Austin Journal of Computational Biology and Bioinformatics

Open Access

Austin Publishing Group

Research Article

Graphics Card Processing: Acceleration of Multiple Sequence Alignment

Hanif MK* and Zimmermann KH
Institute of Computer Technology, Hamburg University of Technology, Germany

***Corresponding author:** Muhammad Kashif Hanif, Institute of Computer Technology, Hamburg University of Technology, 21071 Hamburg, Germany, Email: muhammad.hanif@tuhh.de

Received: September 16, 2014; **Accepted:** December 01, 2014; **Published:** December 05, 2014

Abstract

ClustalW is the most widely used heuristic method for multiple sequence alignment. It consists of three stages: distance matrix calculation, guide tree compilation, and greedy-fashion alignment. The high computational complexity demands methods to accelerate the algorithm. In this work, the efficient mapping of the progressive alignment stage onto graphics processing unit by using a combination of wavefront and matrix-matrix product techniques will be studied. The experimental results exhibit one order of magnitude speed-up over the serial version.

Keywords: Alignment; Progressive alignment; Graphics processor card; ClustalW; Performance

Introduction

Sequence alignment is the fundamental technique in molecular biology to compare sequences and to identify regions of similarity that are eventually consequences of structural, functional, or evolutionary relationships [1-4]. Sequence alignment is performed for all kinds of organic molecules, like DNA, RNA, or protein sequences. Multiple sequence alignment is the technique to align three or more sequences simultaneously. The aligned sequences are obtained by inserting gaps and have equal length. However, multiple sequence alignment is very time-consuming. For instance, optimal dynamic programming methods require $O(2^k n^k)$ steps to simultaneously align k sequences of length $O(n)$ [4].

A variety of heuristic methods have been developed to cope with multiple sequence alignment problems. The most widely accepted heuristic method for aligning multiple sequences is *progressive alignment* [5,6]. This method aligns more closely related sequences first and then gradually adds more divergent sequences [7]. The alignment accuracy can be improved by assessing the sequences according to their relatedness. A progressive alignment algorithm can handle a larger number of sequences in practical time scales. The most widely used progressive alignment programs are ClustalW [5,8, 9], T-Coffee [6,10], MAFFT [11,12], and MUSCLE [13,14].

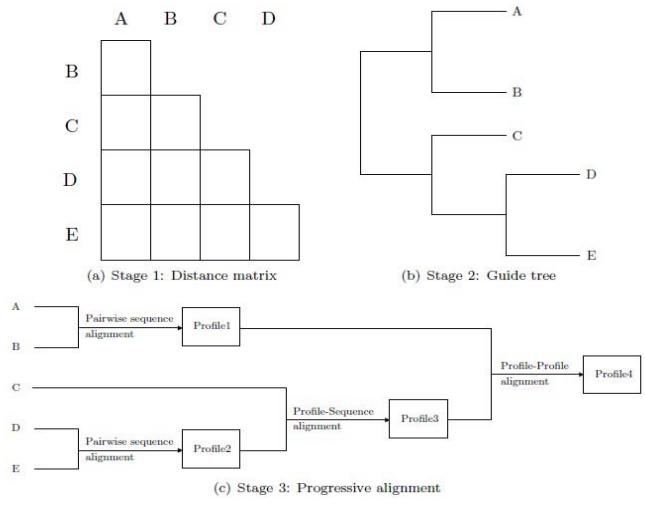


Figure 1: Stages of the ClustalW algorithm. The first stage computes the pairwise distances between the sequences. The guide tree is built in stage two using the distances. In stage three, the sequences are progressively aligned.

