ISSN 2047-3338

Distributed Memory and Shared Distributed Memory Architecture for Implementing Local Sequences Alignment: A Survey

Manhal Elfadil Eltayeeb Elnour[1], Muhammad Shafie Abd Latif[2] and Ismail Fauzi Isnin[3]
[1,2,3]Department of Computer Science, Faculty of Computing, Universiti Teknologi Malaysia, Johor, Malaysia
[1]manhalus@gmail.com, [2]shafie@utm.my, [3]ismailfauzi@utm.my

***Abstract*—Recently, researchers have shown an increase interest in Sequences Alignment Problem (SAP), in which the properties of new DNA and Protein sequences are detected by comparing them with well-known reference sequences defined in genetic databanks. Considerable amounts of literature have been published on SAP. However, less attention has been paid to the revision and classification of current techniques in SAP. Thus, the purpose of this paper is to review recent trends in implementing local SAP in a parallel architecture. The classification is based on Distributed Memory and Shared Distributed Memory architectures for different Dynamic Programing and Smith-Waterman based algorithms. The literature are studying and evaluating in order to highlight their strength and weaknesses.**

***Index Terms*—Sequences Alignment, Shared Distributed Memory, DNA, Protein and Parallel Computing**

I. INTRODUCTION

THE alignment of DNA and Protein sequences has received much attention in recent years due to the vital information may infer from new sequences defined and stored previously in genetic databanks. Furthermore, deciphering DNA sequences help to elucidate genetic information from any given biological system, which known as Next Generation Sequencing (NGS) [1]. However, traditional sequences alignment algorithms are computationally expensive [2], [3] or inaccurate in their results [4], [5]. Usually, complex algorithms always lead to computational overhead using a single processor, which can be minimized by invoking multi-processor and multi-core platforms [6]. Develop robust algorithms to consider these complications in a reasonable time with accurate results would be highly concerned for NGS researchers. Furthermore, the dramatic increasing of bio-data such as DNA, RNA, proteins, and human genome involves adopting powerful techniques to analysis new sequences based on record sequences in genetic databanks.

On the other hand, genetics databanks work closely with bioinformatics institutions to establish bioinformatics projects, such as GenBank [7], [8] and Protein Data Bank in Europe (PDBe) [9], [10]. PDBe establishes a website responsible for maintaining the single worldwide repository of bio-macromolecular structure data. It objectives are to handle the annotation of bio-data through a website, to be high-quality macromolecule resource provider, and to be expertise in determination techniques (X-ray, NMR and EM) by working with bio-community. Most of Molecular Biology Databases (MBD) includes information about DNA, RNA, and Proteins. It is also examine protein structures, prediction, and interaction. The majority of well-known biological databases are depicted in Figure 1. These databases are available online and cover various area of molecular biology [11]. The Listing of Molecular Biology Databases (LiMB) is database of databases aimed to employ a method manipulate molecular biology information and related databases by providing a platform for designing automatic access to distributed biological data sets.

The significant increase of Petabytes of biological datasets makes centralized storage techniques inconvenient [12]. There remains a need for distributed storage techniques providing high durability and flexibility for storing large datasets [13], [14].

P-found, is a project provides tools to support comparison and analysis of large distributed simulation datasets [15].

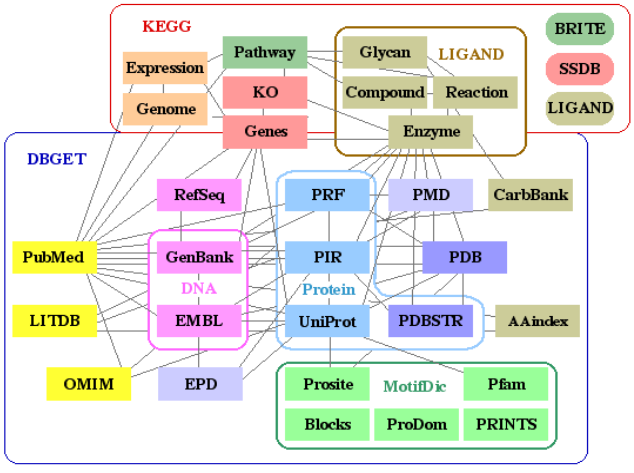

Fig. 1. Noted Genetic-databases dedicated in DNA and Protein problems

The project includes two storages, primary dataset storage for protein simulation and data warehousing, which contain information about protein properties and their transformations. It is aimed to solve the problems of protein simulations by presenting data repository including all the simulated datasets and results. Unfortunately, network lateness causes delays in reading and writing for clients when using distributed storage media. This arbitrary failure arises from synchronization in client with others and servers.

An alternative approach is developed by Storage@home [16], a distributed storage infrastructure developed to solve the problem of backing up and sharing petabytes of DNA and Protein findings using a distributed model of volunteer hosts. The system includes many functions derived-by policy engine, storage clients, the metadata server, the registration server, and identity server. However, many problems arise in the project ranging from slow upload, often-disconnected hosts. The challenging issue is to discover an algorithm considers store and retrieve data efficiently. Some distributed storage algorithms such as ABD algorithm [17], SBQ-L algorithm [18], and ACKM algorithm [19] try to solve the problem of synchronization and client's corruptions. However, these algorithms overlook analyzing procedures to store the data and concentrate on transferring data between base objects.

