Empirical Analysis and Performance Evaluation of various GPU Implementations of Protein BLAST

Sita Rani
Ph. D. Research Scholar
I.K.G. Punjab Technical University
Kapurthala (Punjab) INDIA

O. P. Gupta
Associate Professor
Punjab Agricultural University
Ludhiana (Punjab) INDIA

ABSTRACT
Bioinformatics applications are compute and data intensive by nature. As the size of molecular databases is growing from day to day experiments performed in the field of molecular biology, thoughtful steps need to be taken to exploit various methods to accelerate bioinformatics applications. Many efforts have already been put in the field to optimize most of the bioinformatics algorithms. By incorporating Graphical Processing Units (GPUs), many bioinformatics applications have benefited hugely. Compute Unified Device Architecture (CUDA) is a hardware and software platform, used to exploit multi-threaded architecture of GPUs. Basic Local Alignment Search Tool (BLAST) is one of the most frequently used algorithms for bioinformatics applications. Different GPU implementations of protein BLAST have already been proposed by different authors. For each implementation, the authors claimed different speedups. But these implementations are on different hardware platforms and also were experimented with different databases, so it's difficult to compare their performance accurately. In this paper four different GPU implementations of protein BLAST are explored in detail. To compare their performance, these GPU versions of BLAST are implemented on a common hardware platform, i.e. NVIDIA M2050 GPU with 448 processing cores, 3GB of memory and two hex-core Intel, Xeon 2.93 GHz processors. Experiments are performed on 2.38 GB protein database. Performance is analyzed and compared with standard NCBI-BLASTP. Parameter considered for performance analysis and comparison is the execution time. In the current environment speedup obtained by different implementations varied from 2.3X to 9.8X.

Keywords
BLAST, Bioinformatics, CUDA, GPU, Sequence Alignment, Thread.

1. INTRODUCTION
To compare a new genome sequence with existing sequences in the database is one of the most frequently used tasks in majority of bioinformatics applications [1]. This activity is performed to identify the extent of similarity between the new sequence and existing sequences. The degree of similarity identifies the biological analogy between the sequences. Smith-Waterman algorithm was the very first algorithm proposed to find local similarities among query sequence and database sequences [24]. But this algorithm had high values for time complexity. So a new algorithm BLAST, based on heuristic approach was proposed. This tool is time efficient as compared to Smith-Waterman algorithm [2], [3], [4] and [13]. With day-to-day research, size of molecular databases has already grown immensely and is still growing. Because of its heuristic approach, BLAST became popular very shortly and now it is the most widely used tools for different bioinformatics applications. Because of its popularity, many researchers and scientists have already put immense efforts to further improve its execution time and are still working.

With the incorporation of GPUs in general purpose computing, a new era of parallel computing has started. CUDA is a hardware and software platform provided by NVIDIA, which is specially designed to explore the parallel architecture of GPUs by using multithreaded code [5].

In most of the bioinformatics applications, there is a need to deal with huge databases, so these applications are data intensive [18], [19], [20] and [23]. By using the multithreaded programming approach of CUDA on GPUs, these applications can be accelerated greatly.

Many bioinformatics algorithms and tools have already been optimized on GPUs. BLAST has also been implemented on GPUs by different authors by using different approaches. Authors of different implementations have claimed different speedups.

Because all the proposed implementations are on different hardware platforms, it's difficult to compare the performance mutually. In this paper four different GPU versions of protein BLAST are implemented on a common hardware platform and performance is compared. The scope of further improvement is also analyzed and discussed.

1.1 BLAST
BLAST is the most widely applied tool to compare a query sequence (protein sequence/ nucleic acid) with the database sequences in numerous bioinformatics applications [6], [21], [26] and [27]. Various tasks performed for this search are executed in four different stages of protein BLAST, discussed below [7]:

- **Hit Detection:** Here, the comparison is done between the query sequence and database sequences to locate words matches for specified word length and each identified word match is called a hit.
- **Un-gapped Extension:** In this stage, the score for each word match is compared against the threshold value after extending the word match in both the directions without gaps. Matches, which score higher than the threshold value are called High Scoring Pairs (HSPs), are considered for next stages and all other word matches are discarded.
- **Gapped Alignment:** In this stage, un-gapped alignments retained in the previous stage are extended by inserting gaps. Alignments with a score higher than the threshold value are called High Score Alignments (HSAs).

- **Gapped Alignment with Trace- back:** Here trace-back path is determined for the final alignments and results are displayed to the user.

