Journal Homepage: - www.journalijar.com
INTERNATIONAL JOURNAL OF ADVANCED RESEARCH (IJAR)
Article DOI: 10.21474/IJAR01/2877
DOI URL: http://dx.doi.org/10.21474/IJAR01/2877

RESEARCH ARTICLE

ACCELERATION OF DISTANCE COMPUTATION FOR MULTIPLE SEQUENCE ALIGNMENT ON MULTI-CORE ARCHITECTURES

***Mohammed W. Al-Neama[1] and Kasim A. Al-Salem[2].**
1. Education College for Girls, University of Mosul, Mosul, IRAQ.
2. Department of Computer Science, University of Cihan/Sulaimanya, Kurdstan, Iraq.

……………………………………………………………………………………………………....

Manuscript Info

Manuscript History

Received: 21 November 2016
Final Accepted: 21 December 2016
Published: January 2017

Abstract

Although high quality multiple sequence alignment is an essential task in bioinformatics, it becomes a big dilemma nowadays due to the gigantic explosion in the amount of molecular data. The most consuming time and space phase is the distance matrix computation. This paper addresses this issue by proposing a vectorized parallel method that accomplishes the huge number of similarity comparisons faster in linear space.

……………………………………………………………………………………………………....

Introduction:-

Multiple sequence alignment of several nucleotides or amino acids is an important tool in bioinformatics. It can identify patterns or motifs to characterize protein families, and is therefore utilized to detect homology between sequences as well as to perform phylogenetic analysis [1].

It is playing an increasingly important role in diverse areas, such as elucidation of the tree of life [2], studies of epidemiology and virulence [3], drug design [4], and human genetics [5]. Most popular MSA tools, ClustalW[6], T-Coffee[7], MAFFT[8], and DIALIGN[9], utilize the progressive method that was at first introduced in [10].

It typically consists of three stages. Stage 1 computes a Distance Matrix (DM) comprised of the distance value between each pair of input sequences. Stage 2 computes an evolutionary tree from the DM using some phylogeny reconstruction methods like Neighbor-Joining (NJ) [11] which guides the final multiple alignment process. In stage 3, first closely related sequence or group of sequences is aligned then the most divergent sequences are aligned to get the final MSA.

However, there are some obstacles that must be handled carefully when using the progressive method. First, complexity is of increasing relevance due to the rapid growth of sequence databases, which now contains enough representatives of larger protein families to exceed the capacity of most current programs.

For example, aligning two sequences with one megabyte length each requires several terabytes of memory, which cannot be provided by most of the commodity computational resources. Second, computational load of multiple alignment calculations is of great increasing.

For example, computations of modern homologous sequence data sets could take days. In fact, the best methods sometimes fail to deal with these complexities efficiently and obtain biologically accurate alignments at the same time. The present study overcomes these obstacles by using two main approaches.

Corresponding Author:- Mohammed W. Al-Neama.
Address:- Education College for Girls, University of Mosul, Mosul, IRAQ.

The first is the vectorization, where all matrices are compensated by vectors, which in turn reduces the memory requirement and speedup execution without affecting the accuracy.

The second approach is parallelism, the widespread programming method nowadays that allows multiple independent processes which share the same resources, to be executed concurrently at less time. Thus, this work proposes an optimized method for the progressive MSA distance matrix computation using vectorization and parallelism.

It aims at producing a superlative MSA tool over existing ones in space and execution time. Implementation tests uses MATLAB®2012 compiler and explicit parallel programming with the fork-join model of parallel execution on core i7.

MATLAB was the most appropriate programming language for our work because it is an interpreted language of a high-level scripting and interactive sessions. It tends to be easier to code and debug. Its package comes with sophisticated libraries for matrix and vector operations, general numeric methods and plotting of data. It supports parallelism and MEX-files including C++ codes that accelerate execution and offer full control over parallelization [12]. The multi-core was the candidate platform for implementation due to its availability, but it is intended to extend our work for clusters; grids; or clouds. A multi-core processor uses the shared memory storage mechanism. The relationship between its cores is tightly coupled, and they are often interconnected by shared-cache, therefore, there is almost no communication overhead between cores [13].