Despite of all storage problems in genetic databanks, the issue that has grown an importance in light of recent years is the sequences alignment. It is becoming increasingly difficult to ignore the importance of aligning DNA and Protein sequences. Sequence alignment problem is the first step in detecting homologous sequences, where the evolutionary relationship may infer by comparing sequences. The central problem in sequence alignment is the comparisons of long sequence length, wherein restricted memory is prohibitive and impractical. This paper mainly reviews the sequence alignment problem and their implementations in parallel platform.

The rest of this paper is organized as follows. Section 2 explains dynamic and heuristics methods used in sequences alignment problems. In section 3, variant algorithms based on dynamic programming methods are discussed, while, in section 4, SW-based algorithms for implementing local sequence alignment are addressed. Section 5 concludes the paper with a discussion on the literature contributions.

II. The Sequences Alignment Problem

In sequence alignment, compared sequences are arranged on top of each other in rows wise, while match letters appeared in sequential columns. Similarity between two sequences can be optimizing by inserting spaces and deleting letters at different places. Optimal alignment is achieved via sets of operations and steps such as matches, mismatches, substitutions or replacements, insertions, and deletions (insertions and deletions always abbreviate as *indels*). These operations known as *mutation events*, which in turn change the first sequence into the second [20]. While, the *edits distance* defines the number of mutation events involve to transform sequences. Matches coincide with similar letters while mismatches or substitutions coincide with different letters. Insertion and deletions (also defined as a gap operation) denotes by coinciding to presence or absence of match letter in another sequence. Typical indels in a sequence implies gaps or empty reigns in the other sequence [21].

In aligning two sequences, some regions show high similarity; thus a technique for quantify the best alignment is needed. The striking feature is to assign a score for each alignment. The alignment with the highest score is then recognized as an *optimal score*. However, optimal alignment in similarity detection between sequences is obtained via scoring scheme. Scoring scheme includes a substitution matrix as well as a gap penalty [22]. Substitution matrix is used to calculate matches and mismatches regions, while a gap penalty scores insertion and deletion events. The alignment of sequences is the sum of individual event scores. In literature, different substitution matrix and gap penalty are used in the scoring scheme. The notion behind scoring systems is to reduce the number of mutations events needed to transform one sequence into another. In a biological viewpoint, aligning two sequences is the procedures of minimizing the total number of evolutionary changes estimated by an alignment. While, in a computational viewpoint, it is equivalent to maximizing the score of similarity between two sequences. Thus, finding an optimal score involves recording the best alignment with the maximum score when calculating a substitution matrix and a gap penalty in a scoring system.

A. Dynamic Programming Methods

In recent years, there has been an increasing amount of literature on Dynamic Programming (DP) implementation for sequence alignment problems. Needleman and Wunsch [2], is the pioneer algorithm in global sequence alignments. It mainly used in aligning pairwise of protein sequences; however, it can be used to align DNA sequences as well. The algorithm places two sequences in the matrix S of two dimension $(m \times n)$, where m and n represent the number of nucleotides in any two sequences. It is designed to compare sequences as a whole, while Smith and Waterman [3] (SW) algorithm identifies homologies subsequences among sets of long sequences by considering optimal local pairwise alignments between two sequences. In SW algorithm, sequences are compared using two dimensional matrices, where the values filled in the matrix using different methods such as wave-front [23]. Optimal alignment is obtained by considering scoring scheme including matches, mismatches, gaps, and substitution matrix. The results are obtained using a trace-back procedure to recover the best local alignment. Gotoh [24], improves SW algorithm using affine gap weight for large sequences. The algorithm is assumed to found the minimum cost of aligning two sequences. An alternative approach developed by Nordin, et al. [25] in five phases include query initialization, patterns generating, pattern's scanning, ranking, and optimal local alignment. The model considers sequences with highest exact matching scores are the most similar to the query sequence. However, many stages of the proposed algorithm require a

huge amount of spaces to store results in every stage, which is a source of wasting time.

Linear space offers a mean of enhancing or improving the space complexity in similarity detections for homologous sequences. Hirschberg [26], a pioneered in linear space implementation for sequence alignment problems propose an exact algorithm calculating global alignment between two sequences *M* and *N* in quadratic time. The proposed approach split sequence *M* in the middle and generating subsequences M_1 and M_2, then calculates corresponding place for sequence *N* and generating subsequences N_1 and N_2. In such way, the alignment is solved in divide and conquer recursive manner. This recursion roughly doubles the execution time when compared with the original algorithm. Nevertheless, for long biological sequences, which would otherwise generate very huge similarity matrices, could be appropriate.

In the sequence comparisons using DP, substitution matrix measures the rate of changing over a period for one residue or nucleotide in a sequence. Similarity between sequences depends on these rates, where the values of both characters are considered in scoring scheme. Two well-known substitution matrixes always are used in similarity detections, the Point Accepted Mutation (PAM) series and BLOck SUbstitution Matrix (BLOSUM) families, see Figure 2.

