International Journal of Computer Applications (0975 – 8887)
Volume 115 – No. 4, April 2015

Shared Memory and Hardware Utilizations for the Parallelization of Local Sequences Alignment using SW Algorithm: A Review

Manhal Elfadil Eltayeeb
Faculty of Computing
Universiti Teknologi Malaysia
Johor Bahru, Skudai

Muhammad S. Abd Latiff
Faculty of Computing
Universiti Teknologi Malaysia
Johor Bahru, Skudai

Ismail Fuzi Isnin
Faculty of Computing
Universiti Teknologi Malaysia
Johor Bahru, Skudai

ABSTRACT

It is becoming increasingly difficult to ignore the importance of aligning DNA and Protein sequences to infer properties of new sequences from well-known reference sequences established and sorted in genetics databanks. Many studies in recent years have focused on different implementations of Sequences Alignment Problems (SAP). However, researcher confused with the ambiguous classification of the SAP. This paper is set out mainly to review, investigate, and analysis current trends in shared memory and hardware implementation of local SAP using Smith-Waterman algorithm. The literatures are addressing and evaluating in order to highlight their advantages and disadvantages.

Keywords

DNA, Protein, Sequences Alignment, Shared Memory, Smith-Waterman, Parallel Computing.

1. INTRODUCTION

Much research in recent years has focused on understanding and identifying DNA and proteins problems including prediction of functional linkages between proteins, which range from identifying a single pair of interacting proteins to analysis a large network of proteins. Furthermore, Protein–Protein Interactions (PPIs) have a major current focus on biological problems necessitated computer-based solutions. PPIs is a Bioinformatics field identify and analyze associations and interactions between various proteins. Analysis of PPIs based on network parameters and measures tools of perturbations [1]. On the other hand, structure prediction and protein folding take a share of attention in studies DNA and proteins problems and remain unclear to date, see Figure 1. However, the major basic issue on DNA and Protein problem is the sequences alignment.

Different techniques and algorithms are applied for analysis, manipulation, and storing of DNA and proteins problems for various applications. The purpose of this paper is to highlight, describe, investigate and examine current focus on alignment problems of DNA and Proteins sequences. Specifically, local sequences alignment, which requires tangible efforts to discover and predict relations and properties of nucleotides or amino acid chains. Computing-based efforts for these problems are discussed in details with concentration on parallel computing.

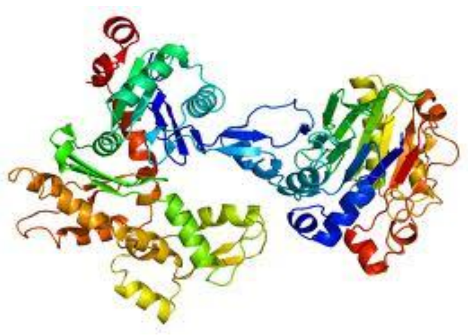
Fig 1: 3D structure of DNA and protein sequences

Parallel computing is a profitable technology for scientist and researchers; it plays a major part in providing fast and reliable tools, see Figure 2. Parallel computing is used as striking feature today in experimental methods for predicting functional linkages in PPIs, protein 3D structure, and the sequences alignment problems. Darriba, et al. [2], adopt a tool for the selecting model for amino acid replacement known as ProtTest. The project is aimed to reduce execution time for model selection in large protein searching using a multicore cluster of desktop PCs. However, ProtTest fails in implementing Petabyte data of protein chains, which represents a major drawback.

A crucial issue in implementing distributed applications over parallel computing is the ability to invent powerful functions and procedures work on a high level of the program middleware. Recently, many organizations adopted Volunteer Computing (VC), a popular term for distributed computing where the work done on computers recruited from people on the Internet. In such way, combination of computer resources from different locations are utilized to reach common objectives. Folding@home as an example of volunteer computing dedicated in statistical calculation of molecular dynamics trajectories for models of biological systems [3]. The project involves a combination of load balancing, result feedback, and redundancy run on volunteer systems. Unfortunately, lack of methodology to consider huge data, offline archives, and off-site backups represent a major challenge in the project.

