Journal of Theoretical and Applied Information Technology
20th June 2014. Vol. 64 No.2

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

PARALLELIZED HIERARCHICAL EXPECTED MATCHING PROBABILITY FOR MULTIPLE SEQUENCE ALIGNMENT

[1]KEDHAR M, [2] DR. M. RAJASEKHARA BABU, [3]MAYANK G, [4] ABHINIVESH M

[1]Student, Department of Computer Science ,VIT University, Vellore
[2]Associate Professor, Department of Computer Science ,VIT University, Vellore
[3]Student, Department of Computer Science ,VIT University, Vellore
[4]Student, Department of Computer Science ,VIT University, Vellore
[1]kedhar1992@gmail.com, [2]mrajasekharababu@vit.ac.in,[3]mayankgarg121@gmail.com
[4] abhiniveshmahto@gmail.com

ABSTRACT

Sequence alignment of two or more than two biological sequences such as protein, DNA (Deoxyribonucleic acid) or RNA (Ribonucleic acid) is called MSA (Multiple Sequence Alignment). Sequence homology can be inferred from the resulting MSA. Existing System uses dynamic programming technique which suffers from exponential growth of time as the sequence grows. A Hierarchical Expected Matching Probability (HEP) Matrix scoring technique improves accuracy but it consumes more time. This paper presents an implementation of HEP on GPU's to speed up the Multiple Sequence Alignment (MSA). The experiment results shown that there is 25% accuracy of MSA and also 4% of speedup.

Keywords: *Multiple Sequence Alignment, Hierarchical Expected Matching Probability, Protein Residue Conservation, Scoring Matrix, Protein And DNA Sequence, Heterogeneous Environment*

1. INTRODUCTION

Sequence alignment of two or more than two biological sequences is called MSA (Multiple Sequence Alignment) [1]. MSA is one of most quickest and faster approach in aligning the sequences [33]. It is used to assess the future genetic database based on the existing sequences [1], [33]. It is a computational procedure for aligning the sequences [33]. The accuracy of MSA is essential because a lot of Bioinformatics techniques and procedures are addict MSA result [3].If the current MSA runs on sequential programming language [4], [5]. It suffers from two problems. (1).The user cannot break the work into several tasks. (2) The overall computation time is increased. Most probably it relies on a single core processor. The increase in the number of cores has laid a foundation for parallel environment that address several problems [36]. It got a chance to split the work into several tasks [36]. Each task is performed by the individual core that reduces the execution time [34]. The further increase in cores it laid the foundation for Graphical Processing Unit (GPU) that contains thousands of cores which raises the power of computation [2][31] . It reduces the overall time complexity [31]. The intent of this paper is to parallelization of a Multiple Sequence Alignment (MSA) with Hierarchical Expected Matching Probability (HEP) on Heterogeneous environment to speed up the process of alignment [37].

.

2. RELATED WORK

MSA is a popular approach that is used to understand the behavior of a newly found gene with the existing sequences [3], [28]. Sequence alignment algorithms are used to find relationship among DNA sequences [29]. These alignment algorithms are broadly classified into two categories Pair wise and MSA algorithms [29]. The main functionality of Pair wise sequence alignment is to identify the similar pattern in two sequences .In Multiple Sequence alignment algorithm the functionality is to identify the similar pattern of two or more sequences of DNA[29],[36]. The definition of Multiple Sequence Alignment is as follows.

$$A := \begin{cases} A_1 = (a_{11}, a_{12}, \ldots, a_{1n1}) \\ A_2 = (a_{21}, a_{22}, \ldots, a_{2n2}) \\ \vdots \\ A_r = (a_{r1}, a_{r2}, \ldots, a_{rnr}) \end{cases} \quad (1)$$

Journal of Theoretical and Applied Information Technology
20th June 2014. Vol. 64 No.2

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

The resulting aligned sequences will be in the form of

$$A := \begin{cases} A_1^* = (a_{11}^*, a_{12}^*, \ldots, a_{1L}^*) \\ A_2^* = (a_{21}^*, a_{22}^*, \ldots, a_{2L}^*) \\ \vdots \\ A_r^* = (a_{r1}^*, a_{r2,\ldots\ldots,}^*, a_{rL}^*) \end{cases} \quad (2)$$

Currently MSA are implemented with the help of scoring methods such as sum of pair method, Valdar's and Trident method, Smith's prima method and many more.