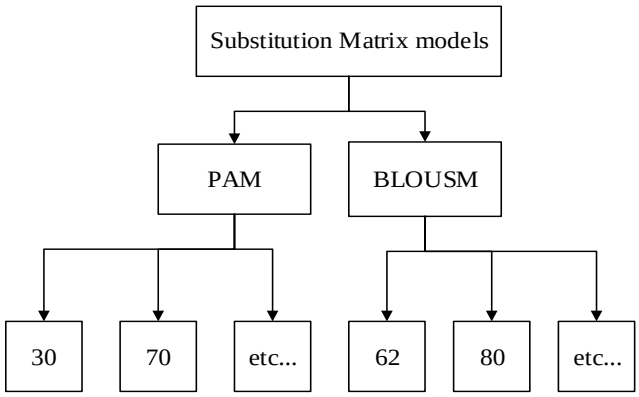

Fig. 2. Substation matrix models

Point Accepted Mutation (PAM) Series:

PAM series, is a substitution model for aligning two sequences to detect regions of similarity [27]. The core notion of the statically series is based on observed percent of evolutionary alteration of structurally and chemically similar amino acids when mutated in a large number of high-quality alignments. The evolutionary alterations occur between any two amino acids to a higher extent when they were similar in the structure and chemical proprieties, otherwise changes between less similar amino acids are assumed to be neglected. The percent of the evolutionary alteration of amino acids is constituted in building a scoring scheme for aligned amino acids. In PAM1, the rates of amino acid substitutions are occurred on an average 1% (one sequence mutated). While for further expectations rates on an average 2% the residues also mutated (PAM2 matrix), following this the matrices is calculated up to PAM250.

BLOck SUbstitution Matrix (BLOSUM) Series:

BLOSUM series proposed by Henikoff and Henikoff [28] is based on observation of distantly related sequences tend to be highly preserved regions or blocks intervened by less conserved stretches of sequences. In BLOSUM, probability values based on chemical proprieties for two residues are used in building substitution matrix for sequence alignment [29]. These values append to the matrix name such as BLOSUM62, which denote that two sequences are compared with less than 62% similarity. BLOSUM80 obtains the best results when aligning closely related sequences than BLOSUM30, which suits for aligning highly diverged sequences.

NW and SW algorithms obtain accurate results, however computationally they are very expensive. Next subsection discusses heuristic-based methods for sequence alignment problems, as it produced faster results than DP algorithms.

B. Heuristic-Based Methods

Heuristic-based methods for sequence alignment are based on filtering technique to enhance the sequence comparison speed. Where any algorithm scans reference sequences to find exact matches of subsequences in the query sequence, it casts down any irrelevant subsequences from searching for an optimal alignment.

FASTA [4], is searched and matched for approximation of *k-tuples* of length *k* or subsequences of related sequences, to produce an optimal alignment. FASTA is based on the notions of related sequences have identical regions. The algorithm produces a hash table of all *k-tuples*, which detects in the query sequences. It then defines the location of all the *k-tuples* in the entire reference sequences and inputs them into the table. Each *k-tuple* in the query sequence is founded in the hash table and any matched regions of the query sequence allow FASTA to mark the matching cells in the matrix. These result in a matrix constituted to mark all points of local identity of length *k*.

On the other hand, BLAST [5] uses words hitting *(w)* in searching for similarity between sequences by heuristically optimize a measure of sequence similarity known as the Maximal Segment Pair (MSP). The MSP defines identical length segments chosen from two sequences, and reported as the highest scoring segments in a query sequence. BLAST considers query sequence and reference sequences in genetic databanks to produce a list of high scoring words of a length *(w)* from the query sequence, and scanned for hits (matches) and the place of occurrence.

An attempt to improve heuristic-based methods is proposed by Hudek and Brown [30], FEAST a pairwise local alignment program is based on probabilistic and prediction models. In the first stage, the program aligns subsequences pair by identified a homologous pair position and seeks forward and backward to produce an extension pairs. While the second stage is based on Hidden Markov Model (HMM) using different segments of an aligned sequences with different alignment parameter. Finally, an expectation maximization

training procedure incorporates extension and alignment algorithms.

Zhang, et al. [31], Li and Homer [32], and [33] in a characterized work compare most of DP and HM sequences alignment algorithms including NW, SW, FASTA, and BLAST. The comparisons are aimed to highlight computations and space complexity in term of performance parameters for optimal alignment. These parameters include speed, running times, and affine gap penalties. As a result, there is a tradeoff between speed and sensitivity, BLAST and FASTA consider subsequences *k-tuples* and *words* to achieve higher speeds, while NW and SW produce optimal and accurate results, but scarifying with the speed in alignment.

This paper set out mainly to review the parallel implementation of DP methods for local sequences alignment. Specifically, the paper discusses Distributed Memory (DM) and Shared Distributed Memory (SDM) architectures. In a DM system, individual processors associated with separated memory and a processor is only allows to access its own memory. DM architecture can be classifying into multicomputer and Massively Parallel Processor (MPP). On the other hand, SDM is a technique allowing users' processes to access shared data without using inter-process communications, see Figure 3.

