A Survey of Bioinformatics Applications on Parallel Architectures

Binay Kumar Pandey
Information Technology Deptt
Govind Ballabh Pant University
of Agriculture and Technology
Pantnagar, Pantnagar, India

Sanjay Kumar Pandey
E.M.E Deptt,
Indian Army

Digvijay Pandey
Electronics Engineering Deptt
Institute of Engineering and
Rural Technology,
Allahabad, India

ABSTRACT
Based on bioinformatics algorithm, there are a wide range of implementations. With the urge for program speed, many applications take the heuristic approach to compensate running time. One of the most critical shortcomings of this technique is the loss of optimality, i.e. the desired results may not always be found. To overcome this problem, many different hardware architectures have been experimented for bioinformatics algorithm such as cell broadband engine, cluster and compute unified device architecture where, the main technique for obtaining high performance is to parallelize the task to be run simultaneously by multiple vector execution units with single instruction multiple data and by multiple processors with multiple instruction multiple data. In this paper we presents a survey of data intensive bioinformatics applications on variety of parallel Architecture that are available for accelerating the processing of large biological data set.

General Terms
Bioinformatics, Algorithms

Keywords
Cell broadband engine; Clusters; CUDA Suffix tree, Weighted suffix tree (key words)

1. INTRODUCTION
Bioinformatics [1] is considering as a branch that use computers to store and perform analysis on molecular data. With this digital format data, bioinformatics applications are able to solve the various existing problems of molecular biology [2] such as structures prediction [3], and simulation of macromolecules [4]. In the starting of the 21st century, researchers focus on entire species genomes sequencing [5] and also storing them in database that helps to know evolutionary changes in a species [6].
A very close examination of genome helps to know how it varies with time [7]. The most well-known and common applications of bioinformatics are sequence analysis [8] and phylogentic analysis [9]. In sequence analysis, nucleotide sequences of various species are put in databases for easy retrieval and comparison. The well-famous Human Genome Project [10] is a well known example of sequence analysis. Using large amount of high performance computers [11] and various techniques of collecting sequences, the whole human genome was sequenced and stored within a structured database [12].

Nucleotide or protein sequences used for bioinformatics may be collected in different ways. One technique is to go through a whole genome and search individual sequences to record and store it. Another method is to simply collect large amounts of fragments and perform comparisons on them, and then determining whole sequences by overlapping the unused fragments. The latter method discussed is known as shotgun sequencing [13], and is presently the most popular method because of its speed and the ease of use. By comparison of known sequences of a genome to specific changes, much information can be obtained about undesirable changes and diseases such as cancers. With the whole sequencing of the human genome, bioinformatics has played very important role in the research of cancers.
The collected information regarding genomes may be used for a different number of other applications, such as determining changes in populations and biomes. There are also many other applications of bioinformatics, including predicting entire protein strands, learning how genes express themselves in various species, and building complex models of entire cells. Fundamentally, bioinformatics applications face the following two major challenges [14]

- Managing and processing exponentially growing data volumes
- Significantly reducing data analysis cycles so that researchers can make timely decisions

The breakthrough technologies needed to address many of the critical problems in bioinformatics will come from collaborative efforts involving several disciplines, including computer science, engineering, and mathematics, and the domain of the problem to be solved. Some of the advances in data structures and hardware help to solve the problems faced by bioinformatics applications [14] are:

- Advances in high-performance computing platforms to provide high computational capability and uniform high-speed memory access to multi-terabyte data structures
- Specialized hybrid interconnect architectures to process and filter multi-gigabyte data streams coming from high-speed networks and scientific instruments and simulations, Many bioinformatics applications are data-path-oriented, making little use of branch prediction and speculation hardware in the CPU. These applications are well suited to streaming data access and cannot effectively use the sophisticated on-chip cache hierarchy. Their ability to process large data sets is hampered by orders-of-magnitude mismatches between disk, memory, and CPU bandwidths.

Emerging technologies can improve data-intensive algorithms performance, at reasonable cost in development time, by an order of magnitude over the state of the art processors. Coprocessors such as graphics processor units and field-programmable gate arrays can significantly speed up some application classes in which

data-path-oriented computing is dominant. Additionally, these coprocessors interact with application controlled on-chip memory rather than a traditional cache.