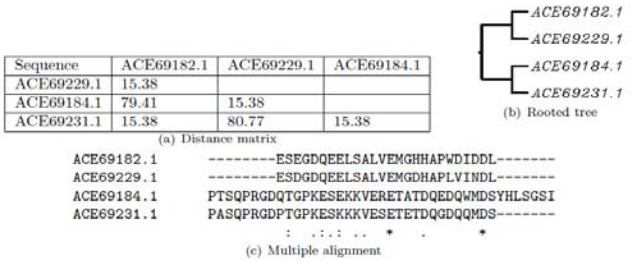
ClustalW is a typical progressive alignment algorithm making use of the policy "once a gap, always a gap", i.e., gaps introduced earlier in the alignment remain valid as new sequences are added [9,15]. It works in three stages (Figure 1). In the first stage, the distances between all pairs of sequences are calculated by pairwise sequence alignment. Pairwise sequence alignment can be calculated by the dynamic programming based method of Needleman-Wunsch [16] or one of its varieties like Smith-Waterman [17] or a fast heuristic method [9,18-20]. The scores of attained pairwise alignments are converted into distances which are input for the subsequent stage [9].

In the second stage, the distance matrix calculated in the first stage is used to build the guide tree which serves as a guide for the calculation of the overall multiple sequence alignment. This tree can be constructed by a heuristic phylogenetic method, like neighbour joining [21] or Unweighted Pair Group Method with Arithmetic mean (UPGMA) [22].

In the final stage, the sequences are progressively aligned using the guide tree. For this, the sequences correspond one-to-one with the leaves of the tree. Three cases can occur:

- An inner node (cherry) whose descendants are leaves is associated with the pairwise alignment of the sequences corresponding to these leaves.
- An inner node whose descendants are a leaf and an inner node is associated to the alignment given by the sequence and the multiple alignments. This can be achieved by profile-sequence alignment where the given multiple alignment is represented by a statistical representative called profile.

Citation: Hanif MK and Zimmermann KH. Graphics Card Processing: Acceleration of Multiple Sequence Alignment. Austin J Comput Biol Bioinform. 2014;1(2): 6.



Sequence	ACE69182.1	ACE69229.1	ACE69184.1
ACE69229.1	15.38		
ACE69184.1	79.41	15.38	
ACE69231.1	15.38	80.77	15.38

Figure 2: ClustalW based sequence alignment between the *tat* and *vpu* proteins from HIV 1 calculated from EMBL-EBI using the BLOSUM substitution matrix. The gap opening and the gap extension penalties for pairwise alignments are 10 and 0.1, respectively, and the initial gap penalty and the gap extension penalty for multiple alignments are 25 and 0.2, respectively.

- An inner node whose descendants are two inner nodes is associated to the alignment given by the corresponding multiple alignments. This can be attained by profile-profile alignment where the given multiple alignments are represented by statistical representatives.

The root of the tree corresponds to the overall multiple sequence alignment. The basic algorithm uses one weight matrix and fixed gap opening and extension penalties.

This approach, however, is not suitable for more divergent sequences. In this case, sequence weights are calculated from the guide tree. Closely related sequences have lower weights while the divergent ones have higher weights. Moreover, different substitution matrices are used at different alignment stages. New penalties are calculated based on the length and similarity of sequences, weight matrix, and gap positions [4,9]. An example using the *tat* and *vpu* proteins from HIV 1 (Human Immunodeficiency Virus) is shown in Figure 2. The complexity of the ClustalW algorithm is shown in Table 1 where n is the number of sequences and l is the average sequence length [20].