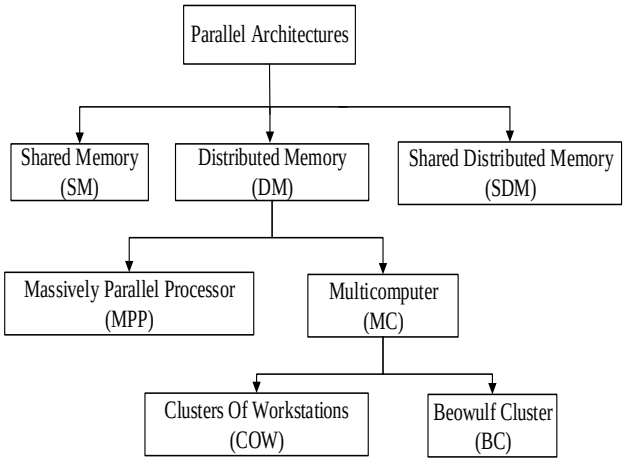

Fig. 3. Classification of parallel architecture

III. Non SW-based Algorithms for Similarity Detection Using DP Methods

A central problem of sequence similarity search focused on matched sequences in a two-dimension matrix and considering deletions, insertions, and gap penalty. This matrix requires complex computations in order to discover a biologically relevance between compared sequences. A key technique of growing interest to obtain an optimal alignment using DP methods is parallel platforms. In DP, the original problem is divided into sub-problems. The final optimal alignment is obtained by considering optimal result for every sub-problem, see Figure 4. The majority of earlier experiments for parallelizing sequence alignment are conducted to run jobs in a parallel with workload distribution methods. Different lengths of sequences are distributed among shared processors. After each job finished results, new sequence started reading at a different time. In most parallel algorithms global communicator is spilled to sub-communicators, each one reads in the genetic databases and broadcasts it inside group and then concatenate the output files that each process generates inside each sub-communicator and writes the result into one file [34].

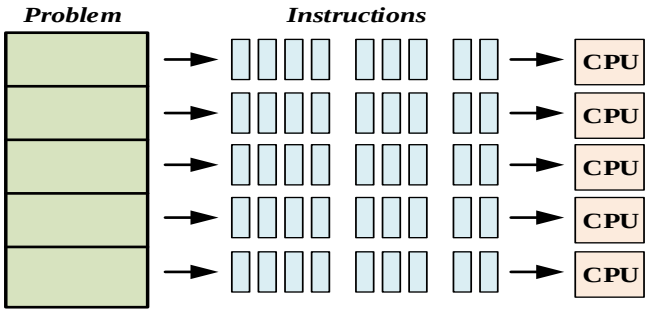

Fig. 4. DP methods distributed chunks of the problem to multiple processors

Several attempts have been made to implement similarity detection problems on SDM architectures using multicore machines. However, one major drawback of this approach is the limitation of memory, which is definitely prohibitive for long sequences comparisons. Performing fast computation results and minimizing the total execution time alway representing major achievements for adopting multicore architectures. In multicore architecture, single processor(s) with independent Central Processing Units (CPUs) called "cores" play an important role in accelerating computing performance, especially for scientific computing, where complex computations required for solving one problem, it significantly reduced the execution time in the comparisons of small sequences length. Pipelined techniques offer a mean of communication techniques between two or more cores and/or processors for decomposing the computation workload in a multi-core architecture. While, a block parallel DP algorithm increases the amount of workloads for each shared core.

A parallel approach for detecting homologies DNA sequences in multicore architecture is proposed by Sathe and Shrimankar [35] to increase data parallelism on sequences comparisons using OpenMP API with tiling techniques. The tiling technique in the algorithm is to optimize compiler and to improve data locality in parallel program in order to maximize parallelism and minimize synchronization. The algorithm scatters different size of chunks to different cores and obtained comparative results.

A modified version with some changes is EasyPDP [36], a runtime system is based on Directed Acyclic Graphs (DAG) Data Driven Model for parallelizing DP algorithms on multicore and multiprocessor platforms. The DAG Data Driven Model consists of three modules: user application module, DAG pattern module and DAG runtime system module. The system handles fault tolerance, data partitioning,

dynamic data task allocation and scheduling, and thread creation.

In contrast to SDM architectures, DM architecture is applicable approach to tackle long sequence comparisons. In DM, many processors with its own private memory incorporated to solve a complex phenomenon. Computational tasks distributed among shared processors while data exchanges through communication medium. An algorithm based on DM approach has a number of attractive features. However, communication complexity is a major challenging issue. Master-Worker is a widely common technique used in DM. In the parallelization of similarity detections algorithms, this technique achieves good performance. The notion behind this technique is to distributed workload as a chunk size between workers according to load balance algorithm [37]. The master calculates chunk size and obtains the tasks of the workers. MPI message-passing defines the communication procedures among different processors. Time delays, which occur between processors in the data communication is a crucial issue for measuring the efficiency of parallel computation [38].