2. PARALLEL IMPLEMENTATION OF SEQUENCE ALIGNMENT ALGORITHM ON CLUSTER

Smith waterman algorithm [16] is used as a first step in sequence matching. The algorithm is used for local alignment between two sequences. The algorithm uses dynamic programming techniques to perform the alignment task. For matching two sequence of size *n*, the algorithm takes $O(n^2)$. For large value of *n*, the time becomes significant and thus the need arises for faster implementation. [17] implements smith waterman algorithm on clusters and the results of implementation are shown in table 1.. The parallelization is based on pipeline model, that is, as a basic unit, each row of the score matrix is computed sequentially by a processor, which block itself till the required elements in the row above are computed.

This algorithm shows the improvement upto 10.30 times for a sequence size of 5000 characters and number of processor is 32. Smith waterman algorithm was also implemented on cell broadband engine [18]. Their parallel algorithm employs a static load balancing strategy, which means that the work load is known at the start and distributed equally across processes and processors. The algorithm starts by reading the input dataset. The power processing unit then pre-processes the set of input sequences such that all the synergistic processing elements will have their respective sequence parts in their local memory. They obtained a speedup of upto 6.5 times for sequence length of 2048 characters.
Multiple sequence alignment algorithms [21] are used for sequencing matching of multiple sequences at a time. For *n* sequence, *n*(n-1)/2* pair-wise alignment must be calculated and this is computationally intensive task as *n* could be large as well as the size of input sequences can also be significant. Once the distance matrix is calculated, it is used in the next phase of the algorithm to produce a phylogenetic or guide tree that determines the order of alignments in the final phase of the algorithms. After experimenting with a number of schemes, the authors found that broadcasting all the *n* sequences to each processor was a better strategy. Each of the *P* processors performs exactly *n*(n-1)/2P* alignments. They got maximum speed up from this strategy despite its heavy communication cost. The cluster implementation showed a speedup of 5.81 times for *n=500* sequences and length of each sequence = *200* characters in each sequence. The result of implementation on cluster is shown in "Fig. 1".

Parallel multiple sequence alignment was also done on the cell broadband engine [15]. The algorithm focused on running parallel portion of code on the synergistic processing units, with the rest of the code executing on the power processing units. While pair-align itself is made up of 4 different functions, forward pass which computes the maximum score and the location of the cell inside the matrix cell for two sequences, is the most time-consuming step of pair-align. The cell broadband engine implementation showed a speedup of 46.37x times for *n*= 8 pair of sequences and length of each sequence = 2048 characters long which is shown in "Fig. 2".

Table 1 Parallel implementation of sequence alignment algorithms on clusters.

Sequence length	Sequential Algorithm	Parallel Algorithm, np Processor					
		np=1	np=2	np=4	np=8	np=16	np=32
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

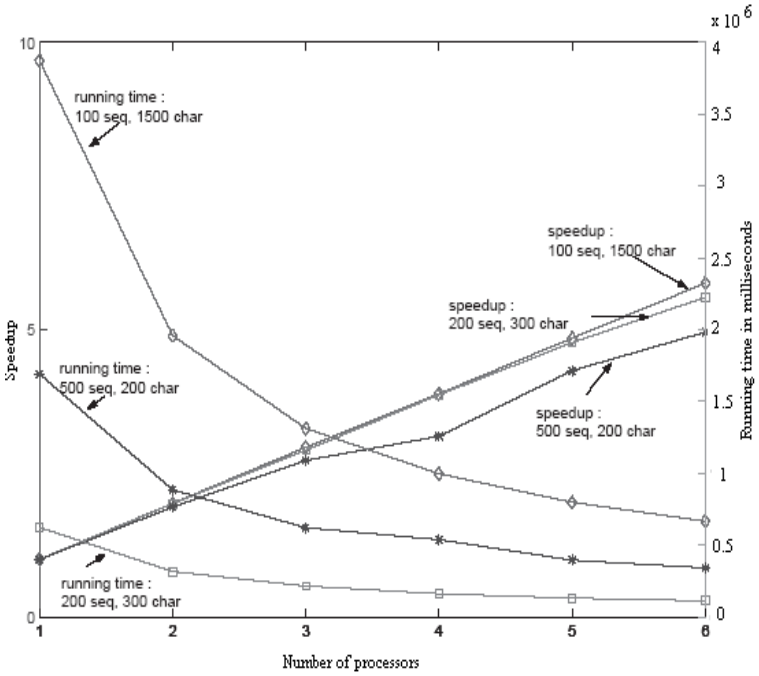

Fig .1 Running times and speedups for parallel implementation of ClustalW.