Many efforts have been made to accelerate the performance of the ClustalW algorithm. ClustalW-MPI [23], Ebedes et al. [24], and pCLUSTAL [25] use MPI to parallelize ClustalW on a cluster. ClustalW-MPI parallelized all three stages and achieved approximately 4.3 speed-up using 16 processors. Ebedes et al. demonstrated a speed-up of 5.5 by parallelizing the stages one and three. Similarly, Tan et al. [26] use MPI/Open MP for symmetric multiprocessors to parallelize the stages one and three. Mikhailov et al. [27] show a 10-fold speed-up by parallelizing all three stages with OpenMP on a shared-memory SGI machine. Aung et al. [28] employed a Field-Programmable Gate Array (FPGA) for acceleration of stage one. Oliver et al. [29] mapped stage one on FPGA and attained a speed-up between 45 and 50. MT-ClustalW [30] utilized pthreads to parallelize all three stages. GPU-ClustalW [31] parallelized the first stage on a GPU with OpenGL to obtain approximately 7 speed-up. MSA-CUDA [19] exploited the parallel architecture of the GPU by implementing all three stages and achieved a maximum average speed-up of approximately 37 for a small number of long sequences. Bassoy et al. [32] formulated a matrix-matrix product algorithm by separating the profile-sequence alignment algorithm into a data dependent and a data independent part to attain an order of magnitude speed-up on a GPU. However, they have ignored the time taken by executing the data dependent part on the CPU which is the reason for their huge speed-up given. Recently, Hanif and Zimmermann [33] described parallel algorithms for profile-profile alignment using matrix-matrix product and the wavefront approach attaining a 20-fold average speed-up for the wavefront approach. The results have shown that the matrix-matrix product and the wavefront methods are the most promising for profile-sequence alignment and profile-profile alignment, respectively.

Table 1: Complexity of the ClustalW algorithm by stage [20].

Stage	O(Time)
Distance matrix	$O(n^2 l^2)$
Guide tree	$O(n^3)$
Progressive alignment	$O(nl^2 + n^2 l)$
Total	$O(n^2 l^2 + n^3)$

A Graphics Processing Unit (GPU) is a highly parallel many-core streaming architecture which can execute hundreds of threads in a concurrent manner. The data parallel architecture of a GPU is particularly suitable to perform computation intensive tasks. GPUs offer orders of magnitude more computation power than CPUs and are becoming increasingly popular for general purpose computations to attain high speed-ups. A large set of problems in molecular dynamics, physics simulations, and scientific computing [34] have been tackled by mapping them onto a GPU. NVIDIA has introduced a GPU programming model called *Compute Unified Device Architecture* (CUDA) which enables the programmer to write C-like functions called kernels with some extensions that leverages programmers to efficiently use the graphics API. Each kernel is executed by a batch of parallel threads. CUDA provides three key abstractions: a hierarchy of thread groups, shared memories, and barrier synchronization [34].

In this paper, a combination of matrix-matrix product and wavefront methods will be used to parallelize the progressive alignment stage of ClustalW. The paper is organized as follows: Section 2 provides a method to accelerate the progressive alignment stage of ClustalW and section 3 evaluates its performance and compares it to the CPU implementation.

Materials and Methods

The performance of the ClustalW algorithm can be improved using the parallel architecture of the GPU. This particularly holds for the third stage, the alignment of the sequences using a guide tree.

First, a matrix-matrix product based approach to profile-sequence alignment will be introduced [35]. The technique of Bassoy et al. [32] is similar, but required additional memory and clock cycles.

Given a multiple alignment of length m by its profile P on the alphabet $\Sigma' = \Sigma' \cup \{-\}$ and a sequence x of length n, the score between a column p of the profile and a character $a \in \Sigma'$ is [4]

$$\sigma(p,a) = \sum_{b \in \Sigma'} \sigma(a,b) \cdot p_b. \qquad (1)$$

Profile-sequence alignment algorithm can be converted into a matrix-matrix product by separating the data dependent and independent parts. First, the data independent part calculates three scalar products. The diagonal entries of the forward table are stored in m × n matrix D. Two vectors hand vof lengths m and n are used to store the vertical and horizontal entries of the forward table. Second,

these values are used in the data dependent part for calculation of the forward table entries.