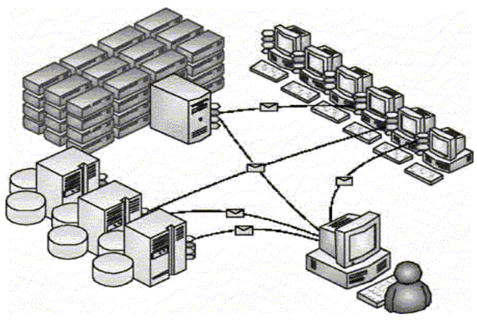
Fig 2: Simple parallel architecture

1.1 The Sequences Alignment Problem

The basic notion in the sequences alignment problem is to find similar regions between two or more sequences by series of mutational processes in order to detect relations between (unknown) common ancestor and known sequences defined in genetic databanks. Alignment of two sequences obtains by stacking them on the top of each other, where matched nucleotides or residues are arranged in successive columns. Matching characters are optimized by inserting spaces at various positions. The final alignment is an assembly of distance events such as matches, mismatches, insertions, and deletions [4]. Matches correspond to similar regions while mismatches or substitutions due to alignments of different characters. Insertion and deletions (known as indels) indicates a lack of coincide matches in one sequence; it represents different biological mutation events. A gap between two sequences occurs if any character matches to the empty space. A key technique in sequences comparisons is to assign a score for each alignment; the optimal score is then the highest score [5]. While, in pattern matching, the comparison consider on words until complete match or mismatch occurred.

Sequences alignments encompass pairwise alignment (particular for two sequences) and multiple alignment dedicated to aligning a number of sequences [6]. Pairwise alignment is a special case from multiple sequences alignments. Most two approaches for aligning pairwise sequences are global and local. Global alignment is convenient if sequences compared as a whole, and compared sequences are homologous across their entire length [7]. Local alignments appropriate for detecting specific conserved regions, and obtain similarity between parts of sequences.

A traditionally challenging area in sequence alignment algorithms for a number of years is to reduce time and space complexities. Numerous algorithms and experiments were established to tackle these issues. Unfortunately, with the tremendous scene of daily increasing of biological data these algorithms do not align precisely or with inaccurate results. There is remains a need for efficient methods with cost effective and accurate results.

1.2 The Smith-Waterman (SW) Algorithm

Smith and Waterman [8], extended Needleman and Wunsch [9] algorithm in order to determine the optimal alignment for local sequence alignment, instead of detecting similarity between the entire sequences. From biologic scene, SW is more relevant because the middle of sequences tend to be highly conserved than the ends [10]. Thus, a weakness at the ends of protein sequences lead to higher mutation, deletion, and insertion rates. The two major differences between SW and NW algorithms including: filling the matrix, in SW no negative values allowed thus 0 value appear as one of the cases in finding an optimal alignment [11], by exclusion negative values SW the stop consider high dissimilarity regions. Furthermore, in NW, the traceback start from the last cell in the scoring matrix, while in the traceback in SW start from the cell with the highest value in the scoring matrix. Considering two sequences (A1,..,Ai) and (B1,..,Bj), local sequence alignment can be resolved using the following equation.

$$s(i,j) = max \begin{cases} s(i-1,j-1) + S(Ai,Bj) & the\ first\ case \\ s(i-1,j) - g & the\ second\ case \\ s(i,j-1) - g & the\ third\ case \\ 0 \quad start\ an\ alignment\ from\ the\ begining & \end{cases}$$

Where, $s(i,j)$ is the optimal alignment of two compared sequence Ai and Bj. In the first case, each residue in query and reference sequences is compared in character-to-character level using the substitution matrix $S(Ai, Bj)$. Second and third cases illustrate the insertion of a gap of length k into one of the compared sequence. Finally, negative values are ignored in a zero case in recursion way. All entries in the first row and column set to zero before the calculations begin, which denote the ability of local similarity to restart at any position in order to perform the comparisons. In SW algorithm, the comparison of sub-sequences recording using a scoring scheme by calculating every possible track for a given cell. In every cell, a score for matches, mismatches, substitutions, insertions, and deletions are considered. The score in each cell represents the maximum score for the alignment of any length ending at specific cell. Optimal alignment is then the highest scoring of the matrix. To allocate optimal alignment a traceback is needed until reaching a zero cell, where the starting point is the cell with the highest value.

The rest of this paper is organized as follows. Section 2 explains hardware implementations for local sequences alignment problem. In section 3, the parallel implementation for the sequences alignment discussed and reviewed in term of shared memory with multicore architecture. Discussion and conclusion are presented in section 4.

2. SHARED MEMORY BASED ON MULTICORE ARCHITECTURES

Aligning similar DNA or protein sequences require powerful tools to reduce complex computation and time consumptions with accurate results. The exponential growth rate of hardware manufacturing in computing component such as CPU, RAMs, caches, etc. promises to alleviate pains of sequences similarity detections. However, the tremendous amounts of biological data reach a Terabyte overcome the sequential computing processing capability. Furthermore, the high costs of extra hardware is dedicated for so problems make it prohibitive. Under this scenario, understanding biological phenomena such as complex evolutionary relation could remain opaque causing to lose vast quantities of valuable information, because of the limitation of CPU-power [12] and the long time required in implementing such problems.