2.1 Sum Of Pair Method

The Sum of Pairs (SP) scoring method is as follows Given A set of N aligned sequences each of length L in the form of a L x N MSA alignment matrix M, A substitution matrix that gives the score s(x, y) for aligning two characters x, y[29],[30] .

Scoring Formula

$$S(x) = \sum_{i}^{n} \sum_{j \neq i}^{n} S(s_i(x), s_j(x)), \quad (3)$$

Where

S_i (x) = amino acid at column x in the i^{th} sequence (a).

S_j (x) = amino acid at column x in the j^{th} sequence (b).

$S_i(a)S_j(b)$ = score aligning two amino acids a and b.

The disadvantage of the sum of pair method is their inability to use some relevant information about the local variation region that appears across multiple sub trees.

2.2 VALDAR'S AND TRIDENT METHOD

The VALDAR'S and Trident Method scoring methods is as follows .It resolves the limitations of existing methods by proposing a conversation score. The conversation score formula [29] is mentioned in figure (4).

$$s(x) = \lambda \sum_{i}^{n} \sum_{j>i}^{n} w_i w_j M(S_i(x) S_j(x)) \quad (4)$$

Where n is the sequence length,

λ scale to range[0,1]

$$\lambda = 1 / \sum_{i}^{n} \sum_{j>1}^{n} w_i w_j \quad (5)$$

w_i weight of sequence S_i

w_j weight of sequence S_j

The disadvantage of this method is inconsistency in calculating the sum of score method. Trident has generalized the above formula in calculating the score matrix [28].

2.3 SMITH PRIMA METHOD

Smith's Pima is an *ad-hoc* scoring method that was developed to measure amino acids assembly quality by pattern induced multiple sequence alignment technique [27]. The advantage of smith prima method is very reliable and robust which describes to combine Amino acid class having substitution scoring matrix. The above said scoring methods are used to find an optimal multiple sequences. The Hierarchical Expected Matching Probability (HEP) is developed based on the above specified methods [29]. With the help of above three specified methods this paper tried to develop HEP method that optimizes the overall complexity of the previous specified methods. The developed HEP method runs on different environments such as heterogeneous environments under various samples of input [37]. This paper evaluates the total time required to execute those sequences in sequentially first [29]. It tried to implement in CPU environment and compare both the results in terms of time complexity. Later on our work has been extended to implement the HEP method in GPU environment also that provides better enhancement in terms of performance of time and power issues [37].

To overcome the disadvantage of the sequential programming languages the developers used microprocessors [10]. The microprocessor are based on a single processing unit that improves the performance and cost reduction [8],[9],[10]. The design methodology in microprocessor is having multiple processing units called cores [7], [11]. This new interest in parallel program development has been called the "concurrency revolution" [12].The two main methodologies that are used to parallelize sequential applications are Auto parallelization and parallel programming [13], [14]. The main

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

working principle of the parallel programming languages is to divide the give problem into the set of task [13]. Each task is carried out by the individual core parallel so that the overall computational complexity is reduced[16]. To address the parallel programming we use OPENMP for shared memory and MPI for distributed memory [7],[15],[16]. The OPENMP multithreading interface is especially designed for high performance computing [17], [18]. With the above said specifications we can improve the performance when compared with the sequential programming. The drawbacks are more heat dissipation and power consumption.