Related Work:-

This section surveys the most popular parallel tools developed for MSA using multi-cores. It highlights their parallel techniques and their performance enhancement. ClustalW presented a fully multithreading optimized version called MT-ClustalW [14]. It utilized the machine resources and achieves higher throughput. It was 2 times faster than the sequential ClustalW using 8 threads. While on Cell BE [15], it makes extensive use of vectorization and schedules the application across all cores which speed up the pairwise alignment phase. In addition, it applies loop unrolling and loop skewing optimizations that speedup the progressive alignment phase. It achieves an overall speedup of 9.1. Also on a QS21 Cell Blade, it demonstrates a speedup of 24.4 times when using 16 synergistic processor units compared to single-thread execution on the power processing unit, and 3.8 times faster than a 3-thread version running on an Intel Core2 Duo[16]. Cloud-Coffee [17] is the parallel implementation of T-Coffee that is based on shared-memory architectures, like multi-core. It was benchmarked on the Amazon Elastic Cloud (EC2) and runs 3.7 times faster. In [18], all stages of MAFEET have been parallelized using the POSIX Threads library with the best-first and simple hill-climbing parallelization strategies. It achieved a speedup of 10 times with different random numbers on a 16 core PC. DIALIGN-TX-MPI [19] is the parallel version of DIALIGN-TX that was implemented using both OpenMP and MPI on a heterogeneous multi-core cluster. It used an iterative heuristic method for MSA that is based on dynamic programming and generates alignments by concatenating ungapped regions with high similarity. It obtains a speedup of 3.13. MSAProbs [20] combines a pair-HMM and a partition function to calculate posterior probabilities. It investigates weighted probabilistic consistency transformation and weighted profile- profile alignment, to achieve high alignment accuracy. In addition, it is optimized for modern multi-core CPUs by employing a multi-threaded design in order to reduce execution time. It statistically demonstrates dramatic accuracy improvements over previous tools. MSACompro [21] incorporates predicted secondary structure, relative solvent accessibility, and residue-residue contact information into the currently most accurate posterior probability-based MSA methods. It uses a multiple-threading implementation on a 32 CPU cores machine. Benchmarks clearly show improvements in accuracy over MSAProbs and all leading tools. We concluded from the study of the above tools that MSACompro and MSAProbs are the most accurate but at expense of speed. Sample-Align-D is the fastest but not available. Clustal and MAFFT are fast, available, portable and can align huge number of sequences but less accurate. MUSCLE and T-coffee provide a good compromise between time spent and quality of the resulting alignment. And most of them exhaust large storage space due to the usage of matrices. Thus this research aims at addressing the problems concerning space and time.

Proposed Approach:-

Some attempts have been made to accelerate DM computations using GPU's [22], and CUDA [23]. This section explains our proposed algorithm. It depends mainly on the espoused idea in this work of switching from matrices to vectors. Its main goal is to speedup computations and reduces storage. The baseline equation used to compute the elements of the distance matrix DM(N×N) for aligning N sequences $\{S_1, S_2, \ldots, S_N\}$ is:

$$DM(i,j) = 1 - nid(S_i,S_j) / \min(L_i,L_j) \quad \forall\ 1 \le i, j \le N \qquad (1)$$

where $nid(S_i,S_j)$ is the similarity score between Si and Sj. It is computed by using the most popular optimal local alignment known as the Smith-Waterman algorithm (SM) [24]. It compares two sequences by computing a distance that represents the minimal cost of transforming one segment into another, with respect to the given scoring system. It identifies common subsequences between any two sequences S_1 and S_2 of length L_1 and L_2, by computing the similarity H(i,j) of two sequences ending at position i and j, using the following recurrence:

$$H(x,y) = \max \begin{cases} 0, \\ H(x-1,y-1) + \mathrm{sbt}(S_i(x), S_j(y)), \\ H(x-1,y) + g, \\ H(x,y-1) + g, \end{cases} \quad , \forall\ 1 \le i, j \le N \qquad (2)$$

where sbt is a nucleotides or amino acid substitution matrix, and g is a gap penalty.

The proposed algorithm vectorizes all above matrices. First, it benefits from the fact that computing any anti-diagonal in the matrix H is based only on the values of the previous two anti- diagonals. Based on Equation (2), each cell H(i,j) depends only on its Northern H(i,j-1), Western H(i-1,j) and North-Western H(i-1,j-1) previously computed. Thus, just one vector V for current anti- diagonal, with two buffers V1 and V2 for two previously computed anti-diagonals, are enough to compute the similarity score.