The profile column $\bar{p} = (0,\ldots,0,1)^T$ represents a column consisting of blanks having relative frequency of blank 1. Take the extended alphabet $\Sigma' = \{a_1,\ldots,a_l\}$, where a_l equals blank, and assign

$$\boldsymbol{w}_a = (\sigma(a,a_1),\ldots,\sigma(a,a_l))^T,\ a \in \Sigma'. \quad (2)$$

The vectors $\boldsymbol{h}$ and $\boldsymbol{v}$ can be computed as

$$h_j = \sigma(-_p, x_j) = \sum_b \sigma(x_j, b) \cdot -_p = -_p^T \cdot w_{x_j}, \quad (3)$$

$$v_i = \sigma(p_i, -) = \sum_b \sigma(-, b) \cdot p_{i,b} = p_i^T \cdot w_-. \quad (4)$$

The matrix $\boldsymbol{D}$ of size m × n is calculated as

$$D_{i,j} = \sigma(p_i, x_j) = \sum_b \sigma(x_j, b) \cdot p_{i,b} = p_i^T \cdot w_{x_j}, \quad (5)$$

The calculations of $\boldsymbol{h}$, $\boldsymbol{v}$, *and* $\boldsymbol{D}$ *can be written into a matrix-vector product. For this, take the* $l \times n$ *matrix* $\boldsymbol{W}$

$$W = (w_{x_1},\ldots,w_{x_n}).$$

The values of the vectors h and v and the matrix D are determined as

$$v = P^T \cdot w_-, \quad (6)$$

$$h = -_p^T \cdot W, \quad (7)$$

$$D_i = p_i^T \cdot W, \quad (8)$$

To calculate first column of the forward table, take the lower triangular $m \times m$ matrix B_m

$$B_m \cdot v = \begin{pmatrix} 1 & & & \\ 1 & 1 & & \\ \vdots & & \ddots & \\ 1 & 1 & \cdots & 1 \end{pmatrix} \left[\begin{pmatrix} p_1^T \\ \vdots \\ p_m^T \end{pmatrix} \cdot w_- \right]$$

$$= \begin{pmatrix} p_1^T \cdot w_- \\ p_1^T \cdot w_- + p_2^T \cdot w_- \\ \ldots \\ p_1^T \cdot w_- + p_2^T \cdot w_- + \ldots + p_m^T \cdot w_- \end{pmatrix}. \quad (9)$$

Similarly, the first row is calculated by having the $n \times n$ upper triangular matrix B_n

$$h \cdot B_n = \left[-_p^T \cdot (w_{x_1}\ w_{x_2} \ldots w_{x_n}) \right] \begin{pmatrix} 1 & 1 & \cdots & 1 \\ & 1 & \cdots & 1 \\ & & \ddots & \vdots \\ & & & 1 \end{pmatrix}$$

$$= \left(-_p^T \cdot w_{x_1} \cdots -_p^T \cdot w_{x_1} + \cdots + -_p^T \cdot w_{x_n} \right). \quad (10)$$

This gives the algorithm PROSEQALIGNMATVECPRODV2.

Algorithm 1: PROSEQALIGNMATVECPRODV2(***x***,***P***).

Require: sequence $\boldsymbol{x} = x_1 \ldots x_n$ and profile $\boldsymbol{P} = \boldsymbol{p}_1 \ldots \boldsymbol{p}_m$

1: $S_{0,0} \leftarrow 0$ {initialization}

2: $\boldsymbol{v} \leftarrow \boldsymbol{P}^T \cdot \boldsymbol{w}_-$

3: $h \leftarrow -_p^T \cdot W$

4: $\boldsymbol{S}_{*,0} \leftarrow \boldsymbol{B}_m \boldsymbol{v}$

5: $\boldsymbol{S}_{0,*} \leftarrow \boldsymbol{h}\boldsymbol{B}_n$

6: **for** $i \leftarrow 1$ to m **do** {calculation}

7: $\boldsymbol{D}_i \leftarrow \boldsymbol{p}^T{}_i \cdot \boldsymbol{W}$

8: **end for**

9: **for** $i \leftarrow 1$ to m **do** {maximization}

10: **for** $j \leftarrow 1$ to n **do**

11: $S_{i,j} \leftarrow max\{S_{i-1,j}+v_i, S_{i,j-1}+h_j, S_{i-1,j-1}+D_{ij}\}$

12: **end for**

13: **end for**

14: **return S**

The matrix $\boldsymbol{D}$ can be computed by matrix multiplication as

$$\boldsymbol{D} = \boldsymbol{P}^T.\ \boldsymbol{W}. \quad (11)$$

This resultant algorithm is PROSEQALIGNMATPRODV2.

