Asian Journal of Computer Science and Technology
ISSN: 2249-0701 Vol.7 No.2, 2018, pp. 70-74

Performance Analysis of Parallelized Bioinformatics Applications

Dhruv Chander Pant[1] and OP Gupta[2]
[1]Research Scholar, I. K. Gujral Punjab Technical University, Kapurthala, Punjab, India
[2]Associate Professor, Punjab Agricultural University, Ludhiana, Punjab, India
E-Mail: dpant9@gmail.com

Abstract **- The main challenges bioinformatics applications facing today are to manage, analyze and process a huge volume of genome data. This type of analysis and processing is very difficult using general purpose computer systems. So the need of distributed computing, cloud computing and high performance computing in bioinformatics applications arises. Now distributed computers, cloud computers and multi-core processors are available at very low cost to deal with bulk amount of genome data. Along with these technological developments in distributed computing, many efforts are being done by the scientists and bioinformaticians to parallelize and implement the algorithms to take the maximum advantage of the additional computational power. In this paper a few bioinformatics algorithms have been discussed. The parallelized implementations of these algorithms have been explained. The performance of these parallelized algorithms has been also analyzed. It has been also observed that in parallel implementations of the various bioinformatics algorithms, impact of communication subsystems with respect to the job sizes should also be analyzed.**
Keywords: **Applications, Bioinformatics, High Performance Computing, Parallel Computing**

I. INTRODUCTION

A. Bioinformatics

Bioinformatics is a swiftly unfolding multifaceted field where different techniques of computer science are applied to solve compute intensive biological problems. Due to technological augmentation, volume of molecular data is increasing expeditiously [1], [2]. With the enormous amounts of data, the challenge of bioinformatics is to store, manage, analyze and interpret the sequence data [3]. Generally the bioinformatics applications deal with the applications in the following areas:

1. To manage and process the huge volume of genome data.
2. To reduce data analysis time.

So break through technological development was needed to solve many critical problems with various bioinformatics applications [4], [15]. Such data management is impractical with the help of uni-processor computers. So the use of parallel computing in bioinformatics applications is important [5], [16]. Now to deal with bulk amount of genome data distributed computers, cloud computers and multi-core processors are also available at very low cost.

B. Parallel Computing Architectures

In parallel computing a problem is broken into discrete parts and instructions of different parts run on different CPUs concurrently as shown in Figure 1.

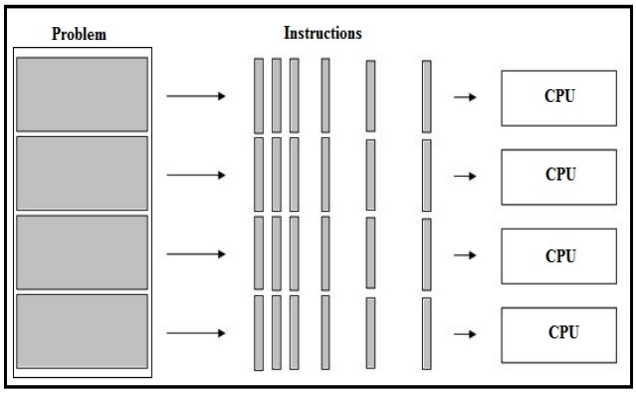

Fig. 1 Concept of Parallel Computation

In parallel computation a computing resource may be a single computer with multiprocessors, different number of computers connected by a network, multi core processors or the combination. And the problem should be able to be broken into different parts that can run simultaneously [6]. The various advantages of using parallel computing are:

1. Save time and/or money
2. Solve larger problems
3. Provide concurrency
4. Use of non local resources

Parallel computer systems can be classified into two main models: Single Instruction Multiple Data (SIMD) Systems and Multiple Instructions and Multiple Data (MIMD) Systems as shown in Figure 2. A SIMD system consists of multiple simple processors with small local memory. These processors use explicit communication to transfer data to each other. All the different processors should be strongly synchronized. Because of the complexity and inflexibility, SIMD systems are not used for very advanced applications.

MIMD systems are more suitable to bioinformatics applications. In MIMD machines each process executes completely independent of the other process asynchronously. MIMD systems are further classified on the bases of shared and distributed memory. A process

running in the shared memory system can access any local or remote memory of the system whereas a process running in distributed memory cannot. Shared memory systems have many advantages for bioinformatics applications. Design of parallel programs is simplified with a single address map. Different processes can also communicate without any time loss, because every CPU has direct access to memory. Whereas in distributed memory systems a time penalty is incurred for intercrosses communication because of the lack of a single address map for the memory.