This is done by computing all cells along anti-diagonal V in parallel. The value of each cell is evaluated in terms of its diagonal neighbour stored at V1, with its left and upper neighbours stored at V2, and the maximum value is selected indicating the highest score, using the following equation repeatedly along all 2L-1 anti-diagonals, where it is assumed that all sequences have the same length L, for simplicity.

$$V^i(k) = \{H(k+1, i-k+1)\}_{k=2}^{i}$$

$$= \left\{ \begin{matrix} \max(0, H(k, i-k) + sbt(S_1(k), S_2(i-k+1)), \\ H(k, i-k+1) + g, H(k, i-k) + g) \end{matrix} \right\}_{k=2}^{i}$$

$$= \left\{ \begin{matrix} \max(0, V^{i-2}(k-1) + sbt(S_1(k-1), S_2(i-k)), \\ V^{i-1}(k-1) + g, V^{i-1}(k) + g) \end{matrix} \right\}_{k=2}^{i}$$

$$V^i(k) = max \left\{ \begin{matrix} 0, \\ V^i1(k-1) + sbt(S_1(k-1), S_2(i-k)), \\ V^i2(k-1) + g, \\ V^i2(k) + g \end{matrix} \right\} \qquad (3)$$

Fig. 1 illustrates the main idea when aligning the two sequences S1={ACCGTCG} and S2={TCCGTCA} of length 7. It shows the computation of the similarity matrix H, with the linear gap cost (-8) and a substitution cost of (5) if the characters are identical and (-4) otherwise, and how it is replaced by V, using V1 and V2.

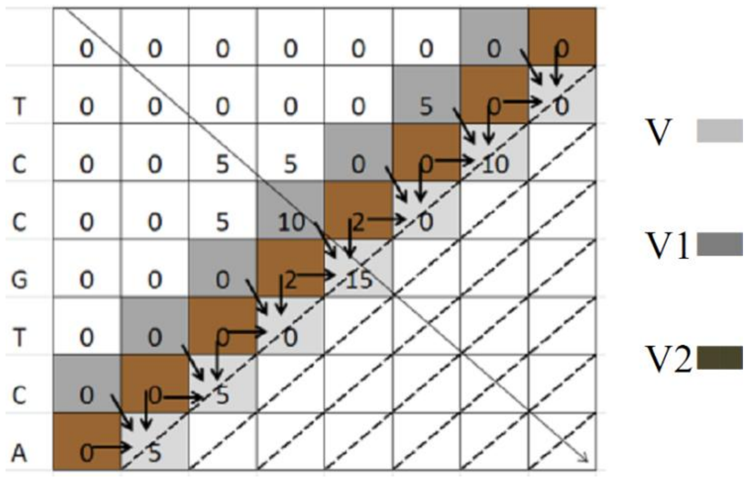

Fig. 1:- Relation between H, V, V1, V2.

Second, the proposed algorithm also replaces the distance matrix DM by a distance vector DV. It exploited the fact that DM is symmetric, and stores only its minor diagonal with its upper triangle, dispensing repeated values. It computed the values of DV by the following equation:

$$DV(k) = 1 - (dv_k)/\min(L) \quad (4)$$

Where (dv_k) is the number of similarity score in the optimal local alignment of Si and Sj. Fig. 2 shows the correspondence between DV and DM cells when N=5.

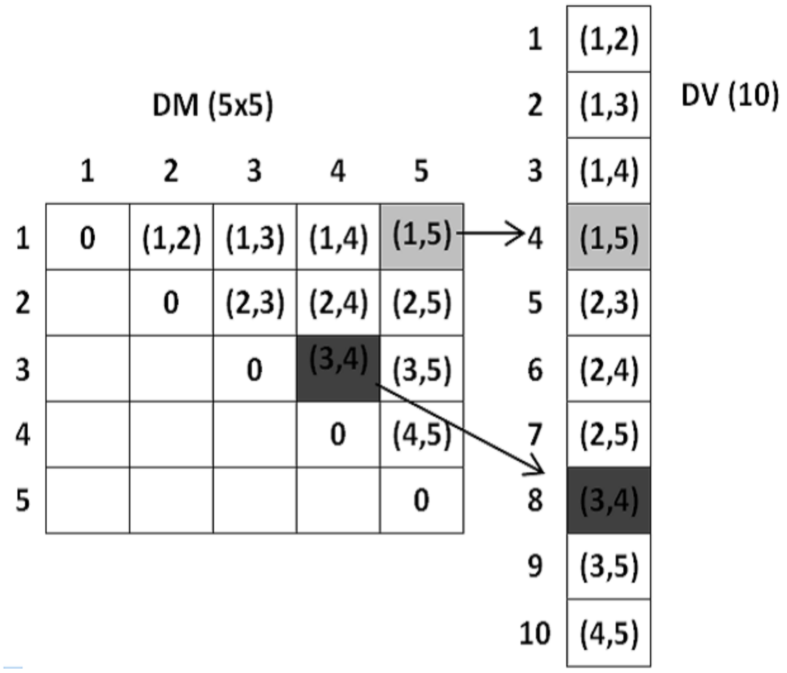

Fig. 2:- The relation between DM, DV.