Five versions of profile-sequence alignment algorithm on GPU have been considered.

Algorithm 2: PROSEQALIGNMATPRODV2(***x***,***P***).

Require: sequence $\boldsymbol{x} = \boldsymbol{x}_1 \ldots \boldsymbol{x}_n$ and profile $\boldsymbol{P} = \boldsymbol{p}_1 \ldots \boldsymbol{p}_m$

1. $S_{0,0} \leftarrow 0$ {initialization}

2. $v \leftarrow \boldsymbol{P}^T \boldsymbol{.w}$-

3. $\boldsymbol{h} \leftarrow -_p^T \cdot \boldsymbol{w}$

4. $\boldsymbol{S}_{*,0} \leftarrow \boldsymbol{B}_m \boldsymbol{v}$

5. $S_{0,*} \leftarrow hB_n$

6. $\boldsymbol{D} \leftarrow P^T.W$

7. **for** $\boldsymbol{i} \leftarrow 1$ to m **do** {maximization}

8. **for** $\boldsymbol{j} \leftarrow 1$ to $\boldsymbol{n}$ **do**

9. $S_{i,j} \leftarrow max\{S_{i-1,j}+v_i, S_{i,j-1}+h_j, S_{i-1,j-1}+D_{ij}\}$

10. **end for**

11. **end for**

12. **return** $\boldsymbol{S}$

- MatVecProd V1: Matrix-vector product implementation using *cublasSgemv* [32].
- MatVecProd V2: Matrix-vector product implementation using *cublasSgemv*.

- MatProd V1: Matrix-vector product implementation using *cublasSgemm* [32].

- MatProd V2: Matrix-vector product implementation using *cublasSgemm*.

- SMWavefront 256: Wavefront approach using shared memory having block size 256 [35].

For MatVecProd V1 and MatProd V1, results are taken up to sequence length 6,000. A sequence length of 10,000 requires approximately 1145 MB to store matrices which is well beyond the capacity of available global memory (1024 MB). A maximum speed-up factor of approximately 28 is attained when compared with optimized Intel CPU implementation (Figure 3). Parallel versions of profile-profile alignment are given in [33]. The results show that a mixture of wavefront and matrix-matrix product methods can be useful for the parallelization of the progressive alignment stage.

The approach adopted is similar to that in [19]. First, the intermediate nodes of the guide tree are labeled by post-order traversal. Two vectors are used to maintain the right child and left child of the nodes. One flag vector is required to keep track whether the node has been aligned. The flag for the leaf nodes is set to 1 when the alignment is not required. The left and right children indices are 0 for the leaf nodes.

The flag vector is checked to identify the nodes of the guide tree to be aligned. An alignment at an intermediate node can only be performed if the right and left children have been aligned. In first phase, the leaf nodes are aligned using pairwise sequence alignment since the left and right children are assumed to be aligned as indicated in the flag vector. In next phase, the flag vector is again checked to find potential candidates for subsequent alignment. This process continues until the overall multiple sequence alignment is obtained and the flag vector contains 1 for each node.

The frequency based profiles are constructed for the intermediate nodes. A profile and a sequence are aligned by the matrix-matrix product while two profiles are aligned using the wavefront method on the GPU. The traceback is performed on the CPU to find the alignment by adding gaps in the aligned sequences.