The further advancement in increasing the number of cores, A new paradigm in the programming languages has paved a powerful tool in the present generation [39]. To overcome the problems of parallel programming that is heat dissipation and power Consumption is CUDA [36]. The developers of the CUDA parallel programming model is carried out by NVIDIA [23], [24], [25]. It is started in the year 2006. The major goal of this CUDA programming languages is to achieve high performance and optimizes the time and space complexity with the previous existing methodologies [39]. Computing the special consideration with the tools available which named as CUDA [20], [21] tool for implementing a parallel computing environment in the Graphical processing unit(GPU). Due to several advantages with low cost option to achieve very powerful and efficient power gains over CPU approaches[19] and Also we get a doubled or tripled folded speedup of GPU compared to a general CPU is already obtained[20],[21],[22]. The programming of CUDA is scalable parallel programming which together includes barrier synchronization, shared memories and grouping of threads parallel. The rest of the paper is organized as follows in section 3 this paper explains proposed HEP method and in section 4 this paper explains the HEP method in CPU using OPENMP programming concepts and in GPU using CUDA Programming languages[29]. In section 5 this paper compares results of HEP method in both the environments and in section 6 the paper explains conclusion.

3. DESIGN METHODOLOGIES:

Multiple sequence alignment with HEP scoring technique gives better performance in sequential CPU implementation for alignment of sequences. This paper suggest MSA with HEP in parallel with OPENMP and CUDA. OPENMP supports Multiple Instruction and Multiple Data (MIMD) and Compute Unified Device Architecture (CUDA)[20],[21] .Support Single Instruction and Multiple Data(SIMD) but they provides different level of parallelism. This paper s runs the MSA with HEP scoring technique both in OPENMP and CUDA and compare the results in terms of performance and efficient for MSA[17],[40]. This paper developed a metric (Hierarchical Expected Matching Probability (HEP)) that measures the chance of residual mutations. It also improves the biological correctness for aligning the MSA results [32]. It is a reliable consistent and powerful for aligning the sequences rather than other grading metrics[29]. The probability of getting a residue from one residue to another residue is very less. Among given sequences the chance of matching from sequence in to column is k-product. It defines the multiple times that seems residue at one particular location. It approaches that the probability of k-product is zero. It shows that event may occur as k-step is increasing[29][30]. It grows conversely with comparison to matching probability function. In order to stop some unrealistic alignments this scoring function should penalize some gap insertions. Hence MSA algorithm optimizes the residue columns. This paper chooses a flexible scoring metric called HEP. It finally quantifies the residue matching conversation that operates with random matching probability function.

3.1 WORK FLOW OF OPENMP

An OPENMP is an API for writing Multithreaded Applications[41]. It is a set of compiler directives and library routines for writing parallel applications[42]. It shares the address by communicating the threads variables. The Block diagram of OPENMP programming model is shown in the Fig 1. If follows fork-join parallelism [1]. The parallelism is introduced on sequential programming code to achieve the better performances [42]. To implement parallelism on sequential codes one should use OMP directives. To implement the proposed HEP method the workflow is shown in the FIG1. The workflow addresses that the division of the core into parallel regions[43]. Each parallel region is designated to a

Journal of Theoretical and Applied Information Technology
20th June 2014. Vol. 64 No.2

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

particular task. Each task is carried out by an individual thread parallel so that overall execution time is reduced and optimizes the performance(1). In the proposed HEP we are having four modules namely MATRIX, SCORING, TRACEBACK and printing the ALIGNED sequences. Each module specified in HEP are carried out by an individual thread parallel(1). When each thread completes its execution it calculates the overall execution time and speedup the process of alignment.