IV. SW-based Algorithms for Similarity Detection Problem

Smith–Waterman algorithm (SW) detects similar regions between two sequences using DP techniques. The algorithm compares two sequences locally on a character-to-character level in order to define optimal local alignments and to discover subsequence that are potentially similar. On sequential machines, SW is an expensive algorithm. However, parallel architecture paves the way for a powerful method to overcome sequential dilemma. Parallelization of SW algorithm is a daunting work, because of the nature of the data dependency arising when calculating the similarity matrix for the compared sequences. Another complexity of data dependency occurs by the designing of the memory, whether it is DM or SDM. Furthermore, the architecture of the parallel model represents an important issue in designing a parallel system able to manage a huge amount of data dependency.

DM systems consist of multiple-processors machines, where each processor has its own memory, see Figure 5. Computational jobs operated on local data for each processor while remote data required complex communications with other processors. An advantage of using DM systems includes feasibility of increasing further numbers of processors as well as memory in order to increase system throughput and efficiency. Furthermore, each processor work with it is own data within local memory. However, the major drawback in DM system includes adopting efficient load balancing algorithm in order to minimize inter-processor communications [39].

Developing parallel algorithms for sequence alignment is daunting work requires in-depth knowledge of Bioinformatics discipline as well as parallel techniques [31], [40]. Few researchers have addressed implementing SW algorithm on a DM. Some of these are divide and conquer algorithms [26], [39], [41]-[43] and wavefront methods [44], [45].

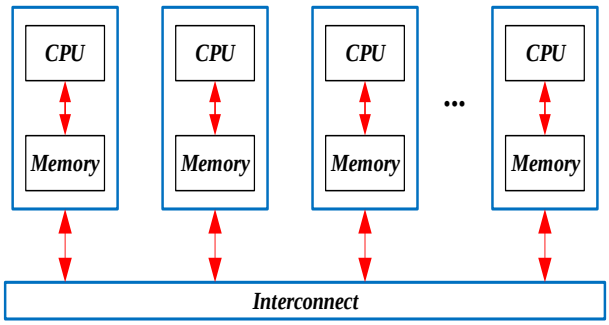

Fig. 5. Main architecture of Distributed Memory (DM) systems

A. Divide and Conquer (DC)

DC technique provides a parallel paradigm based on multiple recursion of the problem into two or more sub-problems. The solution for the original problem is obtained by a combination of sub-problems. DC technique is a changing area for a number of years for most of the models, which are carried out in similarity detection problems.

Hirschberg [26], pioneer in linear space alignment with quadratic time presents a recursive divide and conquer implementations. A modified version of very similar algorithm adopted by Myers and Miller [41] with Gotoh [24] affine gap penalty. The central notion of both algorithms are to find a midpoint for an optimal alignment and recursively determine optimal point on both sides of the midpoint. In very similar work with some changes Huang, et al. [42] extended Myers and Miller [41] algorithm to fit for subsequence matching. However, quadratic time still remains complex.

Nordin, et al. [46], develop FRA-Search model to improve comparisons of large DNA sequence. Two approaches are used in the model: the string-matching algorithm and rough set theory. Sequences are partitioned into *k* parts based on the machine identification number in the parallel version of the model using master worker process farm. Master processors distribute workload to the idle worker and receive the results after calculating optimal alignment; this process iterates until all compared sequences finish. The model lack of load balancing strategy to efficiently improve the movement of the partitions. Furthermore, waiting for idle workers influence a full utilization of resources. A much more systematic approach would identify how workers interact with other in order to reduce waiting time for results from every worker.

In another attempt with the same researcher Nordin and Rahman [39] present a parallel technique for comparing DNA sequence using SW algorithm. Compared sequences are partition into *k* blocks, which it is replicate to each processor. Each worker has its own dataset and memory. While, for the programmer decides how the data should be distributed among the PCs. The existing techniques for distributing workload among processors depend on the user, which is clearly unacceptable way to balance between shared workers. Furthermore, calculating matrix involves periodical

communications between workers and master, thus losing valuable data can be occurred.

Wu, et al. [43], propose pGraph, a parallel algorithm for detecting homology's protein sequences using a hierarchical multiple producer-consumer models. The algorithm uses Generalized Suffix Tree (GST) data structure for distributing tasks dynamically between super-master (the main node) and the producers, which include masters and consumers nodes. Task is distributed equally among subgroups. The main advantage of such model is that it avoids a single point breakdown and hides all overheads of data movement. However, this approach has limitations such as changing subgroup and buffer sizes, which require powerful parallel model. The limitation of consumer local memory hinder alignment input sequences and the dynamicity of workload distribution wasted time in allocating subgroup to perform calculations of scoring matrix.

Montanola, et al. [47], combine distributed and shared memory architecture to implement the SW between nodes and within cores inside node. Pairs of comparing sequences are generated and distribute into three groups according to A-Z alphabetical. Each worker manipulates a part of the subset of pairs that made from letters combinations. Unfortunately, imbalance distribution of workload maximizes the communications and the computation cost of the worker tasks. Furthermore, due to the limited letters used in residues (twenty letters) and nucleotide (four letters) it is clear that the parts are not equal in their length.