Various BLAST variants [5] are available for different type of query and database sequences. These are:

- **Blastn:** In Blastn, nucleotide queries are executed on nucleotide databases for alignment. In this variant of BLAST, short sequences of nucleotides also called oligonucleotides are compared with long sequences to identify characteristic of unknown nucleotide sequences.
- **Blastp:** In Blastp, both query sequences and database sequences are proteins. In this a query sequence of unknown features is compared with protein sequences of known functionality in the databases. Functionality of the unknown sequence is identified from best aligned sequences from the database.
- **Blastx:** In Blastx, a nucleotide sequence is aligned with protein databases. Before the alignment nucleotide query sequence is converted to protein sequence. Three protein sequences are generated for each nucleotide sequence. The first sequence is obtained by converting three nucleotides to a protein. Then another two protein sequences are generated in the same method, but leaving first and then first two letters of the original nucleotide sequence. The Blastx is mainly used in genomic DNA to identify protein coding genes.
- **tBlastn:** In tBlastn, a protein query sequence is executed on a nucleotide database. To obtain accurate alignments, before query execution, each database sequence is converted to three protein sequences as explained above in Blastx. Then query sequence is aligned with different protein sequences in resultant database. Its application is in protein mapping of the genome.
- **tBlastx:** In tBlastx, both query sequence and databases, both are of nucleotide type. So both query sequence and database sequences are converted to protein sequences first (as explained above), then query sequence is processed in the database. tBlastx is used to determine transcripts of unknown functions.

These different blast variants are summarized in Table 1.

Table 1: BLAST variants, type of query and database sequences

BLAST Variant	Type of Database	Type of Query Sequence
Blastn	Nucleotide	Nucleotide
Blastp	Protein	Protein
Blastx	Protein	Translated Nucleotide
tBlastn	Translated Nucleotide	Protein
tBlastx	Translated Nucleotide	Translated Nucleotide

In this paper protein BLAST i.e. BLASTP is being used for study and analysis.

1.2 BLAST Parameters

Some important parameters of protein BLAST are discussed below [8]:

- **Word size (w):** It specifies the length of the match. This parameter is defined by the user. The algorithm is executed with specified word length to find the matches in the sequences. The smaller the value of w, more the number of matches will be identified during the alignment and vice-versa.
- **Threshold (t)**: It is also a user defined parameter. It is the minimum defined value for a word match. Matches with score less than the threshold are assumed to occur by chance and discarded. Lesser the value threshold, there is probability of higher number of hits.
- **Drop-off (x):** Score in an alignment, is permitted to fall by drop-off value as compare to already available highest score value.
- **Lambda (ƛ):** ƛ is matrix specific constant. The score is normalized using matrix specific constant.
- **Adjustment (k):** It is called Adjustment factor. It is assumed that alignments at different locations and their scores may be co-related.
- **Gap penalty:** Gap penalty reduces the effect of idles in an alignment.

During our analysis of different GPU versions of protein BLAST, the values used for different parameters are the default values as shown below in the Table 2.

Table 2: Different parameters of protein BLAST with their values for current analysis

Parameter	Value
Word size	3
drop-off value for un-gapped extension	7
drop-off value for gapped extension	15
drop-off value for triggering gaps	22
drop-off value for final gapped alignment	25
open gap penalty	-7
extension gap penalty	-1

1.3 GPUs and CUDA

GPUs were initially designed to accelerate the speed of graphical operations because in graphical computations same operation is performed on similar type of data. But later on, GPUs was started to be used in general purpose computing. GPUs work on the principal of Single Instruction Multiple Data (SIMD). A GPU consists of many streaming multiprocessors (SMs) where each SM further consists of some scalar processors (SPs) [16] and [17].

In 2007 NVIDIA launched CUDA. CUDA is a software and hardware environment. Since then many compute intensive applications in different domains have taken the advantage of

GPU computing. Even many bioinformatics applications have also been implemented on GPUs and have achieved a considerable speed up [22].

CUDA is an extended version of C/C++. It is used to program GPUs. A CUDA program contains a kernel which consists of sequential executable part of the program. It is used to code the part of a program which can be processed by a single thread. Then kernel is called to execute multiple threads concurrently. These threads are organized as thread blocks and further as grids [9], [14] and [25]. One SM executes all the threads of a thread block. These threads are processed in Single Instruction Multiple Thread (SIMT) fashion and all the threads executed in parallel by an SM is called a warp [15] as shown in Figure 1.