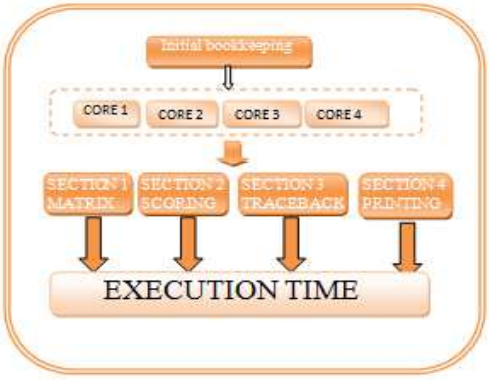

Fig1: OPENMPIMPLEMENTATIONWORKFLOW

3.2 WORK FLOW OF CUDA:

CUDA (Compute Unified Device Architecture) is parallel computing platform created by NVIDIA corporation[38]. It is designed for graphical processing unit (GPUs) to enhance better performance. It overcomes the problem of MPI such as scaling ability due to serialization and synchronization phases that increases with core count [46]. It provides an extension to c ANSI to do heterogeneous computing. The Fig 2 shows the CUDA implementation workflow. The flow of the data in CUDA is classified into two categories. The first flow of data represents the flow among the multithreads. The second flow represents the flow within the GPU [45]. These multithreaded programs are represented as blocks. Each block is designed for a particular task and work independent of each block. Based on the number of cores available on the GPU the blocks are classified according to it. The four blocks that are represented in multithreaded program namely block0, block1, block2, block3. If the number of cores are two in GPU then each block can execute in one core as we are having four blocks, two blocks can execute simultaneously on each core. Similarly if we implemented in 4-core GPU architecture all the blocks can be executed simultaneously. The CUDA tool used to perform parallel sparse matrix computations.

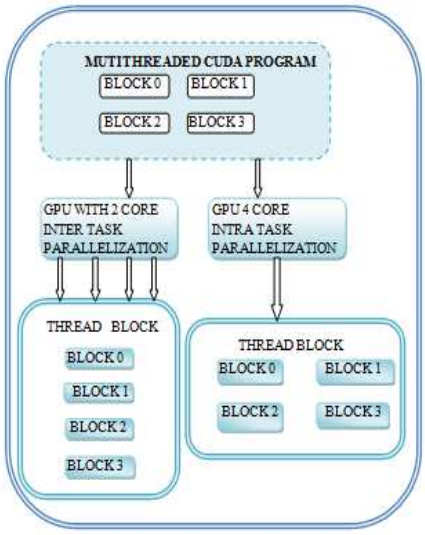

Fig2 : Cuda Implementation Workflow

3.3 GPU COMPUTATIONAL MODEL

The FIG.3 shows the computational model of GPU. The configuration of GPU systems are purely based on the hardware of the system. Most of the GPU systems come with Tesla C2070 and Single Instruction Multiple data (SIMD) that are organized into a Symmetric Multiprocessor. In this all the threads are grouped into blocks and blocks are organized into Grid. Each block is assigned serially to the symmetric multiprocessor and each block are divided into SIMD groups called WRAPS [45]. Computations on the GPU are demonstrated in the code as explicit kernels. CUDA code is broadly classified into host code and device code. The host code executes on (CPU) thread and device code that executes in (GPU) threads [39].

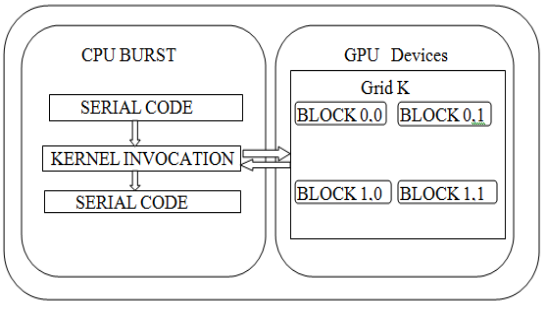

Fig3:Heterogenous Computing Model

4. IMPLEMENTATION

This paper implements the Multiple Sequence Alignment using HEP method. First this paper implements sequentially on CPU and calculates the execution time. The specifications

Journal of Theoretical and Applied Information Technology
20th June 2014. Vol. 64 No.2

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

that are required to run the sequential code parallel on CPU is OPENMP and GPU is CUDA.