Results and Discussion:-

This section presents results obtained when measuring the performance of the above proposed parallel method implemented using MATLAB® R2012a on a core-i7 processor with 8 cores of 3.4GHz and 8GB RAM running on Windows7. Evaluations were concerning two attributes of MSA tools that are of great importance to users. They are time and space.

Runtime:-
The most important criterion of measuring the quality of our tool is the consumed time during its execution. A set of performance evaluation experiments has been conducted using ten protein sequence datasets, consisting of sequences with different combinations of sequences' number (N) and length (L), selected from the Human Immunodeficiency Virus (HIV) dataset downloaded from NCBI. Table (1) presents the execution time and speedup of the proposed parallel algorithm for computing the distance vector DV, and the original algorithm computing the distance matrix DM. It shows that the greater the length of sequences, the more acceleration of DV due to the high computation speedup of the optimal exact matches.

Table 1:- Performance Comparison between DM, DV Computations

N	L	DM(sec.)	DV (sec.)	Speedup
400	856	760.812	141.245	5.386
400	408	213.575	48.063	4.444
600	462	595.460	142.750	4.171
800	454	1087.349	230.806	4.711
1000	858	4895.379	900.068	5.439
1000	446	1640.334	385.649	4.253
2000	266	2624.920	570.873	4.598
4000	247	22720.202	5579.069	4.072
4000	83	4989.181	1462.492	3.411
8000	73	16972.691	4733.101	3.586

Our work has achieved higher performance due to four reasons: (1) the superiority of MATLAB on other languages at dealing with vectors, (2) the optimal use of Multi-core machine when parallelizing the computation of DV independent elements, (3) the perfect use of C++ mix-file when dealing with memory, (4) the use of RAM only for storing the V's and H's.

Usage Space:-
Storage space is the second parameter of measuring the quality of the MSA tool because of the huge growth of sequence databases that exceed current programs' capacity. It is measured by the space needed to store data. Experiment results of used storage during computations of both DM and DV were recorded as given in Table (2), with respect to the size of the N input sequences. It is clear that the proposed algorithm has reduced the overall space almost to the half. This is because the space required for the matrix DM whose size is $N \times N$ has been reduced to DV of size $N \times (N-1)/2$.

Table 2:- Storage Comparisons between DM, DV

N	L	DM(Mbyte)	DV(Mbyte)
400	408	0.64	0.31
400	856	0.64	0.31
600	462	1.44	0.70
800	454	2.56	1.25
1000	446	4	1.95
1000	858	4	1.95
2000	266	16	7.80
4000	247	64	31.19
4000	83	64	31.19
8000	73	256	124.78

In addition, Table (3) shows the RAM storage exhausted when storing H's in comparison to that of V's. This remarkable achievement comes from the fact that SW algorithm consumes $(L+1)^2$ word to find similarity between two sequences of length L, while DV uses only $3(L+1)$ word.

Table 3:- Storage Comparisons between H's, V's

L	H's(Kbyte)	V's(Kbyte)
408	334.56	2.45
856	1468.90	5.14
462	428.74	2.78
454	414.05	2.73
446	399.62	2.68
858	1475.76	3.54
266	142.58	1.60
247	123.01	1.49
83	14.11	0.51
73	10.95	0.44

Conclusion and Future Work:-

The contribution of this work contain optimizations for SW algorithm, and DM computation for addressing the problem of building a parallel tool for multi-cores that produces the best alignment of multiple sequences in short time without using much storage space. Results prove that the proposed approach for DM and SW has good ability to aligning large number of sequences through powerful improved storage handling capabilities with efficient improvement of the overall processing time.

For future, it is planned to apply the same mechanism on NJ stage, and combine all algorithms to produce the aligner. Then the aligner will be extended to operate on different parallel platforms. Challenges expected to be tackled when merging optimization techniques for improving accuracy may affect performance improvements.