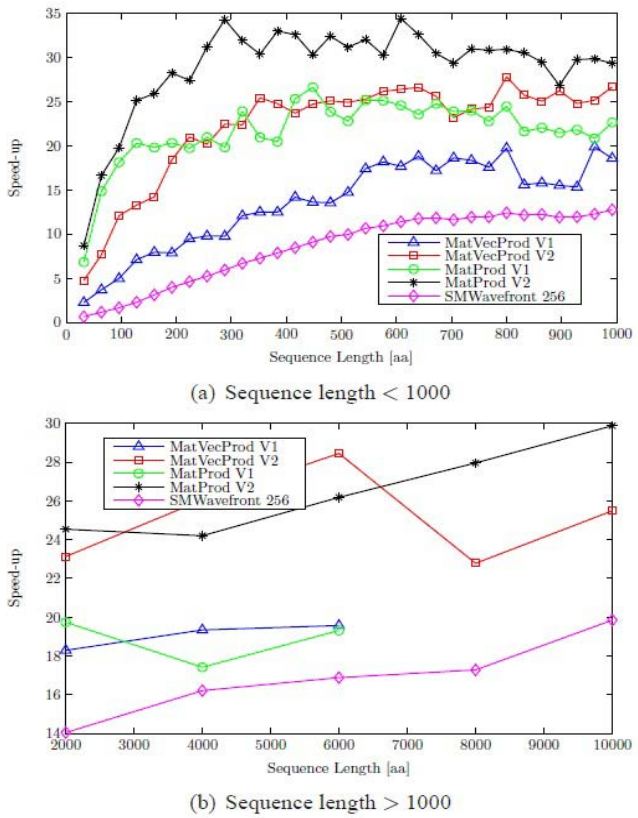


Figure 3: Speed-ups of profile-sequence alignment algorithms.

Results and Discussion

The ClustalW progressive alignment stage has been implemented on an Intel Core 2 Duo 6600 CPU (2.40 GHz) running openSUSE 11.4 linux distribution and CUDA version 4.0 on an NVIDIA GeForce GTX 560 Ti graphics card. The tests have been conducted using a serial gcc compiler (version 4.4.1) and an NVIDIA nvcc compiler. The performance of the parallel progressive alignment stage has been measured by the speed-up. The dataset for the implementation of the ClustalW algorithm is similar to that in [19]. The protein sequence dataset consists of the HIV dataset available at the NCBI database. The dataset has been divided into several subsets:

- 400 sequences of average length 856 and 1000 sequences of average length 858;

- 2000 sequences of average length 266 and 4000 sequences of average length 247;

- 4000 sequences of average length 57 and 8000 sequences of average length 73.

The execution times have been averaged over ten runs for each data subset. The times for memory allocation and data transfer to or from the GPU have been neglected. The input to the progressive alignment stage is a guide tree which has been generated by the ClustalW program from the EMBL-EBI website.

The speed-up for the progressive alignment stage is illustrated in Figure 4. Note that profile creation and traceback have been performed on the CPU and have not been neglected. The data subsets given by longer sequences have achieved a speed-up of one order of magnitude

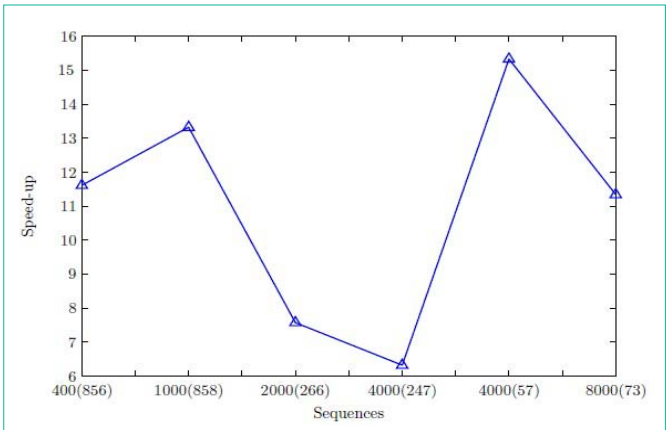


Figure 4: Speed-up for progressive alignment stage of the ClustalW.

since the matrices to be multiplied utilize the hardware resources rather efficiently. The data subsets given by shorter sequences show a similar behavior due to the possibility to process multiple sequences in each pass. The computation-to-communication ratio of wavefront approach for sequences and profiles [33] of average length is low which impacts the speed-up exhibited by the data subsets of average length sequences. The maximum speed-up attained is much better than the speed-up of approximately 6 exhibited by progressive alignment stage of MSA-CUDA [19].