B. Wave-front Method

Wave-front is a technique adopted by Wozniak [23], it mainly breaking the computation problems into fragments where it distributed through shared processors. Optimal granularity for any problem increase machine efficiency and performance such as computational speed, and inter-process communication latency [48]. However, a serious weakness in wave-front method is that it required flood of communications in orders to solve any problems. This limitation for long sequences length is a bottleneck.

Few researchers use wave-front method to tackle the problem of filling matrix in sequences alignment. Z-align algorithm proposed by Batista, et al. [44] is to align biological sequences locally into four stages using a parallel platform. In order to run the comparisons in limited memory space, the matrix is divided vertically into sub-matrices, each processor calculating $(\frac{m}{p} \times n)$, where m and n are sequences lengths and p is the number of processors. Each sub-matrix is divided horizontally and vertically into blocks of two rows. Wave-front method requires each processor to send the calculated cell to the next processor. Communication complexity between processors specifically for a long sequence makes this solution impractical.

SWAMP+ [45] extends an approach for implementing SW in parallel platforms using Associative Computing model (ASC). ASC consists of an array of cells and a single instruction stream; the cells compose of a Processing Elements (PEs) each with a local memory and connecting via network links. PEs listens to the instruction stream, which broadcast data and instructions, see Figure 6. In associative models, *(m+1)* PEs are used to calculate the scoring matrix using wave-front approach, each PE calculates one row in SW. The major drawbacks of this approach are the limitations of PEs and wave-front vector due to the memory constraints in each PE as well as the data dependencies in vector, which limit the sequences length and make long sequences comparisons unreasonable.

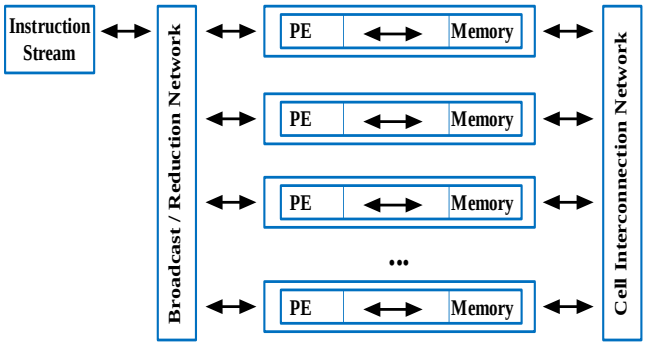

Fig. 6. Associative computing models (ASC)

For evaluating DM and SDM Sánchez Castaño, et al. [49] and Sánchez, et al. [50] analyses the performance of the Ssearch, a parallel application implements the SW algorithm on shared and DM architecture machines. Two different synchronization strategies for the distribution of workload on multicore architectures are analyzed. The study suggested the efficient use of coarse and fine grain parallelism accelerates the sequence comparisons. However, the limitation of memory could be prohibitive. An observation in the study concludes the possibility to minimize the impact of memory latency by adopting double buffering techniques with large data blocks. Furthermore, a strategy based on preventing a worker to wait for the response of the previous signal can minimize synchronization overhead and maximize the performance of the application.

V. Discussion and Conclusion

Most studies in the sequence alignment focus only on comparing DNA sequences [39], [44], [46], [51]-[56] more than RNA [1] or proteins sequences [43], [47], [57]-[59]. In reality, sequences comparison is a problem for long sequences length, while, in small sequences sizes, SW performs good results. Nowadays, DNA length reaches 130,000,000,000 (130GBP) in largest vertebrate genome known while the largest known protein is 27,000 residues [10], [60]. However, the existing studies fail to compare more than 3 MBP [44], there is still a need for an algorithm capable of comparing longest length of DNA sequences.

DM and SDM are prominent platforms tackle long sequence comparison problems. A variety of algorithms are used to implement sequence alignment problems in DM and

SDM [39], [46], [47]. Each has its advantages and drawbacks. Applicable parallel platform for DM and SDM is multiprocessors architecture, where each processor has it is own memory. However, one of the major drawbacks of multiprocessor platforms is communication complexity in shared processor. Hence, it could conceivably be hypothesized that the existing load balancing algorithm to distribute tasks across multiple processors could significantly reduce this complexity.

One of the most current discussions in parallel computing is the task distribution between shared processors. Difficulties are arising, however, when an attempt made to distribute tasks as well the system performance would increase significantly. To date various methods have been developed and introduced in workload and tasks distribution in sequences alignment problems. In most recent studies, workload distribution for parallelization of the similarity matrix in sequence alignment problems has been implemented in four different ways: substring of character [44], [47], [53], [55], [59], rows and column [44], [51], [54], [56], partitioning into segments [39], [46], [52], [58], and stages of distribution [1], [43], [57]. In substring of three or four characters, algorithms tend to distribute the workload between shared processors into groups of characters or substrings. Since sizes of these groups are too small compared to sequence length, thus a fine-grain parallelism is applied. Traditionally, fine-grain is very complex in communications; these can be time-consuming and technically difficult to perform. Rows and column method distribute workload as a matrix of rows and columns to every shared processor. Unfortunately, these methods do not always guarantee passing dependent cells for other processors. Furthermore, there are no clear techniques adopting for controlling shared variables in memory. Partitioning into segment's methods, always distribute workload dynamically into *k* parts with different sizes based on machine identification number, programmer setting, and the number of register elements in the SIMD. However, dynamicity is always a source of waiting time, which is for long sequences are unreasonable and impractical. Finally, stages of distribution method assign workload in every stage according to periodic processing progress notifications or in super-master and producer model. Synchronization cost in these methods is always higher than others, this may cause delays in results as well as communication complexity for such methods.