For a number of years, numerous investigations were proposed to address the lack of computing power in sequence alignment problems ranging from incorporating new algorithms into the ROM of a specialized chip to adopting parallel computing model. In parallel computing platforms, two or more processors can be used simultaneously for distributed workload, which represent a solution overwhelm a single sequential processor dilemma. Sequence comparisons are a challenging area in parallel computing; there remains a need for an algorithm to harness additional processing power.

Parallel platforms represent an efficient way to tackle sequences comparison problems.

To develop parallel algorithms for sequences similarity problems, many considerations for designing parallel programs must be studied in essential and advanced stages to produce an efficient parallel application and to maximizing performance and usability within limits of technology and cost. These considerations include the type of parallel programming model, problem partitioning, load balancing, communications, data dependencies, synchronization and race conditions, and memory and I/O issues [13, 14]. However, an important issue in the parallelization of algorithms is the organization of the memory at the parallel models [15]. This section covers in details different directions in parallel platform for implementing sequence alignment problems as general and especially pairwise local sequence alignments.

In a multi-core architecture, the single processor combine more than one processing unit called cores, which are shared one main memory, see Figure 3. The goal of placing multiple cores in a single processor is to create a system that achieves more tasks at the same time, thereby gaining better overall system performance. Many researchers have addressed the problem of implementing the SW algorithm on a multi-core architecture in an attempt to accelerate the similarity detection between two sequences. However, the limitation of memory prohibitive in comparing long sequences, there is remains a need for powerful methods with large spaces. In order to design an efficient parallel version of the SW algorithm based on multi-core platforms many performance parameters must be considered such execution time, scalability, and efficiency.

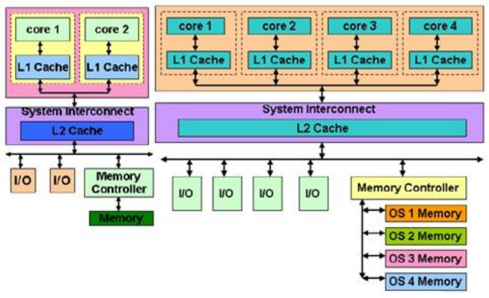

Fig 3: Dual core and Quad core in multicores platforms

Most current discussions in local sequence alignment focus on multicore architectures with shared memory such as divide and conquer techniques [16-18], striped SW [19-22], Instruction-set Processor (ASIP) architecture [23], data compression [24, 25], genome assembly (re-sequence) algorithms [26, 27], and Symmetric Multi-Processing (SMP) architecture [28, 29]. This section discusses in detail these algorithms along with their advantage and disadvantage.

2.1 Divide and Conquer

The divide-and-conquer algorithms solve the problem by breaking it into sub-problems of the same type and recursively solving these sub-problems. The result is a combination of sub-results obtained by sub-problems. Sequences alignment identifies region that is potentially alienable in two input sequences. In a parallel platform, the sequences alignment can be achieving by fragmenting the query sequence and distributed to parallel nodes in master-worker model.

RPAlign algorithm is used to detect regions of similarity between two DNA or protein sequences [16]. It uses BLAST to consider regions that are potentially alienable, while SW algorithm is used for distantly related sequences. The workload is distributed by considering overlapping subsequences w, as the displacement of fragments from the longer sequences. Each fragment Fi is read by any processor from the large sequence, and divided into substrings by considering starting and end positions of the fragment. The comparing sequences distribute for all shared processor to be comparing with the overlapping subsequences. The dynamicity of workload through the running of the program burdens the system and decrease the speedup. Furthermore, each processor generates substring pairs, compute frequency of each element type from a substring pair, compute composite score based on frequency, generate a binary matrix based on composite scores, and transmit binary matrix to master processor. Load balancing is performed through a shared memory framework in order to reduce overhead. However, shared memory is prohibitive for long sequences comparisons.

Bi-Directional Filling (BDF), is the multiple directions algorithm to parallelize SW algorithm proposed by Delgado and Aporntewan [17]; it fill the scoring matrix in row and column wise. The algorithm is designed to work ideally with two cores; every core constraints a loop of (n-(m-1)) column, where n and m are query and the reference sequence respectively. The algorithm successes in reducing waiting time for calculating dependence cells in short sequence. However, for long sequences it increases the time significantly due to constrain of two cores.

Sebastião, et al. [18], analyses the implementations of SW algorithm using data structure of suffix array in order to accelerate DNA sequences alignment. To reduce the time of sequences comparisons a multi-core system is considered in the implementation phase. Two compared sequences are distributed in rows and columns to every core and the scoring matrix is calculated using wave-front method. However, limitation with this approach comes out from shared memory structure, where for long sequences it is prohibitive method. Furthermore, fluid of communication involve in this approach that hinder accelerating of sequence comparisons.