Conclusion

This paper has provided a parallel algorithm for the progressive alignment stage of the ClustalW algorithm onto the GPU using a mixture of algorithms: matrix-matrix product for profile-sequence alignment and wavefront for profile-profile alignment. The results have shown a performance increase of more than one order of magnitude for several data sets considered.

Acknowledgement

The authors would like to thank Yongchao Liu for providing the dataset. The work of the first author has been sponsored by DAAD and Higher Education Commission of Pakistan.

References

1. Durbin R, Eddy SR, Krogh A, Mitchison GJ. Biological sequence analysis: probabilistic models of proteins and nucleic acids. Cambridge University Press. 1998.
2. Gusfield D. Algorithms on strings, trees, and sequences: computer science and computational biology. New York: Cambridge University Press. 1997.
3. Waterman MS. Introduction to computational biology: maps, sequences, and genomes: interdisciplinary statistics. CRC Press. 1995.
4. Zimmermann KH. An introduction to protein informatics. Boston: Kluwer Academic Publishers. 2003.
5. Higgins DG, Sharp PM. CLUSTAL: a package for performing multiple sequence alignment on a microcomputer. Gene. 1988; 73: 237-244.
6. Notredame C, Higgins DG, Heringa J. T-Coffee: a novel method for fast and accurate multiple sequence alignment. Journal of Molecular Biology. 2000; 302: 205–217.
7. Feng DF, Doolittle RF. Progressive sequence alignment as a prerequisite to correct phylogenetic trees. J Mol Evol. 1987; 25: 351-360.
8. Chenna R, Sugawara H, Koike T, Lopez R, Gibson TJ, Higgins DG, et al. Multiple sequence alignment with the Clustal series of programs. Nucleic Acids Res. 2003; 31: 3497-3500.
9. Thompson J, Higgins D, Gibson T. CLUSTAL W: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice. Nucleic Acids Research. 1994; 22: 4673–4680.
10. Magis C, Taly JF, Bussotti G, Chang JM, Di Tommaso P, Erb I, et al. T-Coffee: tree-based consistency objective function for alignment evaluation. Methods in Molecular Biology. 2014; 1079: 117–129.
11. Katoh K, Misawa K, Kuma K, Miyata T. MAFFT: a novel method for rapid multiple sequence alignment based on fast Fourier transform. Nucleic Acids Res. 2002; 30: 3059-3066.
12. Katoh K, Standley DM. MAFFT multiple sequence alignment software version 7: improvements in performance and usability. Mol Biol Evol. 2013; 30: 772-780.
13. Edgar RC. MUSCLE: multiple sequence alignment with high accuracy and high throughput. Nucleic Acids Res. 2004; 32: 1792-1797.
14. Edgar RC. MUSCLE: a multiple sequence alignment method with reduced time and space complexity. BMC Bioinformatics. 2004; 5: 113.
15. Higgins DG, Thompson JD, Gibson TJ. Using CLUSTAL for multiple sequence alignments. Methods Enzymol. 1996; 266: 383-402.
16. Needleman SB, Wunsch CD. A general method applicable to the search for similarities in the amino acid sequence of two proteins. J Mol Biol. 1970; 48: 443-453.
17. Smith TF, Waterman MS. Identification of common molecular subsequences. J Mol Biol. 1981; 147: 195-197.
18. Bashford D, Chothia C, Lesk AM. Determinants of a protein fold. Unique features of the globin amino acid sequences. J Mol Biol. 1987; 196: 199-216.
19. Liu Y, Schmidt B, Maskell DL. MSA-CUDA: Multiple sequence alignment on graphics processing units with CUDA. Proceedings of 20th IEEE International Conference on Application-Specific Systems, Architectures and Processors, ASAP 2009. Boston, MA, USA. IEEE. 2009; 121–128.