4.1 Implementation Of Openmp

The method of HEP on MSA can be done using OPENMP. The parallelism in OPENMP is done using shared memory. This method consists of four parts. The first part is generation of matrix which is based on the input sequences. The second part is scoring ,it is computed by HEP scoring method for aligning of sequences. The third part is trace back that performs diagonal adjustment of the matrix. The fourth part specifies the printing of the aligned sequences with optimal sequence length. All these parts are executed simultaneously based on the number of cores available in the system.

4.2 Implementation Of Cuda

The implementation of HEP on MSA can be done in the heterogonous computing using CUDA. In this CUDA the code is classified into two parts such as host code and device code. All the specified modules such as matrix, scoring, trace back are written in device mode. The host mode issues a kernel call to the GPU after all the functions are performed again the GPU sends the results to the CPU and print the aligned sequences.

5. RESULTANALYSIS

This paper compares the results of the proposed HEP method for aligning of multiple sequences on different platforms for various length of sequences and represented in the forms of Graphs. The FIG 4 compares the execution time and length of the sequence in sequential and parallel two core machine. From the graph it is inferred that as the sequence grows the execution time is decreases in a parallel two core machine when compared to the sequential execution time. The FIG 5 demonstrates that as the number of cores increases execution time decreases when compared to the sequential time. The FIG 6 specifies further decrease in execution time when implements in a six-core. The FIG 7 specifies the further decrease in execution time when we implement in an eight core processor. The FIG 8 Specifies the increase in speedup as the number of cores increases for a particular length of a sequence. The FIG 9specifies the inter and intra task parallelism on GPU. The FIG 10 specifies the comparison of OPENMP and CUDA in terms of speedup for various length of sequence.

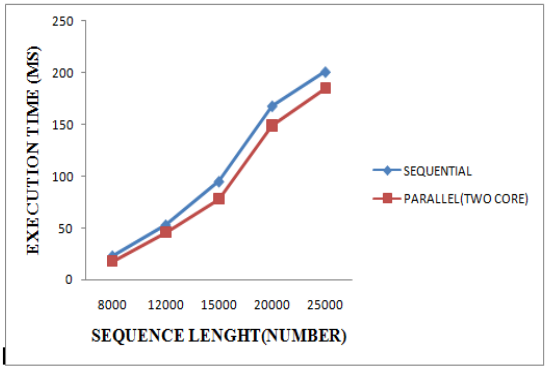

Fig 4: Two Core Vs Sequential

The fig 4 demonstrates that blue line specifies sequential time and the red line specifies the parallel time. As we inferred from the graph for given input sequence the blue line increase in time as the sequence grows when compared to the red line.

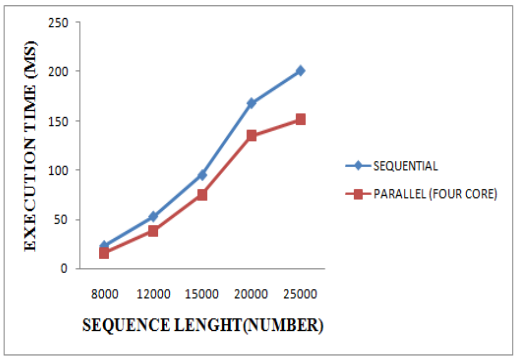

Fig 5: Four Core Vs Sequential

The FIG 5 demonstrates that as the number of cores increases execution time decreases when compared to the sequential time

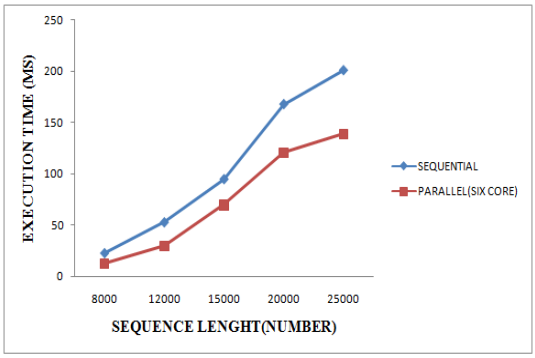

Fig 6: Six Core Vs Sequential

Journal of Theoretical and Applied Information Technology
20th June 2014. Vol. 64 No.2