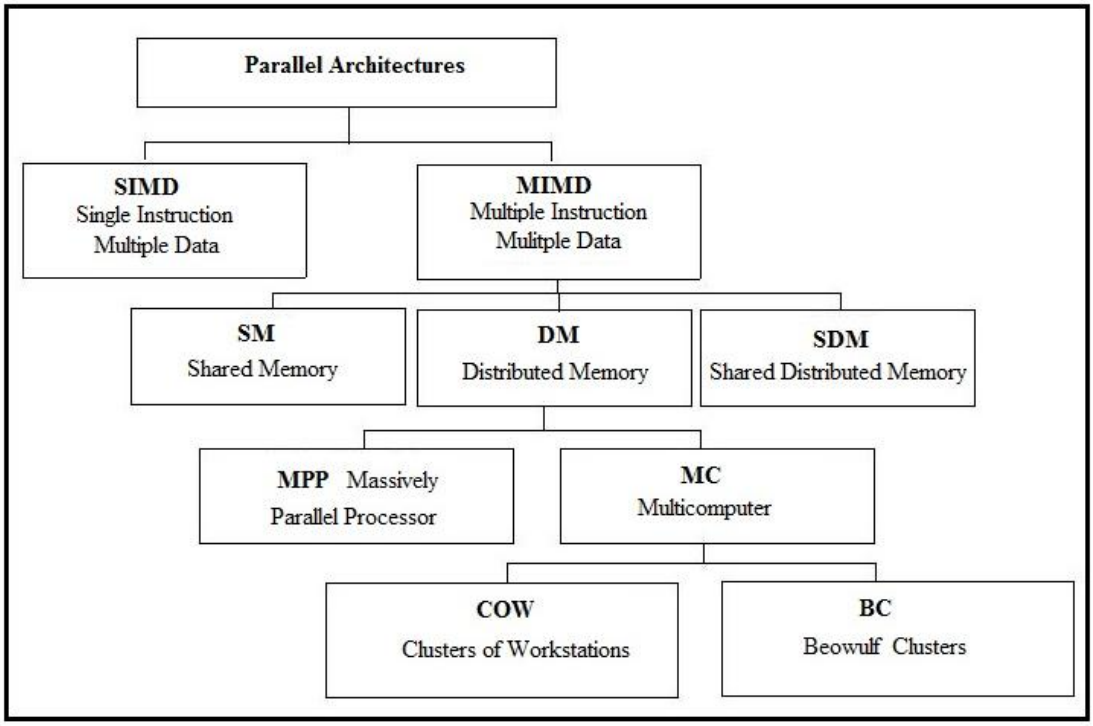

Fig. 2 Summarized Parallel Architectures

Current trends in multiprocessor design try to utilize the positive factors of both the architectures. Each CPU has some local memory attached to it and hardware creates an illusion of common memory shared by the whole system. So the memory installed in any node may be accessed by any other node with very low less time penalty. But as now very fast processors are available in the work stations, so microcomputers are connected with the help of Local Area Network. In this way virtual parallel computers are developed. These computers are also called Multi-computers which are constructed with the help of Cluster of Workstations (CoWs). One more architecture of multi – computers is Beowulf – clusters which consist of very simple hardware components like ordinary PCs. In this architecture a public domain server controls the whole cluster.

II PARALLELIZED IMPLEMENTATION OF DIFFERENT BIOINFORMATICS ALGORITHMS

A. Cluster Implementation of Sequence Alignment Algorithms

Smith waterman algorithm is used for local alignment between two sequences [7]. The algorithm is based on dynamic programming technique. If the two sequences of size *n* are to be matched then algorithm takes time *O (n*n).* As the value of *n* increases the time required becomes significantly high. Thus the need of parallel implementation of Smith waterman algorithm arises [8]. In smith waterman algorithm is implemented on clusters. The results of this cluster implementation are shown in Table 1.