2.2 Striped SW

Striped Smith-Waterman developed by Farrar [30], is a parallel implementation for sequences similarity search using the SW algorithm. It is based on Single-Instruction Multiple-Data (SIMD) architecture, where the sequences are partitioned into p segments equal to the number of register elements in the SIMD.

Borovska, et al. [19], expand the weight of the segment to depend on the processing byte and word integers. If the query sequence is not filling all the segments then the weight is zero. Nevertheless, the shared memory structure has its own drawback, the proposed algorithm wastes much time in measuring weights of sequences in order to decide whether or not to dispatch segments.

Another similar work to Farrar’s Striped is set out by Rognes [20], which define SWIPE, a parallel strategy to manipulate SW algorithm using SIMD. A temporary sequences profile score is considered to reduce searching for similar sequences. While, the comparisons between sequences are carried out in column by column. Furthermore, block of eight cells computed in the each iteration of the inner loop. Many series of blocks in long sequences is an error borne on shared memory architecture.

Ivan, et al. [21], present a web-server technique (SwissAlign) used SW algorithm to find the best alignments for the query sequence from those stored in Swiss-Prot database. The technique is based on a striped SW algorithm to improve processing speed on multiple data in parallel using the SSE2 instruction set. The comparisons with database sequences consider only sequences above a defined value. However the main weakness of the study is the failure to consider other sequences below the e-value.

Mendonca and Melo [22], propose a biological sequences comparisons mechanism for adjusting workload in a master-slave model. The strategy is design mainly for SW algorithm and operated on Multicores architecture and accelerators. Workload is distribute by master to shared Processing Elements (PEs) in two stages; for the first time each PEs assigns to one work, while in the second stage tasks are distributed according to periodic processing progress notifications sent by PEs in order to calculate the weight for each of them. A mechanism adopted to assign the tasks for another slave in case of a slow node receives one of the last tasks. A serious weakness in this study is that the periodical report sends dynamically at the execution time in order to judge the workload distribution, which is wasting too much time. Another problem with this approach is that the data dependency arises in sequence alignment required prior statically allocated for slaves to accelerate sequence comparisons.

2.3 Instruction-set Processor Architecture (IPA)

IAP instruction set is mainly designed to implement the assembly instruction set with minimum hardware cost and to accelerate heavy and most used functions. Neves, et al. [23], extend an algorithm to work on Application Specific Instruction Processor (ASIP) architecture dedicated for sequences alignment problems. Fine grain is obtained by adopting pipeline architecture for SIMD instructions, while coarse grain is obtained by using multicore of multiple ASIPs with a shared memory architecture. Query sequence is divided into (m+p-1)/p parts, where m is the length of the query sequence and p is the number of elements equal to the number of register in a SIMD. Data dependency in SIMD architecture involves complex computation in restricted shared memory, which is an error born and prohibitive for long sequences.

2.4 Data Compression

New technique for comparing sequences after remove repeated nucleotide from sequences is known as Data Compression (DC). DC algorithms [24, 25] remove redundant data in order to understand biologic relevance between sequences.

A compression algorithm to align sequences is proposed by Satyanvesh, et al. [24], it includes two stages. Query sequence is divided into four characters chunks. In the first stage, each character is represented using two bits, while the chunks are represented using either one or two bytes in the next stage. Chunked characters are distributed with reference sequences among multicore architectures in order to find the repeated regions. In a similar study with some changes, Satyanvesh, et al. [25] present a new approach for aligning sequences after compressing them using multi-cores architecture. The alignment is obtained in two phases, which consists gapped and un-gapped alignment. In the gapped alignment, each core synchronizes with others in a loop to calculate the similarity matrix. However, these techniques offer no guarantee to preserve the biologic relevance of comparing sequences. No techniques discuss for balancing workload or even a data-flow algorithm for sending and receive results. Furthermore, due to the rearranging of the query sequence inaccurate results would be a normal corollary.

2.5 Genome Assembly Algorithms (re-sequence)

The re-sequences is a process of constructing an original long DNA sequence by aligning and combining fragments from the source sequence. HPG-aligner [27] implements a parallel pipeline technique for fast and accurate RNA sequences alignment. Query sequence is divided into short regions (seeds); then SW algorithm is applied to detect the region of similarity between these seeds and the reference genome. In the pipeline technique, more than one core are divided into stages where each core holds a seed; these stages connected via queues that act as data buffers and synchronize the consecutive stages. Queue in computing always is a dilemma of time consuming. Thus, the main weakness and drawback of the study is the lack of balancing technique to distribute evenly seeds between cores in order to avoid waiting queues. Furthermore, complex communication is required between stages and inside every stage between cores.

