. Thompson, T. J. Gibson, F. Plewniak, F. Jeanmougin, and D. G. Higgins, "The clustal_x windows interface: flexible strategies for multiple sequence alignment aided by quality analysis tools," *Nucleic Acids Research, 1997*, vol. 25, no. 24, pp. 48764882, 1997.

[8] U. Catalyurek, E. Stahlberg, R. Ferreira, and J. Saltz, "Improving performance of multiple sequence alignment analysis in multi-client environments," *Proceedings of the First IEEE Int. Workshop on High Performance Computational Biology (HiCOMB 2002, IPDPS 2002)*, 2002.

[9] U. Catalyurek, M. Gray, T. Kurc, J. Saltz, E. Stahlberg, and R. Ferreira, "A component-based implementation of multiple sequence alignment," *Proceedings of the 2003 ACM Symposium on Applied Computing (SAC2003)*, pp. 122–126, 2003.

[10] http://www.sgi.com, "Silicon graphics, inc," .

[11] D. Mikhailov, Haruna C., and R. Gomperts, "Performance optimization of clustal w: Parallel clustal w, ht clustal, and multiclustal," *SGI ChemBio*.

[12] J. Cheetham, F. Dehne, S. Pitre, A. Rau-Chaplin, and P. J. Taillon, "Parallel clustal w for pc clusters," *Proceedings of International Conference on Computational Science and Its Applications (ICCSA)*, vol. 2668, pp. 300–309, 2003.

[13] K.B. Li, "Clustalw-mpi: Clustalw analysis using distributed and parallel computing," *Bioinformatics 19*, vol. 19, no. 12, pp. 1585–1586, 2003.

[14] J. Luo, I. Ahmad, M. Ahmed, and R. Paul, "Parallel multiple sequence alignment with dynamic scheduling," *Proceedings of the International Conference on Information Technology: Coding and Computing (ITCC'05)*, 2005.

[15] D. Nathan, R. Clemens, T. Oliver, B. Schmidt, and D. Maskell, "Multiple sequence alignment

on an fpga," *Proceedings of the Forth IEEE Int. Workshop on High Performance Computational Biology (HiCOMB 2005, IPDPS 2005)*, 2005.

[16] N. Saitou and M. Nei, "The neighbor-joining method: a new method for reconstructing phylogenetic trees," *Mol Biol Evol*, vol. 4, no. 4, pp. 406–425, 1987.

[17] R.C. Edgar, "Muscle: a multiple sequence alignment method with reduced time and space complexity," *BMC Bioinformatics. 2004*, vol. 5, 2004.

[18] http://bioinfo.pbi.nrc.ca, "Clustalw_smp ver. 0.99-9," 2002.

[19] http://www.intel.com, "Intel thread profiler," .