In parallelization of the algorithm pipelining is used. In the score matrix, each row is computed sequentially and is blocked till the required cells in the above row are computed. When 32 processors are used with a sequence of 5000 characters long, the implementation showed an improvement up to 10.30 times. Smith waterman algorithm is also parallelized by its implementation on cell broadband engine [9]. In this implementation a static load balancing strategy is used. Under this strategy, work load at the beginning is divided equally among all the processors and processes. In the first step, algorithm reads the input dataset. In the next step the input sequences are processed by processing units to acquire the respective sequence parts in their local memories. For a sequence of 2048 characters long with this algorithm a speed up of 6.5 times is obtained. For multiple sequence matching multiple sequence alignment algorithms are used [10]. If there are n sequences, *n*(n-1)/2* pair-wise alignments need to be calculated. As the number of sequences increase, number of pair wise alignments also increase and the complexity of the algorithm also.

TABLE I PARALLEL IMPLEMENTATION OF SEQUENCE ALIGNMENT ALGORITHMS ON CLUSTER

Job Size (Bytes)	Sequential Algorithm	Parallel algorithm, np cores					
		4	8	16	32	64	128
500	0.24	0.3	0.2	0.1	0.1	0.5	1.3
1000	1.7	2.7	1.5	0.9	0.6	1.1	1.5
1500	5.9	8.8	4.8	2.9	1.8	1.7	2.1
2000	13.9	20.3	10	6.3	3.7	3.2	3.4
2500	26.2	39.5	21	11.6	6.9	5.1	4.5
3000	45.5	67.2	35.4	19.5	11.4	8.1	6.5
3500	71.6	106	55	30.1	17	11	8.9
4000	107.2	158	82	44.2	25	16	12
4500	152	225	118	62.4	34	21	15
5000	208	310	158	86.4	46	28	20

Once the distance matrix is calculated, in the next phase of the algorithm phylogenetic tree is produced. And in the final phase of the algorithm, previously generated phylogenetic tree is used to determine the order of the alignment. Experiments were performed with a number of techniques and concluded that to distribute all the n sequences to each processor was a better method. In this technique each of the *P* processors performs exactly *n*(n-1)/2P* alignments. Although this method has very high communication cost, even then it showed maximum speed up. For n = 500 sequences, where each sequence had 200 characters this technique showed a speedup of 5.81 times [11]. The result of implementation on cluster is shown in Figure 3.

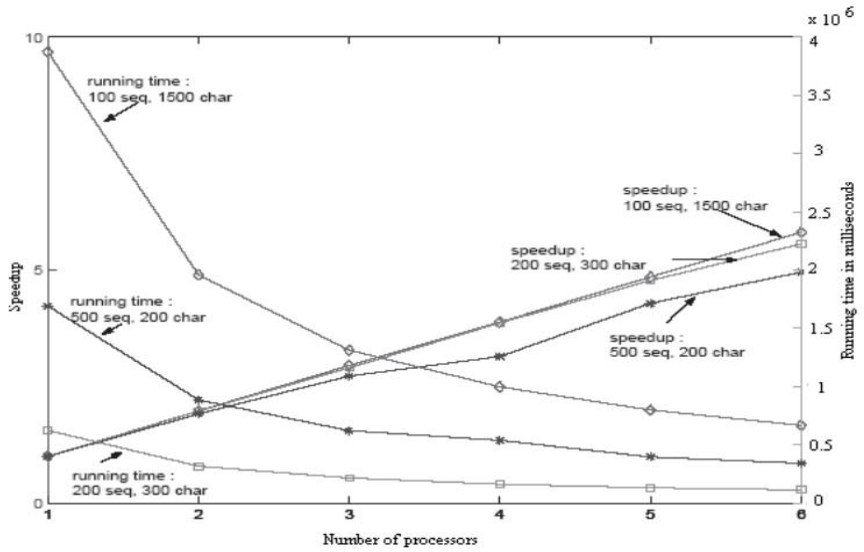

Fig. 3 Running Time and Speed ups for Parallel Implementation of Clustlaw

Parallel multiple sequence alignment was also performed on the cell broadband engine [12] where the parallel portions of the code were executed on synergistic processing units whereas sequential code on power processing units. For n= 8 pair of sequences where each sequence had 2048 characters showed a speed up of 46.37 x times.

B. Cell Implementation of Sequence Alignment Algorithms

Cell broadband engine based implementation for global alignment was performed on IBM Cell SDK 3.0 to obtain the results the implementation was executed on Sony Play Station 3 (SP3) and was compiled with optimized level -O3. The performance of this implementation was studied on different number of SPUs. The result of the implementation is shown in Figure 4 by using up to 6 synergistic processing units (SPUs). When this implementation is compared with a sequential implementation on a desktop with 3.2 GHz Pentium 4 Processor, a speed up of 6.5 xs is obtained. When this implementation was compared with best sequential algorithm with single SPU and a Pentium 4 Processor the speed ups were 4.5x and 3.5x respectively [11].