Libgapmis [26] is a library package based on SW algorithm to extend pairwise short-read alignments between substrings of the query sequence to reference genomes sorted and stored in genetic databanks. The Streaming SIMD Extensions (SSE) implementation is used to accelerate SW algorithm under vector scheme. Query sequence is divided to multiple matrixes each with 4 bit vectors and compared with references sequences concurrently on SSE instructions. The fine granularity of the four bits substring increases the communication overhead. Furthermore, a data-flow technique is not offered by algorithm to control the data dependency between shared elements.

2.6 Symmetric Multi-Processing (SMP) Architecture

SMP is the cluster architecture of multiple processors, which shared a common memory, operating system, and the I/O data path, see Figure 4. Clusters of SMP's nodes support differently parallel programming models. However, it significantly increases programming complexity when using the low-level interfaces such as MPI and OpenMP in order to deal with both DM and SM architecture [31]. OpenMP provides an interface for programming SMP between cores, while MPI defines parallelism across processors by calling library function to send and receive messages. Striking features of the hybrid programming using OpenMP and MPI is to combine process level coarse-grain parallelism and fine grain parallelism on a loop level [32].

Numerous experiments have established test-bed for comparing three models to parallelize the SW algorithm including pure MPI, pure OpenMP, and a hybrid model using MPI/OpenMP [28, 29]. Furthermore, evaluations and measurements for the performance of a hybrid model are tested. However, SMP architecture shows circular drawbacks such as limitation of shared processors, hardware complexity, existence of at least one single point of failure, and needing for regular maintenance or update for the whole system.

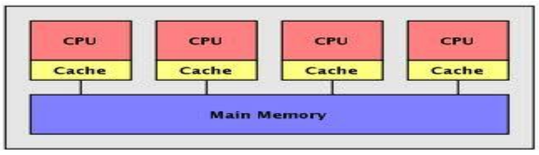

Fig 4: Simple SMP's architecture

For evaluation, Luecke [33] lists difficulties of using shared memory architectures in points, which included lack of parallel programming model, lack of standards, and immaturity of multicore specific development and debug software tools. These obstacles compel researchers to focus on implementing SW in distributed memory where each processor has it is own resources. A central problem of implementing the SW algorithm in multicore and SMP architectures is the shared memory architecture. In the short sequence length, this architecture obtains possible results. However, deficiency arises in comparing long sequences length.

3. HARDWARE UTILIZATIONS FOR SEQUENCES ALIGNMENT PROBLEM

Many hardware manufacturers move toward inventing and developing a dedicated hardware accelerating the computing performance and establishing a scalable architecture consider challenging in identifying and discovering new DNA or proteins sequences from a well-known sequences, which classified and stored in genetic databanks [34]. These new hardware include new chip multiprocessors, which are the cornerstone in hardware accelerator. However, using dedicated hardware in sequences alignments problems involve an algorithm(s) to utilize the feasibility of increasing performance [35].

Using the hardware in sequence alignment problems is restricted by the amount of the hardware memory, which may not support long sequences. Furthermore, the high cost of these devices represent additionally burdening costs [36]. On this section, a review of various hardware accelerator used in sequence alignment problems are discussed in details. While the next section, discusses the parallel implementations of sequences alignment problem with concentration on SW algorithm. The revision includes a Field Programmable Gate Array (FPGA), Graphics Processing Units (GPU), and Network-on-Chip (NoC) and Cell Broadband Engine.

3.1 Field Programmable Gate Array (FPGA)

A Field Programmable Gate Array (FPGA) is an integrated circuit formatted by demands of designers and/or clients after manufacturing [37]. FPGA has a lower clock speed than traditional CPUs, it is mainly used in a high throughput algorithms, which is designed for parallel processing such as prominent algorithms [38].

Meng and Chaudhary [39], propose parallel data prefetching scheme for sequences similarity to accelerate data transfer and improve communication efficiency between host computer and the FPGA coprocessor. Because FPGA consumes much time for communication to host machine a double buffering parallel implementation is designed for the scheme using DMA on the FPGA board and Pthreads on the host machine. In more identical work with some changes Allred, et al. [40] implement SW algorithm in Xeon Front Side Bus module (FSB) using the Intel Accelerator Abstraction Layer (IAAL), a released middleware layer. The proposed algorithm is based on a modification of SSEARCH35, standard industrial software tiles of the SW algorithm, to introduce hardware accelerated option to users. While YILMAZ and GÖK [41], present two systems perform pairwise and multiple sequences comparision. The proposed system is simulated on the FPGA chip (Mezzanine card), while the results are obtained from sychronization with a PC.

3.2 Graphics Processing Units (GPU)

Graphics Processing Units (GPU) is an electronic circuit innovated to accelerate the processing time of images and enhance the computing performance. The notion behind using GPU is to optimized memory access with efficient framework to maximize occupancy. Many experiments were conducted to implement sequences alignments on GPU such as Pairwise Statistical Significance Estimation (PSSE) algorithm [42], which aimed to accelerated estimation of large biological sequences.