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

The FIG 6 demonstrates that as the number of cores increases execution time decreases when compared to the sequential time

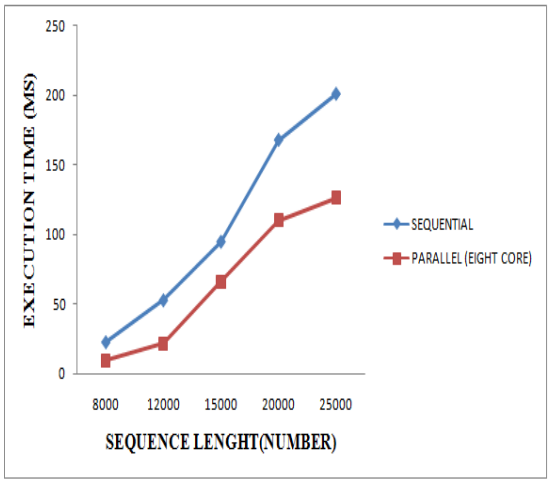

Fig 7: EIGHT CORE VS SEQUENTIAL

The FIG 7 demonstrates that as the number of cores increases execution time decreases when compared to the sequential time

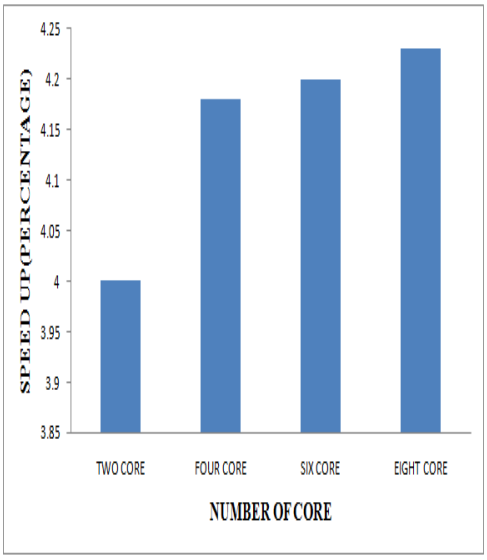

Fig 8: No Of Cores Vs Speed Up

The fig 8 demonstrates the speedup for aligning of the sequences. From the graph it is inferred that as the number of cores increases there is increase in speedup for aligning of the sequences.

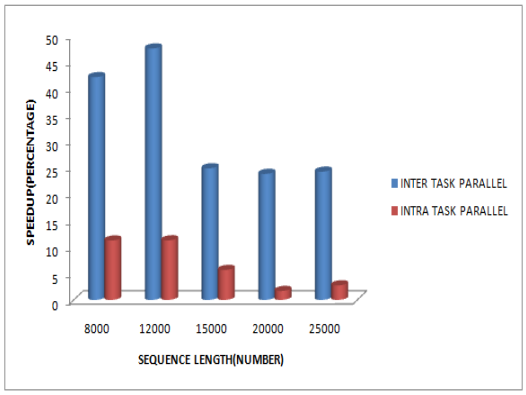

Fig 9: Sequence Length Vs Speedup On Gpu

The fig 9 specifies how the parallelism take place in CUDA both inter and intra task parallelism. From the graph it is inferred that inter task parallelism speedup is more when compared to the intra task parallelism.