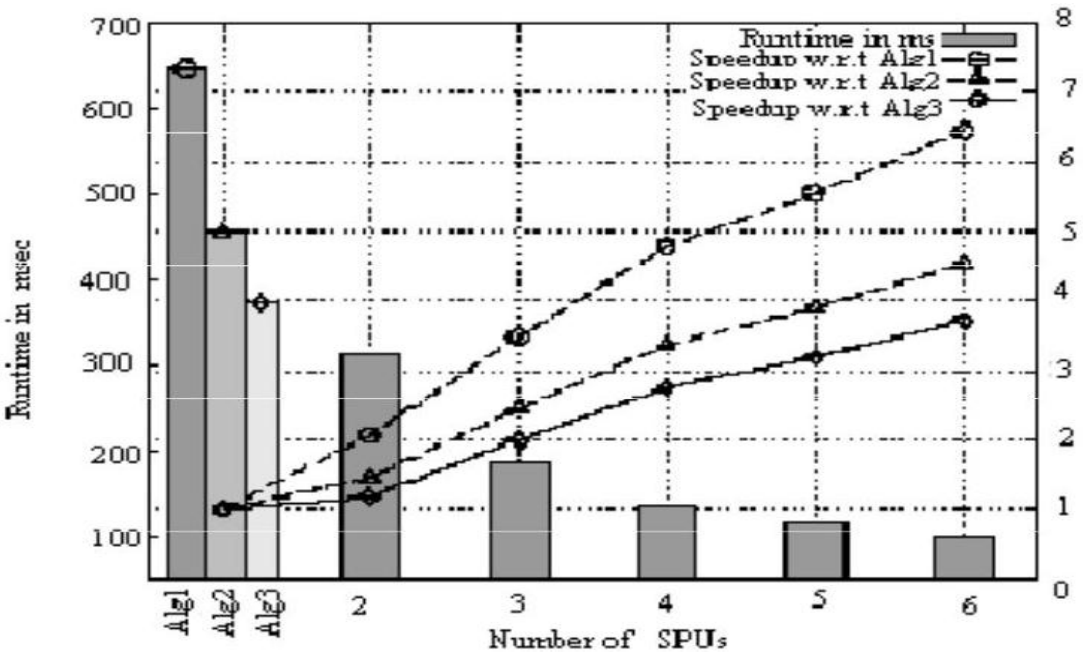

Fig. 4 Global Alignment for input size of 2048X2048

FASTA is a multiple sequence alignment algorithm. It is used to compute pair of match and mismatch between the sequences. Then this computation is used to detect the similarity between the sequences.

C. Cell Broadband Engine Implementation of FASTA for Multiple Sequence Alignment

AltiVec application-programming interfaces, already there in the Fasta package, was converted to the synergistic processing unit application programming interfaces to run Fasta on cell broadband engine. The interfaces which are not converted are implemented with the help of multiple instructions i.e. Vec max and Vec subs [11]. Vec max application programming interfaces is used to determine the maximum of two vectors. Result is stored in the output vector. From the stored result, synergistic processing unit find the greater vector. Vec subs application programming interfaces are used to perform saturated subtraction. In saturated subtraction any element with negative value is set to zero. In smith waterman a positive value of each cell is needed, so this application programming interface is helpful in the execution of smith water man.

Once the alignment scores are calculated with the help of power processing units, scores, query and library sequences are delivered to synergistic processing unit to execute the smith waterman kernel. But this cell implementation is limited by the size of the sequence. A sequence of more than 2048 characters cannot be compared in this implementation because of the size of the synergistic processing unit local memory. This problem can be rectified with the help of pipeline approach. Once smith waterman is implemented on the cell, then it can be used in FASTA package. In FASTA each query sequence is compared with every sequence in the database. Hence balancing load between each pair of sequences is evaluated.

D. Protein Structure Prediction Algorithms

The most important application of protein structure prediction is drug design. In protein structure prediction tertiary structure of the proteins is predicted from its amino acid sequences. On the bases of physical properties many protein structures are possible. So it is very difficult to understand the stability of a structure.