CUDA is a parallel programming architecture increases the computing performance by harnessing GPU power. It mainly used as parallel implementations for sequences comparisons using different hardware such as nVidia [43, 44] and GeForce [45]. Many algorithms are developed and implemented based on CUDA platforms, for instance SpecAlign algorithm [46] propose to fast SW alignments using GPU memory. The algorithm is designed to reduce memory accesses and to minimize bandwidth synchronization.

3.3 Network-on-Chip (NoC)

Network-on-Chip (NoC) is a packet switched (integrated circuit) designed using a layered architecture to improve the communication between cores in a networking system. A preliminary attempt to solve sequences alignment problems using NoC is adopted by Sarkar, et al. [47]. A simple pass transistor-based switch boxes are designed to forward the data from one to the other instead of designing network routers for data communication [48]. The NoC architecture achieves speedup, reduced latency, and energy dissipation in communication.

Tile64 is another NoCs platforms produced by Tilera Corporation; it is used in implementing sequences alignment problem. The Tile64 is composed of 64 cores (tiles) integrated into a single Tile64 processor. Each card includes multicore processor, RAM memory, and communication ports. The parallel version of the algorithms FstaLSA and MC64-NW/SW is implemented on Tile64 card in order to detect similarity regions in two compared sequences [7]. The algorithm is based on NW and SW algorithms to optimize the performance of pairwise sequence alignments.

4. DISCUSSION AND CONCLUSION

DNA and Protein problems attract much attention in recent years due to the assumption of it is relations with most diseases. This paper studies and reviews one of the most important problem in Bioinformatics. Sequences alignment problem for two sequences sustains in detecting properties of new query sequence from a well-known reference sequence. This problem is the first step to study some related problems such as functional linkage, protein-protein interaction, structure prediction, and protein folding. Computing methods used in DNA and proteins problems are reviewed in advance as well as storage capacities and space problems. Hardware utilizations are detail and lists such as Field Programmable Gate Array (FPGA), Graphics Processing Units (GPU), and Network-on-Chip (NoC).

This paper is set out with the aim of assessing the shared memory implementation for sequences alignment problem. Specifically, local sequences alignment using the SW algorithm. Current problems of great concern in SW algorithm are computations and spaces complexity, which required powerful algorithm(s) to utilize the power of parallel machines. Implementing SW algorithm in parallel platforms plays a key role in sequences comparisons problems in order to achieve accurate results for long sequence's length within a reasonable time.

The first serious discussions and analyses of long sequences emerge during spaces complexity for comparing sequences. Memory constraint in long DNA comparisons is a prohibitive and compel biologist to lose valuable information from new detecting sequences. Most recent studies in sequence alignment problem have only been carried out in shared memory architecture [16-19, 22-27]. However, a serious weakness with this architecture is the limitation and constraint of fixed sizes of memory available for all shared processors. This major drawback makes any algorithms and/or techniques for long sequences comparisons based on shared memory is unreasonable and impractical. Multicore platforms are designed to work on shared memory architecture; a constraint for memory size would be the normal corollary in these platforms.

5. ACKNOWLEDGMENTS

The authors gratefully acknowledge use of the facilities of the Faculty of Computing at the University Technology Malaysia, funded and support from the grant R.J130000.7828.4F676.

6. REFERENCES

[1] K. Raman, "Construction and analysis of protein-protein interaction networks," *Autom Exp,* vol. 2, p. 2, 2010.

[2] D. Darriba, G. L. Taboada, R. Doallo, and D. Posada, "ProtTest 3: fast selection of best-fit models of protein evolution," *Bioinformatics,* vol. 27, p. 1164, 2011.

[3] A. L. Beberg, D. L. Ensign, G. Jayachandran, S. Khaliq, and V. S. Pande, "Folding@home: Lessons From Eight Years of Volunteer Distributed Computing," *2009 Ieee International Symposium on Parallel & Distributed Processing, Vols 1-5,* pp. 1624-1631, 2009.

[4] M. Axelson-Fisk, "Sequence Alignment," in *Comparative Gene Finding*, ed: Springer London, 2010, pp. 89-155.

[5] M. Imelfort, "Sequence comparison tools," *Bioinformatics.,* pp. 13-37, 2009.

[6] S. Brenner, "Optimal Pairwise Alignment," in *Introduction to computational biology: an evolutionary approach*, B. Haubold and T. Wiehe, Eds., ed: Springer, 2006, pp. 11-42.

[7] D. Díaz, F. J. Esteban, P. Hernández, J. A. Caballero, G. Dorado, and S. Gálvez, "Parallelizing and optimizing a bioinformatics pairwise sequence alignment algorithm for many-core architecture," *Parallel Computing,* vol. 37, pp. 244-259, 2011.