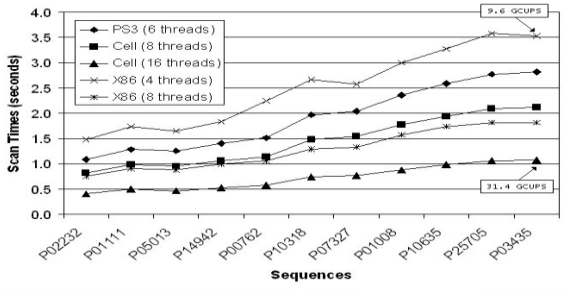

Fig 2. Performance of sequence search algorithms on Cell [This Figure taken from http://farrar.michael.googlepages.com/CellTimeBL50.GIF/CellTimeBL50- full;init:GIF]

3. PARALLEL IMPLEMENTATION OF SEQUENCE ALIGNMENT ALGORITHM ON CELL

In [12] author implements the global alignment and spliced alignment on cell broadband engine. The implementation was performed on IBM Cell SDK 3.0, and executed on the Sony Playstation 3 (PS3) to get the performance results. In this implementation author use the PS3 running Linux that use only six SPUs instead of total eight SPUs that are actually available on the cell broadband engine. The implementation results of PS3 and cell simulator was also given to analyze the speedups and scaling for eight SPUs. The implementation on cell broadband engine was compiled with optimization level *−O3*. The system-sim simulator tool spu-timing was very helpful to perform static analysis of the implementation of algorithm and it also help in optimizing the code for further improvement in its execution time. The performance of implementation was analyzed by varying number of SPUs.

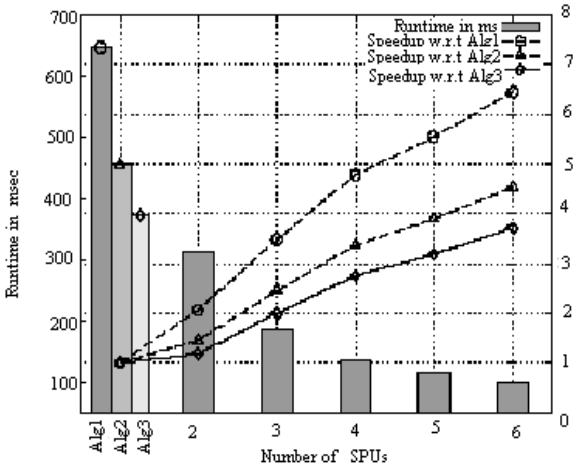

Fig 3. Global Alignment – This graph shows the runtimes and speedup of global alignment implementation for an input size of 2048×2048.

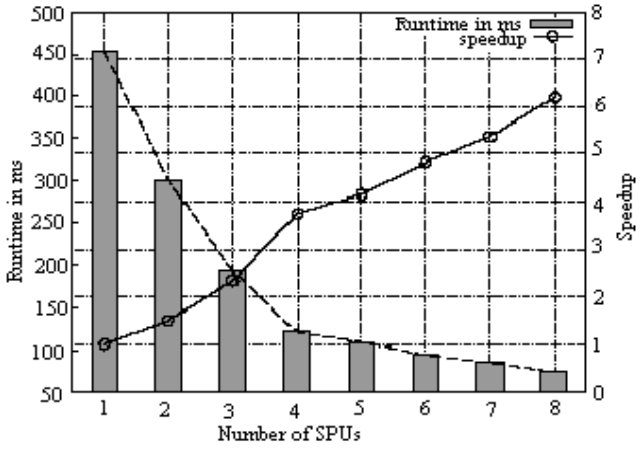

Fig 4. Global Alignment – This graph shows the runtimes and speed up of global alignment implementation