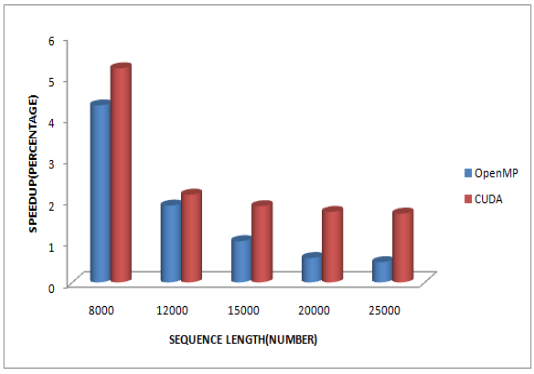

Fig 10: Comparision Of Execution Time On Openmp And Cuda

The fig 10 specifies the speedup comparison between OPENMP and CUDA. From the graph it is inferred that speedup for aligning of sequences is more in CUDA when compared to the OPENMP

6. CONCLUSION

This paper explains the parallelization of Multiple Sequence Alignment for aligning of sequences. From the results it can be inferred that as the number of core increases there is a decrease in execution time as the length of the sequence grows. This paper also says that there is a increase in speedup for aligning of the sequences as the number of core increases. From the result analysis it can be inferred that there is a increase in speedup and decrease in execution time when it is implemented in a heterogeneous computing.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

REFRENCES:

[1] hindawi.com/isrn/biomathematics/2013/615630/.

[2] Peter N. Glaskowsky. NVIDIAs Fermi: The First Complete GPU Computing Architecture (White Paper). Nvidia Corporation, September 2009.

[3] L.Wang and T. Jiang, "On the complexity of multiple sequence alignment," *J. Compum. Biol.*, vol. 1, no. 4, pp. 337–348, 1994.

[4] WIRTH, N.: 'Program development by stepwise refinement', Commun. ACM, 1971, 14, pp. 221-227.

[5] ELSHOFF, J.L., and MARCOTTY, M.: 'Improving computer program readability to aid modification', Commun.ACM, 1982, 25, pp. 512-521.

[6] ZPL: A Machine Independent Programming Language for Parallel Computers IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, VOL. 26, NO. 3, MARCH 2000

[7] A survey of parallel programming models and tools in the multi and many-core era IEEE transactions on parallel and distributed systems, vol. 23, no. 8, august 2012

[8] P.B. Hansen, Studies in Computational Science: Parallel Programming Paradigms. Prentice-Hall, 1995.

[9] K.M. Chandy and J. Misra, Parallel Program Design: A Foundation. Addison-Wesley, 1988.

[10] OPENMP, "API Specification for Parallel Programming," http://openmp.org/wp/openmp-specifications, Oct. 2011.

[11] W. Hwu, K. Keutzer, and T.G. Mattson, "The concurrency challenge," IEEE Design and Test of Computers, vol. 25, no. 4, pp. 312-320, July 2008

[12] H. Sutter and J. Larus, "Software and the Concurrency Revolution,"ACM Queue, vol. 3, no. 7, pp. 54-62, 2005.

[13] M.J. Quinn, Parallel Computing: Theory and Practice. McGraw-Hill,1994.

[14] P.B. Hansen, Studies in Computational Science: Parallel Programming Paradigms. Prentice-Hall, 1995.

[15] T.G. Mattson, B.A. Sanders, and B. Massingill, Patterns for Parallel Programming. Addison-Wesley Professional, 2005.

[16] V. Kumar, A. Grama, A. Gupta, and G. Karypis, Introduction to Parallel Computing: Design and Analysis of Algorithms. Benjamin/ Cummings Publishing Company, 1994.

[17] OPENMP, "API Specification for Parallel Programming,"http://openmp.org/wp/openmp-specifications, Oct. 2011.

[18] K. Kedia, "Hybrid Programming with OpenMP and MPI," Technical Report 18.337 J, Massachusetts Inst. of Technology, May 2009.

[19] Y. Liu, B. Schmidt, and D.L. Maskell, "MSA-CUDA: Multiple Sequence Alignment on Graphics Processing Units With CUDA," Proc. IEEE Int'l Conf. Application-Specific System, Architecture and Processors (ASAP), pp. 121-128, 2009.