References:-

1. Albert Y. Zomaya, Parallel computing for bioinformatics and computational biology: models, enabling technologies, and case studies, John Wiley & Sons Inc., 2006.
2. CW. Dunn, A. Hejnol, DQ. Matus, K. Pang, WE. Browne, et al., Broad phylogenomic sampling improves resolution of the animal tree of life, Nature, vol. 452, , 2008 pp. 745–749.
3. Y. Bao, P. Bolotov, D. Dernovoy, B. Kiryutin, L. Zaslavsky, et al., The influenza virus resource at the National Center for Biotechnology Information, J. Virol., vol. 82, 2008, pp. 596–601.
4. RK. Kuipers, HJ. Joosten, WJ. van Berkel, NG. Leferink, E. Rooijen, et al., 3DM: systematic analysis of heterogeneous superfamily data to discover protein functionalities, Proteins, vol. 78, 2010, pp. 2101–2113.
5. S. Singh, R. Tokhunts, V. Baubet, JA. Goetz, ZJ. Huang, et al., Sonic hedgehog mutations identified in holoprosencephaly patients can act in a dominant negative manner, Hum. Genet., vol. 125, 2009, pp. 95–103.
6. J.D. Thompson, D.G. Higgins, and T.J. Gibson, CLUSTAL W: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, positions-specific gap penalties and weight matrix choice, Nucleic Acids Res., vol. 22, No. 22, 1994, pp. 4673– 4680.
7. C. Notredame, D.G. Higgins, J. Heringa, T- Coffee: a novel method for fast and accurate multiple sequence alignment, J. Mol. Biol., vol. 302, No. 1, 2000, pp. 205–17.
8. K. Katoh, K. Misawa, K. Kuma, and T. Miyata, MAFFT: a novel method for rapid multiple sequence alignment based on fast Fourier transform, Nucleic Acids Res., vol. 30, No. 14, 2002, pp. 3059-3066.
9. B. Morgenstern, K. Frech, A. Dress and T. Werner, DIALIGN: Finding Local Similarities by Multiple Sequence Alignment, Bioinformatics, vol. 14, 1998, pp. 290-294.
10. D. Feng, R. Doolittle, Progressive sequence alignment as a prerequisite to correct phylogenetic trees, J. Mol. Evol., vol. 25, 1987, pp. 351–360.
11. Saitou N, Nei M, The neighbor-joining method: a new method for reconstructing phylogenetic trees, Mol. Biol. Evol., vol. 4, pp. 406-425, 1987.
12. Jan Urban., Interfacing C++ libraries to Matlab, MSc. Thesis, Universitas Masarykiana, 2012.
13. Xiaozhong Geng, A Task Scheduling Algorithm for Multi-Core Cluster Systems, JCP, Vol. 7, No. 11, 2012, pp. 2797-2804.
14. Kridsadakorn Chaichoompu and Surin Kittitornkun, Multithreaded ClustalW with improved optimization for Intel multi-core processor, ISCIT '06, 2006, pp. 590-594.
15. Hans Vandierendonck, Sean Rul, Michiel Questier et al., Experiences with parallelizing a bio-informatics program on the cell BE, HiPEAC'08, vol. 4917, 2008, pp. 161–175.

16. Hans Vandierendonck, Sean Rul, and Koen De Bosschere, Accelerating multiple sequence alignment with the Cell BE processor, Comput. J., vol. 53, No. 6, 2010, pp. 814–826.
17. P. Di Tommaso, M. Orobitg, F. Guirado, F. Cores, T. Espinosa, C. Notredame, Cloud- Coffee: implementation of a parallel consistency-based multiple alignment algorithm in the T-Coffee package and its benchmarking on the Amazon Elastic-Cloud, Bioinformatics, vol. 26, No. 15, 2010, pp. 1903-1904.
18. K. Katoh, and H. Toh, Parallelization of the MAFFT multiple sequence alignment program, Bioinformatics, vol. 26, No. 15, 2010, pp. 1899- 1900.
19. E. de Araujo Macedo, A.C. Magalhaes Alves de Melo, G.H. Pfitscher, A. Boukerche, Hybrid MPI/OpenMP Strategy for Biological Multiple Sequence Alignment with DIALIGN-TX in Heterogeneous Multicore Clusters, IPDPSW'11, IEEE Xplore Press, 2011, pp. 418- 425.
20. Yongchao Liu, Bertil Schmidt, Douglas L. Maskell, MSAProbs: multiple sequence alignment based on pair hidden Markov models and partition function posterior probabilities, Bioinformatics, vol. 26, No. 16, 2010, pp. 1958 -196.
21. X. Deng and J. Cheng, MSACompro: Protein Multiple Sequence Alignment Using Predicted Secondary Structure, Solvent Accessibility, and Residue-Residue Contacts, BMC Bioinformatics, vol. 12, 2011, pp. 472-488.
22. Zhi Ying, Xinhua Lin, Simon Chong-Wee and Minglu Li, GPU-Accelerated DNA Distance Matrix Computation, ChinaGrid'11, 2011, pp. 42-47.
23. Balaji Venkatachalam, Parallelizing the Smith-Waterman Local Alignment Algorithm using CUDA, 2012, http://www.zl50.com/ 2012061030871115.html.
24. T.F. Smith, M.S. Waterman, Identification of common molecular subsequences, J. Mol. Biol., vol. 147, 1981, pp. 195-197.