The results of the implementations are shown in "Fig 4" for up-to six SPUs on the PS3. The performance improvement are obtained by performing comparisons between a sequential implementation on a desktop with a 3.2 Ghz Pentium 4 processor, a sequential implementation of the space-saving global alignment for a single SPU on the PS3, parallel cell implementation and parallel implementation running on a single SPU on the PS3. The parallel implementation run on one SPU is performs worse when, as it includes the additional work that is decomposition phase which divide the whole problem as the sub-problem to solve. This is used to perform analysis on the scaling of the existing algorithm, and as analysis done on results of implementation. It is found that a speedup of about 6.5x was obtained on six SPUs and when a comparisons was done with best sequential algorithm, a speedup of 4.5x over a single SPU was obtained, and a speedup of more than 3.5x over a Pentium 4 processor was obtained. In "Fig 5", the results obtained using the cell simulator for up-to eight SPUs are given. The speedups are computed by comparing parallel implementation with the sequential implementation of global alignment that run on a single SPU. A speedup of about 6X was obtain for up-to eight SPUs.

4. MULTIPLE SEQUENCE ALIGNMENT USING FASTA TOOL ON CELL BROADBAND ENGINE

A graph is below comparing the total search times of 11 queries, against the swiss-prot 45 databases, 59,631,787 residues in 163,235 sequences. The programs were tested on three computers. The first was a Dell blade with two 1.6 GHz Xeon 5130 processors (8 cores total) with 4 Gbytes of RAM. The next computer was an IBM QS20 blade with two 3.2 GHz Cell B.E. processors and 1 Gbytes of RAM. The last computer was Sony's PlayStation 3 with a single 3.2 GHz cell broadband engine processor and 256 Mbytes of RAM. The tests were run with the scoring matrices BLOSUM50 with gap penalties. When porting the striped smith waterman algorithm to the cell broadband engine the following architectural features of the synergistic processing unit were taken into account:

- Limited size of the LS
- No support for saturated math
.
- No support for byte arithmetic

The synergistic processing unit has 256K of memory in the LS. This is used to hold both the program and its data. Many times smith waterman implementations build a position specific scoring matrix based on scoring matrix. The query sequence with the position specific scoring matrix would require 50 bytes for each residue of the query sequence. To reduce the amount of memory needed per query residue, the position specific scoring matrix is replaced with the query sequence and the shuffle instruction to dynamically generate the scoring vector. The memory requirement per query residue is now reduced to six bytes, allowing a query sequence of length 32Kbytes to fit in the local storage. The synergistic processing unit architecture does support saturated math. Saturated math is used in the smith waterman calculations to keep the scores from falling below zero and from rolling over when the maximum of the single instruction multiple data type is reached. Saturated math could be implemented with three instructions, but this could easily dominate the calculation time in the inner loop.

FASTA is a multiple sequence alignment algorithms used to align multiple sequences and thereafter compute pair of match and mismatch between the sequences. On the basis of this computation, it computes a score matrix to determine the similarity between the sequences. Eric Lindahl included an efficient implementation of an

AltiVec-enabled smith waterman is already in the FASTA package. For making the FASTA package to be executed in the cell broadband engine, the AltiVec application-programming interfaces are converted to the synergistic processing unit application programming interfaces. Many of the synergistic processing unit application programming interfaces obtained from AltiVec application programming interfaces with little effort (replacing the Vec of AltiVec to SPU for the SPU). However, Two of AltiVec application programming interfaces used in FASTA which are not presented on the synergistic processing unit, so it is necessary us to implement these application programming interfaces with multiple instructions.

- **Vec max:** Vec max application programming interfaces determines the element-wise maximum of two vectors and stores the result in the output vector. We implemented vec max for the synergistic processing element by using synergistic processing unit cmpgt to create a mask from the comparison of the two input vectors, and synergistic processing unit use this mask to extract the greater of the two vectors.

- **Vec subs:** Saturated subtraction is to be performs by this application programming interfaces, meaning that if any element of the result vector is negative, that element is set to zero. It is one of the useful application programming interfaces for the smith-waterman execution, since it needs a positive value at every matrix cell for local alignment. We implemented vec subs on the synergistic processing unit by performing a signed subtraction using synergistic processing unit sub and then finding the maximum of the signed result and a constant vector of all zeros, using our implementation of vec max described above.