[8] T. Smith and M. Waterman, "Identification of common molecular subsequences," *J. Mol. Bwl,* vol. 147, pp. 195-197, 1981.

[9] S. B. Needleman and C. D. Wunsch, "A general method applicable to the search for similarities in the amino acid sequence of two proteins," *Journal of molecular biology,* vol. 48, pp. 443-453, 1970.

[10] S. Henikoff and J. G. Henikoff, "Amino acid substitution matrices from protein blocks," *Proc Natl Acad Sci U S A,* vol. 89, pp. 10915-9, Nov 15 1992.

[11] R. B. Batista, A. Boukerche, and A. C. M. A. de Melo, "A parallel strategy for biological sequence alignment in restricted memory space," *Journal of Parallel and Distributed Computing,* vol. 68, pp. 548-561, 2008.

[12] G. M. Amdahl, "Validity of the single processor approach to achieving large scale computing capabilities," in *Proceedings of the April 18-20, 1967, spring joint computer conference*, 1967, pp. 483-485.

[13] J. T. Dudley and A. J. Butte, "A quick guide for developing effective bioinformatics programming skills," *PLoS Comput Biol,* vol. 5, p. e1000589, Dec 2009.

[14] S. Hosangadi and S. Kak, "An Alignment Algorithm for Sequences," *arXiv preprint arXiv:1210.8398,* 2012.

[15] T. Rauber and G. Rünger, *Parallel programming: For multicore and cluster systems*: Springer Science & Business, 2013.

[16] S. Bandyopadhyay and R. Mitra, "A parallel pairwise local sequence alignment algorithm," *NanoBioscience, IEEE Transactions on,* vol. 8, pp. 139-146, 2009.

[17] G. Delgado and C. Aporntewan, "Data dependency reduction in Dynamic Programming matrix," in *Computer Science and Software Engineering (JCSSE), 2011 Eighth International Joint Conference on*, 2011, pp. 234-236.

[18] N. Sebastião, G. Encarnação, and N. Roma, "Implementation and performance analysis of efficient index structures for DNA search algorithms in parallel platforms," *Concurrency and Computation: Practice and Experience,* pp. n/a-n/a, 2012.

[19] P. Borovska, V. Gancheva, G. Dimitrov, and K. Chintov, "Parallel performance evaluation of multithreaded local sequence alignment," in *Proceedings of the 12th International Conference on Computer Systems and Technologies*, 2011, pp. 247-252.

[20] T. Rognes, "Faster Smith-Waterman database searches with inter-sequence SIMD parallelisation," *BMC bioinformatics,* vol. 12, p. 221, 2011.

[21] G. Ivan, D. Banky, and V. Grolmusz, "Fast and Exact Sequence Alignment with the Smith-Waterman Algorithm: The SwissAlign Webserver," *arXiv preprint arXiv:1309.1895,* 2013.

[22] F. M. Mendonca and A. C. M. A. d. Melo, "Biological Sequence Comparison on Hybrid Platforms with Dynamic Workload Adjustment," in *Parallel and Distributed Processing Symposium Workshops & PhD Forum (IPDPSW), 2013 IEEE 27th International*, 2013, pp. 501-509.

[23] N. Neves, N. Sebastiao, A. Patricio, D. Matos, P. Tomás, P. Flores, *et al.*, "BioBlaze: Multi-core SIMD ASIP for DNA sequence alignment," in *Application-Specific Systems, Architectures and Processors (ASAP), 2013 IEEE 24th International Conference on*, 2013, pp. 241-244.

[24] D. Satyanvesh, K. Balleda, A. Padyana, and P. Baruah, "GenCodex-A Novel Algorithm for Compressing DNA sequences on Multi-cores and GPUs," in *19th IEEE International conference on High Performance Computing., December 2012.*, 2012.

[25] D. Satyanvesh, K. Balleda, and P. Baruah, "Genalign—A high performance implementation for aligning the compressed DNA sequences," in *Advanced Computing*

Technologies (ICACT), 2013 15th International Conference on, 2013, pp. 1-6.

[26] N. Alachiotis, S. Berger, T. Flouri, S. P. Pissis, and A. Stamatakis, "libgapmis: extending short-read alignments," *BMC Bioinformatics,* vol. 14, p. S4, 2013.

[27] H. Martínez, J. Tárraga, I. Medina, S. Barrachina, M. Castillo, J. Dopazo*, et al.*, "Concurrent and Accurate RNA Sequencing on Multicore Platforms," *arXiv preprint arXiv:1304.0681,* 2013.

[28] M. Noorian, H. Pooshfam, Z. Noorian, and R. Abdullah, "Performance enhancement of smith-waterman algorithm using hybrid model: comparing the MPI and hybrid programming paradigm on SMP clusters," in *Systems, Man and Cybernetics, 2009. SMC 2009. IEEE International Conference on*, 2009, pp. 492-497.