All the above reviews in this paper suffer from the fact that they are ignoring or overlooking the load balancing techniques, CPU complications in waiting-queue, space complexity, and data-flow technique between processor with synchronization and communication complexity. There is remains a need for an efficient method to harness the huge power obtainable by parallel platforms.

REFERENCES

[1] 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.

[2] 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.

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

[4] W. R. Pearson and D. J. Lipman, "Improved tools for biological sequence comparison," *Proceedings of the National Academy of Sciences*, vol. 85, p. 2444, 1988.

[5] S. F. Altschul, W. Gish, W. Miller, E. W. Myers, and D. J. Lipman, "Basic local alignment search tool," *Journal of molecular biology*, vol. 215, pp. 403-410, 1990.

[6] M. Elnour, M. Latif, and I. Isnin, "Coarse Grain Load Balance Algorithm for Detecting Similar Regions in DNA and Proteins Sequences," *Indian Journal of Science and Technology*, vol. 7, pp. 47-57, 2014.

[7] D. J. Lipman, J. Ostell, and E. W. Sayers, "Dennis A. Benson, Mark Cavanaugh, Karen Clark, Ilene Karsch-Mizrachi," *Nucleic Acids Research*, vol. 1, p. 7, 2012.

[8] N. Kaur, "GenBank: Implementation Model and Retrieval Techniques," 2013.

[9] S. Velankar, C. Best, B. Beuth, C. Boutselakis, N. Cobley, A. W. S. Da Silva, *et al.*, "PDBe: protein data bank in Europe," *Nucleic Acids Research*, vol. 38, pp. D308-D317, 2010.

[10] Protein Data Bank in Europe. (2014, 4 Jan). *BDPe*. Available: http://www.ebi.ac.uk/pdbe/

[11] M. Y. Galperin and X. M. Fernandez-Suarez, "The 2012 Nucleic Acids Research Database Issue and the online Molecular Biology Database Collection," *Nucleic Acids Res*, vol. 40, pp. D1-8, Jan 2012.

[12] M. E. ELTAYEEB, M. S. A. LATIF, and I. F. ISNIN, "PARALLEL MODEL AND SCHEDULING TECHNIQUE FOR SPACES COMPLEXITY AND SYNCHRONIZATION PROBLEMS IN SEQUENCES ALIGNMENT," *Journal of Theoretical & Applied Information Technology*, vol. 63, pp. 251-260, 2014.

[13] C. Sheel, M. I. Khan, M. I. H. Sarker, and T. Alam, "Algorithm for Optimal Storage of a Distributed Bioinformatics System for Analysis of DNA Sequences," *Int J Comput Bioinfo In Silico Model*, vol. 2, pp. 106-109, 2013.

[14] F. Yu and C. Coarfa, "Sequence Alignment, Analysis, and Bioinformatic Pipelines," in *Next Generation Sequencing*, ed: Springer, 2013, pp. 59-77.

[15] M. Swain, C. G. Silva, N. Loureiro-Ferreira, V. Ostropytskyy, J. Brito, O. Riche, *et al.*, "P-found: Grid-enabling distributed repositories of protein folding and unfolding simulations for data mining," *Future Generation Computer Systems-the International Journal of Grid Computing-Theory Methods and Applications*, vol. 26, pp. 424-433, Mar 2010.

[16] M. Ok, "An Auxiliary Storage Subsystem of Storage Space Hidden for User Data to Distributed Computing Systems," in *Web Information Systems Engineering–WISE 2011 and 2012 Workshops*, 2013, pp. 281-291.

[17] K. Samejima and T. Omori, "Adaptive internal state space construction method for reinforcement learning of a real-world agent," *Neural Networks*, vol. 12, pp. 1143-1155, 1999.

[18] J.-P. Martin, L. Alvisi, and M. Dahlin, "Minimal byzantine storage," in *Distributed Computing*, ed: Springer, 2002, pp. 311-325.

[19] G. Chockler, R. Guerraoui, I. Keidar, and M. Vukolic, "Reliable Distributed Storage," *Computer*, vol. 42, pp. 60-+, Apr 2009.

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

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

[22] 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.

[23] A. Wozniak, "Using video-oriented instructions to speed up sequence comparison," *Computer applications in the biosciences: CABIOS*, vol. 13, pp. 145-150, 1997.

[24] O. Gotoh, "An improved algorithm for matching biological sequences," *Journal of molecular biology*, vol. 162, pp. 705-708, 1982.

[25] A. Nordin, M. Yazid, A. Aziz, and M. Osman, "A guided dynamic programming approach for searching a set of similar DNA sequences," in *Applications of Digital Information and Web Technologies, 2009. ICADIWT'09. Second International Conference on the*, 2009, pp. 512-517.