To execute the smith waterman kernel on the synergistic processing unit, the alignment scores are pre-computed on the power-processing unit, and are DMAed to the synergistic processing unit along with the query and the library sequence. Other parameters such as the alignment matrix and the gap penalties are also included in the context for every synergistic processing unit. Despite the absence of instructions described above the cell processor, still outperforms in comparison to every superscalar processor currently in the market. This superior result is mainly due to the presence of eight synergistic processing unit cores and the vector execution on the synergistic processing units. Since the power-processing unit also supports Alti-vec instructions, it is also possible to use the power-processing unit as a processing element, thus enabling nine cores on the cell processor, with even better performance results. Further profiling of our implementation indicates that the computation dominates the total runtime (up to 99.9% considering a bandwidth of 18 Gbytes/s for the synergistic processing units), and hence multi-buffering is not needed for this class of computation.

The cell implementation discussed above is not fully functional as of now the current implementation requires both sequences to fit entirely in the synergistic processing unit local store of 256 Kbytes, which limits the sequence size to at most 2048 characters. To do genome-wide or long sequence comparisons, a pipelined approach similar to among the synergistic processing units could be implemented.

Each synergistic processing unit performs the smith-waterman alignment for a block, notifies the next synergistic processing unit through a mailbox message, which then uses the boundary results of the previous synergistic processing unit for its own block computation. Support of bigger sequences on the cell is a key goal of future research. Once a fully functional smith-waterman implementation exists on the cell, we can employ this kernel in the FASTA package. The FASTA package compares each sequence in a query sequence file with every sequence in a Library sequence file, and hence multiple issues for load balancing could be evaluated. For now, we have a simple round-robin strategy, in which the sequences in the query library are allocated to the synergistic processing units based on the sequence numbers and the synergistic processing unit number.

5. PARALLEL IMPLEMENTATION OF PROTEIN STRUCTURE PREDICTION ALGORITHMS

Protein structure prediction is the method of prediction of the three dimensional structure (that is, the prediction of tertiary structure) of a protein from its given amino acid sequences. Protein structure prediction plays important role in the field of medicine for example, in drug design. A large quantity of protein sequence data is produced by a large-scale DNA sequencing project such as the Human Genome Project. There are so many factors that make protein structure prediction a very difficult task. Out of which, two main problem are that the number of possible protein structures is so large, and protein's structure stability is not fully understood based on its physical property. Thus, any protein structure prediction method requires a way to explore efficiently the space of possible structures of protein.

Protein structure prediction on computational grid was described in [19]. They employ genetic algorithm. Applying, the genetic operator results in modification of the population's structure so as to intensify exploration inside a delimited segment or for diversification purposes. Protein structure prediction was also attempted on cell broadband engine [20]. They mapped the different stages of the BLASTP algorithm onto the Cell broadband engine. The two important contributions in the paper are (a) overcoming challenges due to limited local storage of the synergistic processing element data transfer (b) coordination between power processing element and synergistic processing elements. An efficient and flexible mechanism to transfer sequences from the database to the synergistic processing elements is implemented. A speedup of 3.2x to 3.6x times was achieved by this implementation.

6. CONCLUSIONS

In conclusion, we would like to say that the execution time of data intensive bioinformatics applications can be significantly improve by parallelization of highly time consuming portion of whole applications and data structure used in bioinformatics application. In future we take weighted suffix tree as data structure and we parallelize the weighted suffix tree construction on all possible parallel architecture Architecture .

7. REFERENCES

[1]. Y. L. Kuo, C. T. Yang, C. L. Lai and T. M. Tseng, "Construct a Grid Computing Environment for Bioinformatics", Proceedings of the Seventh International Symposium on Parallel Architectures, Algorithms and Networks, Hong Kong, 10-12 May 2004

[2] B. Turgeon, D. McCourt, J. Cowper, F. Palmer, S. McClean and W. Dubitzky, "Can the Grid help to solve the data integration problems in molecular biology", Third IEEE/ACM International Symposium in Cluster Computing and the Grid, Tokyo (Japan), Page(s):594 – 600, 12-14 May 2003

[3] B. Wang, Y. Liu, J. Yun and S. Liu, "Application Research of Protein Structure Prediction Based Support Vector Machine", In International Symposium on knowledge Acquisition and Modelling, Wuhan (China), Page(s): 581-584, 21-22 Dec 2008