20. Liu Y, Schmidt B, Maskell DL. Parallel reconstruction of neighbor-joining trees for large multiple sequence alignments using CUDA. Proceedings of 23rd IEEE International Sym-posium on Parallel and Distributed Processing, IPDPS 2009. Rome, Italy. Washington: IEEE Computer Society. 2009; 1–8.
21. Saitou N, Nei M. The neighbor-joining method: a new method for reconstructing phylogenetic trees. Mol Biol Evol. 1987; 4: 406-425.
22. Larkin MA, Blackshields G, Brown NP, Chenna R, McGettigan PA, McWilliam H, et al. Clustal W and Clustal X version 2.0. Bioinformatics. 2007; 23: 2947-2948.
23. Li KB. ClustalW-MPI: ClustalW analysis using distributed and parallel computing. Bioinformatics. 2003; 19: 1585-1586.
24. Ebedes J, Datta A. Multiple sequence alignment in parallel on a workstation cluster. Bioinformatics. 2004; 20: 1193-1195.
25. Cheetham J, Dehne F, Pitre S, Rau-Chaplin A, Taillon PJ. Parallel Clustal W for PC clusters. Proceedings of the International Conference on Computational Science and Its Applications: Part II, ICCSA 2003. Montreal, Canada. Lecture Notes in Computer Science. Berlin: Springer. 2003; 2668: 300–309.
26. Tan G, Feng S, Sun N. Parallel Multiple Sequences Alignment in SMP Cluster. Proceedings of the Eighth International Conference on High-Performance Computing in Asia-Pacific Region, HPCASIA'05; 2005. Beijing, China. Washington: IEEE Computer Society. 2005; 425–431.
27. Mikhailov D, Cofer H, Gomperts R. Performance Optimization of Clustal W: Parallel Clustal W, HT Clustal, and MULTICLUSTAL. SGI ChemBio. 2001.
28. Aung YL, Maskell DL, Oliver TF, Schmidt B, Bong W. C-based design methodology for FPGA implementation of ClustalW MSA. Proceedings of the 2nd IAPR International Conference on Pattern Recognition in Bioinformatics; 2007. Lecture Notes in Computer Science. Berlin: Springer. 2007; 4774: 11–18.
29. Oliver T1, Schmidt B, Nathan D, Clemens R, Maskell D. Using reconfigurable hardware to accelerate multiple sequence alignment with ClustalW. Bioinformatics. 2005; 21: 3431-3432.
30. Chaichoompu K, Kittitornkun S, Tongsima S. MT-ClustalW: multithreading multiple sequence alignment. Proceedings of the 20th International Conference on Parallel and Distributed Processing, IPDPS'06; 2006. Rhodes Isl and, Greece. Washington: IEEE Computer Society. 2006; 254–254.
31. Liu W, Schmidt B, Voss G, Muller-Wittig W. GPU-ClustalW: using graphics hardware to accelerate multiple sequence alignment. Proceedings of the 13th International Conference on High Performance Computing, HiPC'06; 2006. Bangalore, India. Lecture Notes in Computer Science. Berlin: Springer. 2006; 4297: 363–374.
32. Bassoy CS, Torgasin S, Yang M, Zimmermann KH. Accelerating Scalar-Product Based Sequence Alignment using Graphics Processor Units. Signal Processing Systems. 2010; 61: 117–125.
33. Hanif MK, Zimmermann KH. Graphics card processing: accelerating profile-profile alignment. Central European Journal of Computer Science. 2012; 2: 367–388.

34. NVIDIA Corporation. NVIDIA Corporation, editor. CUDA C Programming Guide Version 4.0. NVIDIA Corporation. 2011.

35. Hanif MK. Mapping dynamic programming algorithms on graphics processing units. PhD Thesis, Hamburg University of Technology. 2014.

Austin J Comput Biol Bioinform - Volume 1 Issue 2 - 2014
ISSN : 2379-7967 | www.austinpublishinggroup.com

Citation: Hanif MK and Zimmermann KH. Graphics Card Processing: Acceleration of Multiple Sequence Alignment. Austin J Comput Biol Bioinform. 2014;1(2): 6.