[26] D. S. Hirschberg, "A linear space algorithm for computing maximal common subsequences," *Communications of the ACM*, vol. 18, pp. 341-343, 1975.

[27] M. O. Dayhoff and R. M. Schwartz, "A model of evolutionary change in proteins," in *In Atlas of protein sequence and structure*, 1978.

[28] 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.

[29] M. E. E. Elnour, M. S. A. Latif, and I. F. Isnin, "Validation Experimentations of Local Alignment Parameters for Comparing DNA and Protein Sequences," *IOSR Journal of Computer Engineering (IOSR-JCE)* vol. 16, pp. 56-61, 2014.

[30] A. Hudek and D. Brown, "FEAST: sensitive local alignment with multiple rates of evolution," *Computational Biology and Bioinformatics, IEEE/ACM Transactions on*, pp. 1-1, 2011.

[31] X. Zhang, X. Zhou, and X. Wang, "Basics for Bioinformatics," in *Basics of Bioinformatics*, ed: Springer, 2013, pp. 1-25.

[32] H. Li and N. Homer, "A survey of sequence alignment algorithms for next-generation sequencing," *Briefings in bioinformatics*, vol. 11, pp. 473-483, 2010.

[33] W. Haque, A. Aravind, and B. Reddy, "Pairwise sequence alignment algorithms: a survey," in *Proceedings of the 2009 conference on Information Science, Technology and Applications*, 2009, pp. 96-103.

[34] H. You, B. Rekapalli, Q. Liu, and S. Moore, "Autotuned Parallel I/O for Highly Scalable Biosequence Analysis," in *Proceedings of the 2011 TeraGrid Conference: Extreme Digital Discovery*, 2011, p. 29.

[35] S. Sathe and D. Shrimankar, "Parallelization of DNA sequence alignment using OpenMP," in *Proceedings of the 2011 International Conference on Communication, Computing & Security*, 2011, pp. 200-203.

[36] S. J. Tang, C. Yu, J. Z. Sun, B. S. Lee, T. Zhang, Z. Xu, *et al.*, "EasyPDP: An Efficient Parallel Dynamic Programming Runtime System for Computational Biology," *Ieee Transactions on Parallel and Distributed Systems*, vol. 23, pp. 862-872, May 2012.

[37] C. C. Wu, L. T. Huang, L. F. Lai, and M. L. Chen, "Enhanced parallel loop self-scheduling for heterogeneous multi-core cluster systems," in *Pervasive Systems, Algorithms, and Networks (ISPAN), 2009 10th International Symposium on*, 2009, pp. 568-573.

[38] 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.

[39] M. Nordin and A. Rahman, "Utilizing MPJ Express Software in Parallel DNA Sequence Alignment," in *International Conference on Future Computer and Communication, 2009, ICFCC, *, 2009, pp. 567-571.

[40] 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.

[41] E. W. Myers and W. Miller, "Optimal alignments in linear space," *Computer applications in the biosciences: CABIOS*, vol. 4, pp. 11-17, 1988.

[42] X. Huang, R. C. Hardison, and W. Miller, "A space-efficient algorithm for local similarities," *Computer applications in the biosciences: CABIOS*, vol. 6, pp. 373-381, 1990.

[43] C. Wu, A. Kalyanaraman, and W. R. Cannon, "A scalable parallel algorithm for large-scale protein sequence homology detection," in *Parallel Processing (ICPP), 2010 39th International Conference on*, 2010, pp. 333-342.

[44] 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.

[45] S. Steinfadt, "Fine-grained parallel implementations for SWAMP+ Smith–Waterman alignment," *Parallel Computing*, vol. 39, pp. 819-833, 2013.

[46] A. Nordin, M. Yazid, A. Aziz, and M. Osman, "Parallel Guided Dynamic Programming Approach for DNA Sequence Similarity Search," *International Journal of Computer and Electrical Engineering.*, vol. 1, pp. 402-409, 2009.

[47] A. Montanola, C. Roig, and P. Hernandez, "Pairwise sequence alignment method for distributed shared memory systems," in *Parallel, Distributed and Network-Based Processing (PDP), 2013 21st Euromicro International Conference on*, 2013, pp. 432-436.

[48] A. Hoisie, O. Lubeck, and H. Wasserman, "Performance analysis of wavefront algorithms on very-large scale distributed systems," in *Workshop on wide area networks and high performance computing*, 1999, pp. 171-187.

[49] F. Sánchez Castaño, A. Ramirez, and M. Valero, "Quantitative analysis of sequence alignment applications on multiprocessor architectures," in *Proceedings of the 6th ACM conference on Computing frontiers*, 2009, pp. 61-70.

[50] F. Sánchez, F. Cabarcas, A. Ramirez, and M. Valero, "Long DNA sequence comparison on multicore architectures," *Euro-Par 2010-Parallel Processing*, pp. 247-259, 2010.

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

[52] 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.

[53] 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.

[54] 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.

[55] 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.

[56] 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.

[57] 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.

[58] 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.

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

[60] NCBI. (2014, 15 Feb). *National Center for Biotechnology Information*. Available: http://www.ncbi.nlm.nih.gov/