[29] M. J. Chorley and D. W. Walker, "Performance analysis of a hybrid MPI/OpenMP application on multi-core clusters," *Journal of Computational Science,* vol. 1, pp. 168-174, Aug 2010.

[30] M. Farrar, "Striped Smith–Waterman speeds database searches six times over other SIMD implementations," *Bioinformatics,* vol. 23, pp. 156-161, 2007.

[31] H. Q. Jin, D. Jespersen, P. Mehrotra, R. Biswas, L. Huang, and B. Chapman, "High performance computing using MPI and OpenMP on multi-core parallel systems," *Parallel Computing,* vol. 37, pp. 562-575, Sep 2011.

[32] G. Hager, G. Jost, and R. Rabenseifner, "Communication characteristics and hybrid MPI/OpenMP parallel programming on clusters of multi-core SMP nodes," in *Proceedings of Cray User Group Conference*, 2009, p. 5455.

[33] K. R. Luecke, "Software Development for Parallel and Multi-Core Processing," in *Embedded Systems - High Performance Systems, Applications and Projects*, ed: InTech., 2012, pp. 35-58.

[34] S. Aluru and N. Jammula, "A Review of Hardware Acceleration for Computational Genomics," 2013.

[35] P. K. Lala, "A Digital Hardware-based Approach for Molecular Sequence Comparison," in *Information Engineering*, 2013.

[36] T. Majumder, P. P. Pande, and A. Kalyanaraman, "Hardware Accelerators in Computational Biology: Application, Potential and Challenges," 2013.

[37] B. Vibishna, K. S. Beenamole, and A. K. Singh, "Understanding single-event effects in FPGA for Avionic system design," *Iete Technical Review,* vol. 30, pp. 497-505, Nov-Dec 2013.

[38] A. Surendar, M. Arun, and C. Bagavathi, "EVOLUTION OF RECONFIGURABLE BASED ALGORITHMS FOR BIOINFORMATICS APPLICATIONS: AN INVESTIGATION," *International journal of life sciences, Biotechnology and Pharma Research,* vol. 2, 2013.

[39] X. Meng and V. Chaudhary, "Boosting data throughput for sequence database similarity searches on FPGAs using an adaptive buffering scheme," *Parallel Computing,* vol. 35, pp. 1-11, Jan 2009.

[40] J. Allred, J. Coyne, W. Lynch, V. Natoli, J. Grecco, and J. Morrissette, "Smith-Waterman implementation on a FSB-FPGA module using the Intel Accelerator Abstraction Layer," in *Parallel & Distributed Processing, 2009. IPDPS 2009. IEEE International Symposium on*, 2009, pp. 1-4.

[41] C. YILMAZ and M. GÖK, "System designs to perform bioinformatics sequence alignment," *Turkish Journal of Electrical Engineering & Computer Sciences,* vol. 21, pp. 246-262, 2013.

[42] Y. Zhang, S. Misra, D. Honbo, A. Agrawal, W. Liao, and A. Choudhary, "Efficient pairwise statistical significance estimation for local sequence alignment using GPU," in *Computational Advances in Bio and Medical Sciences (ICCABS), 2011 IEEE 1st International Conference on*, 2011, pp. 226-231.

[43] P. Borovska and M. Lazarova, "Parallel models for sequence alignment on CPU and GPU," in *Proceedings of the 12th International Conference on Computer Systems and Technologies*, 2011, pp. 210-215.

[44] A. Papadopoulos, I. Kirmitzoglou, V. J. Promponas, and T. Theocharides, "GPU technology as a platform for accelerating local complexity analysis of protein sequences," in *Engineering in Medicine and Biology Society (EMBC), 2013 35th Annual International Conference of the IEEE*, 2013, pp. 2684-2687.

[45] S. Manavski and G. Valle, "CUDA compatible GPU cards as efficient hardware accelerators for Smith-Waterman sequence alignment," *BMC bioinformatics,* vol. 9, p. S10, 2008.

[46] X. Feng, H. Jin, R. Zheng, L. Zhu, and W. Dai, "Accelerating Smith-Waterman Alignment of Species-Based Protein Sequences on GPU," *International Journal of Parallel Programming,* pp. 1-22, 2013.

[47] S. Sarkar, G. R. Kulkarni, P. P. Pande, and A. Kalyanaraman, "Network-on-chip hardware accelerators for biological sequence alignment," *Computers, IEEE Transactions on,* vol. 59, pp. 29-41, 2010.

[48] S. Kaur, "On-chip Networks!," *Iete Technical Review,* vol. 30, pp. 168-172, May-Jun 2013.