Genetic algorithm is used to implement protein structure prediction on computational grid [13]. Cell broadband engine implementation of protein structure prediction is also done [14]. In this implementation sequences are shifted from database to synergistic processing units with this implementation a speed up between 3.2 x and 3.6 x was achieved.

III. CONCLUSIONS

It is concluded that Parallel Computing is having very good impact on computational and data intensive applications. The processing time of bioinformatics algorithms can be improved by parallelization. When the jobs are parallelized and executed in a distributed computing environment, communication sub-systems also play a major role and contribute in processing time. So, impact of communication sub-systems also need to be analyzed in parallelized bioinformatics applications.

IV. ACKNOWLEDGEMENT

The authors express deep gratitude to the Dean, Research, Innovation and Consultancy Department of I.K.G. Punjab Technical University, Kapurthala, for giving them the opportunity to carry on this research work.

REFERENCES

[1] D. Jawadat, "Era of Bioinformatics", in *Proceedings of 2nd IEEE international conference on Information and Communication Technologies: From Theory to Applications*, pp 18060-1865, 2006.

[2] R. Hughey and K. Karplus, "Bioinformatics: A New Field in Engineering Education" in *Proceedings of 31st ASEE/IEEE Frontiers in Education Conference*, pp 15-17, 2001.

[3] O.P. Gupta and S. Rani, "Bioinformatics applications and Tools: An Overview", *CiiT- International Journal of Biometrics and bioinformatics*, Vol 3, No 3, pp. 107-110, 2010.

[4] I. Gorton, P. Greenfield, A. Sazalay and R. Williams, "Data Intensive Computations in 21st Century", in *Computer Magazine of IEEE Computer Society*, Vol. 41, No. 4, pp. 30 -32, 2008.

[5] C. Mueller, M. Dalkilic and A. Lumsdaine, "Implementing Data Parallel algorithms for Bioinformatics", in *proceedings of SIAM Conference on Computational Science and Engineering*, pp 226-232, 2005.

[6] K. Hwang and Z. Xu, "Scalable Parallel Computing: Technology, Architecture and Computing", *Mc-GrawHill Series in Computer Engineering*, 1998.

[7] T.F. Smith and M.S. Waterman, "Identification of Common Molecular Subsequences", *Journal of Molecular Biology*, Vol. 147, No. 1, pp. 195-197, 1981.

[8] Y. Chen, S. Yu and M. Leng, "Parallel Sequence Alignment Algorithms for Clustering System", *International Federation for Information Processing*, Vol. 207, pp. 311-321, 2006.

[9] A. Wirawan, K.C. Keong and B. Schmidt, "Parallel DNA Sequence Alignment on Cell Broadband Engine", *Springer – Verlag Berlin Heidelber*, pp. 1249– 1256, 2008.

[10] J. Ebedes and A. Datta, " Multiple Sequence Alignment in Parallel on a Workstation Cluster", Oxford University Press, Vol. 20, No. 77, pp. 1193-1195,2004.

[11] B.K. Pandey, S.K. Pandey and D. Pandey, "A Survey of Bioinformatics Applications on Parallel Architectures", *International journal of Computer Applications*, Vol. 23, No. 4, pp. 21 – 25, 2011.

[12] V. Sachdeva, M. Kistler, E. Speight and T.H.K. Tzeng, "Exploring the Viability of Cell Broadband Engine for Bioinformatics applications", in *Proceedings of IEEE International Parallel and Distributed Processing Symposium*, pp. 1-8, 2007.

[13] G. Minervini, G.L. Rocca, P.L. Luisi and F. Polticelli, "High Throughput Protein Structure Prediction in a Grid Environment", *Journal of Bio- Algorithms and Med System*, Vol. 3, No. 5, pp. 39-43, 2007.

[14] H. Zhang, B. Schmidt and W.M. Witting, " Accelerating BLASTP on the Cell Broadband Engine", in *Proceedings of the 3rd International Conference on Pattern Recognition in Bioinformatics*, pp. 46 – 47, 2008.

[15] S. Rani and O.P. Gupta, "CLUS_GPU-BLASTP- accelerated protein sequence alignment using GPU- enabled cluster", *Journal of Supercomputing*, Vol. 73, No. 10, pp. 4580-4595, 2017.

[16] M. Al-Rajab and J. Lu, "Bioinformatics: an overview for cancer research", *Proc. 13th International Conference on Bioinformatics and computational Biology, the University of Georgia, USA*, pp. 123-128, 2012.