[4] S. Parthasarathy and M. Coatney, "Efficient Discovery of Common Substructures in Macromolecules" , In Proceedings of IEEE International Conference of Data Mining, Maebashi City (Japan), Page(s):362-369, 9-12 Dec 2002

[5] E. R. Mardis, "Engineering in genomics: Technical improvements in high throughput genome sequencing", Engineering in Medicine and Biology Magazine, vol.14 (6), Page(s):794 – 797, Nov-Dec 1995

[6] S. Zhaofeng, Q. Hongze, Z. Daming and H. Feng, "New Evolutionary Subset: Application to Symbiotic Evolutionary Algorithm for Job-Shop Scheduling Problem", Fourth International Conference Natural Computation, vol.1, Jinan (China), Page(s):470 – 475, 18-20 Oct. 2008

[7] C. Kwangmin, A. Saple and S. Kim, "Genome Data Type: a Vehicle to Deliver a Genome Comparison System on the Web", Sixth IEEE International Conference on Data Mining, Hong Kong (China), Page(s):142 – 146, 18-22 Dec- 2006

[8] Y. Liu, T. Yan and R. Zhang, "Protein Sequence Analysis Based on Smooth PST", Third International Conference on Bioinformatics and Biomedical Engineering, Beijing (China), Page(s): 1-4, 11-13 June 2009

[9] K. H. Emden, "Molecular Phylogenetic Analysis and real Life Data", Computing in Science and Engineering, vol.7 (3), Page(s): 86-91, May-June 2005

[10] Y. B. Choi and I. K. Cheang, "Providing white page service to the Human Genome Project", In Proceedings IEEE International Conference on Communication Systems, vol.3, Singapore, Pages(s): 1185-1189, 14-18 Nov 1994

[11] T. Almas, Z. Tesfave and I. D. Donn, "Simulation of Electrical Conduction in Cardiac Tissues on High Performance Computing", 19th International Symposium on High Performance Computing and Application, Guelph Ontario (Canda), Pages(s):244-250, 14-18 May 2005

[12] Z. Aung, W. Fu and K. L. Tan, "An efficient index-based protein structure database searching method" International Conference on Database System for Advance Application, Kyoto(Japan), Page(s):311-318, 26-28 March 2003

[13] C. Wilks and S. Khuri, "A Fast Shotgun Assembly Heuristic", IEEE Computational Systems and Bioinformatics Conference, Stanford (CA), Page(s):122-123, 8-11 August 2005

[14] I. Gorton, P. Greenfield, A. Szalay and R. Williams, "Data-Intensive Computing in the 21st Century", In Computer Magazine of IEEE Computer Society, vol41(4), Page(s) 30-32, April 2008

[15] V. Sachdeva, M. Kistler, E. Speight and T. H. K. Tzeng, "Exploring the Viability of the Cell Broadband Engine for Bioinformatics Applications", IEEE international Parallel and Distributed Processing Symposium, Long Beach California (USA), Page(s):1-8, 26-30 March 2007

[16] T. F. Smith and M. S. Waterman, "Identification of Common Molecular Subsequences", Journal of Molecular Biology, vol. 147(1), Page(s): 195-197, 1981

[17] Y. Chen, S. Yu and M. Leng, "Parallel Sequence Alignment Algoritms for Clustering System", International Federation for Information Processing (IFIP) (Boston: Springer), vol.207, Page(s): 311-321, 2006

[18] A. Wirawan, K. C. Keong and B. Schmidt, "Parallel DNA Sequence Alignment on the Cell Broadband Engine", Springer-Verlag Berlin Heidelberg, Page(s): 1249-1256, 2008

[19]. 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(5), Page(s): 39-43, 2007

[20]. H. Zhang, B. Schmidt and W. M. Witting, " Accelerating BLASTP on the Cell broadband Engine", In Proceedings of the Third International Conference on Pattern Recognition in Bioinformatics, vol.5265, Lecture Notes in Bioinformatics, Springer Berlin Heidelberg, Page(s) 460-470, 8 Oct 2008

[21] J. Ebedes and A. Datta, "Multiple Sequence Alignment in Parallel on a Workstation cluster", Oxford University Press, vol.20 (77), Page(s):1193-1195, 2004