[20]NVIDIA Corporation, NVIDIA CUDA Programming Guide, Version 3.1.1, July 2010.

[21]NVIDIA Corporation, NVIDIA CUDA Programming Guide, Version 2.3.1, Aug. 2009.

[22]NVIDIA.CUDAzone,http://www.nvidia.com/object/cuda_home.html, 2009.

[23]NVIDIADeveloperZone,http://developer.nvidia.com, Oct. 2011.

[24] NVIDIA Company. Nvidia CUDA Programming Guide, v3.0, 2010.

[25] NVIDIA Company. Nvidia CUDA C Programming Best Practices Guide, Version 3.0, 2010.

[26] Ken D. Nguyen and Yi Pan,"An Improved Scoring Method for Protein Residue Conservation and Multiple Sequence Alignment," IEEE TRANSACTIONS ON NANOBIOSCIENCE, VOL. 10, NO. 4, DECEMBER 2011

[27] R. F. Smith and T. F. Smith, "Pattern-induced multi-sequence alignment (PIMA) algorithm employing secondary Structure dependent gap penalties for use in comparative protein modeling," *Protein Eng. Design Selection*, vol. 5, pp. 35–41, 1992

[28] J. Taheri and A. Y. Zomaya, "RBT-GA: A novel met heuristic for solving the multiple sequence alignment problem," *BMC Genomics* vol. 7, p. S10, 2009.

[29] An Improved Scoring Method for Protein Residue Conservation and Multiple sequence Alignment IEEE TRANSACTIONS ON NANOBIOSCIENCE, VOL. 10, NO. 4, DECEMBER 2011

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

[30] H. Carillo and D. Lipman, "The multiple sequence alignment problem in biology," *SIAM J. Appl. Math.*, vol. 48, no. 5, pp. 1073–1082, 1988.

[31] K. Fatahalian and M. Houston, "GPUs: A Closer Look," ACM Queue, vol. 6, no. 2, pp. 18-28, 2008.

[32] A reliable Metric for Quantfying Multiple Sequence Alignment 1-4244-1509-8/07/$25.00 2007 IEEE.

[33] Multiple Biological Sequence Alignment: Scoring Functions, Algorithms, and Evaluations Ken D. Nguyen

[34] Exact Multiple Sequence Alignment using Forward Dynamic Programming by LIU KAI.

[35] Multiple Sequence Alignment Group by Liu Kai.

[36] A Survey of Parallel Programming Models and Tools in the Multi and Many-Core EraJavier Diaz, Camelia Mun˜oz-Caro, and Alfonso Nin˜oA

[37] A Heterogeneous Parallel Framework for Domain-Specific Languages Kevin J. Brown_ Arvind K. Sujeeth_ Hyouk Joong Lee_ Tiark Rompf.

[38] http://en.wikipedia.org/wiki/CUDA.

[39] GrABFAST: A CUDA based GPU Accelerated Fast Short SequenceAlignment Algorithm Ankur Narang Jyothish Soman and Sheetal LahabarIBM India Research Labs Vasant Kunj, Delhi, India-110070.

[40]perilsofparallel.blogspot.in/2008/09/larrabee-vs-nvidia-mimd-vs-simd.html.

[41] www.openmp.org.

[42]msdn.microsoft.com/en/us/magazine/cc163717.aspx.

[43] msdn.microsoft.com/en-us/library/0ca2w8dk.aspx.

[44] Global Sequence Alignment using CUDAcompatible multi-core GPUT. R. P. Siriwardena & D. N. Ranasinghe

[45] CUDACL: A Tool for CUDA and OPENCL Programmers Ferosh Jacob David Whittaker, Sagar Thapaliya,Purushotham Bangalore, Marjan Mernik and Jeff Gray.

[46] Fast Parallel Markov Clustering in Bioinformatics Using Massively Parallel Computing on GPU with CUDA and ELLPACK-R S parce Format vol9 NO.3.