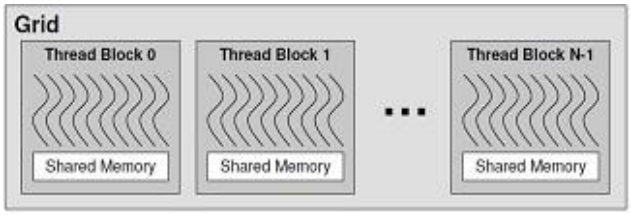

Figure 1: Threads organized as Blocks and Grids

2. RELATED WORK

In the very first GPU implementation of protein BLAST by Ling et al., the authors used two-hit method for gapped-BLAST. Two kernels were designed in this implementation. First kernel was used to perform the steps up to un-gapped alignment and last two steps were implemented by second kernel. It was discussed that seeding stage of BLAST plays main role in the performance of a GPU implementation of BLAST. When there are fewer matches identified, lesser number of threads are executed in parallel. GPU memory is utilized to store different data structures. Experiments were performed by the authors against Swiss-Prot protein database. The authors claimed a speedup of 1.7X to 2.7X as compare to standard NCBI- BLAST. It was also highlighted that for smaller query sequences, speedup achieved was high as compare to longer query sequences [10].

In another implementation by Xiao et al., it was demonstrated that 99% of the execution time is consumed by first three stages of BLASTP. So authors focused on these stages in their implementation. They also explained that the first two stages cannot be isolated from each other. In this implementation, for hit detection words were picked from the database sequence one by one and compared with query sequence. In the next step, multithreading was used to align sequences, where each thread aligned a query sequence and a database sequence. In their implementation, they also optimized the performance of the algorithm by placing different data structures in suitable memories of the GPU. Scoring matrix and query sequence were placed in constant memory because of the fast access time and small size of these structures. Subject sequence and word lookup table were stored in texture memory because they are large in size and faster to access. In this implementation, sequences were assigned to different threads using a greedy algorithm.

In this implementation, parallelization was done only for first two stages i.e. hit detection and un-gapped extension. They demonstrated with their experiments that when first two phases were implemented on the GPU, a speedup of 7X was obtained for these two phases only. But when both, execution of Dual core CPU and GPU was pipelined then a speedup of 6X was obtained in total execution time [7]. To provide these speedup figures, following five different implementations were tested by the authors:

- Serial execution on CPU.
- The GPU was used to implement all the stages (three).
- The GPU was used for first and second stage and the third stage was implemented on the CPU.
- The GPU was used for first and second stage and third stage executed on two threaded CPU
- The GPU is used for first and second stage in parallel to two threaded CPU for the third stage.

Liu et al., gave another GPU implementation of BLAST for protein databases. In this implementation, the authors focused on data structure design. Hit detection information was stored with compressed Deterministic Finite State Automaton (DFA). The GPU was used by the authors to implement stage 1, 2 and 3, whereas stage 4 was implemented on the CPU. Two different kernels were designed for the GPU implementation of this approach. Stage 1 and 2 were implemented with one kernel and a second kernel for stage 3.

During algorithm execution, each thread worked on one database sequence to identify word matches. For each word match, first un-gapped extension was performed by taking threshold value. Then gapped extension was performed by the GPU. Final results were compiled using trace-back by the CPU in stage 4 and are displayed. This implementation was memory optimized and all the data structures were placed in best possible memories by considering the size of data structure and memory and frequency of access.

The authors claimed a speedup of 10X as compared to standard NCBI-BLAST. But alignments generated did not match with standard NCBI-BLAST, so there is a doubt for its use by other researchers [9].

GPU-BLAST by Vouzis et al., is the most reliable among all the discussed implementations because this was designed on the top of standard NCBI-BLAST's source code. Input and output format were also same. Alignments generated exactly matched with NCBI-BLAST.

The authors explained that the majority of the data structures used in their implementation were the same as NCBI-BLAST. Only some optimization was done to exploit the GPU. Along with GPU, even multi core CPU was also utilized to its maximum level. And load was well distributed between the CPU and GPU, so that when GPU is in execution CPU should not be idle.

GPU's global, constant and shared memories were used in this design. Authors have told that local and texture memories were not being used in their approach. Some of the important data structures used by the authors in this implementation are:

- Substitution matrix
- Presence bit vector
- Query index table
- Overflow table

These data structures were also optimized by considering the size and frequency of usage. In this implementation, GPU and CPU both were used during seeding and extension phases in parallel. After calculation of High Scoring Pairs by the CPU and GPU, the results were given back to the CPU and the rest

of the execution was performed by the CPU as done in standard NCBI-BLAST.

The authors claimed a speedup between 3X and 4X in comparison to standard NCBI-BLAST. There is a special feature of this implementation; both the options are available for CPU BLAST as well as for GPU BLAST. This feature is not provided in any previous implementation [11].

3. PRESENT WORK

3.1 Hardware and Software Environment

Hardware used in the present work for analysis of performance of different GPU implementations of BLASTP is NVIDIA M2050 GPU with 448 processing cores, 3GB of memory and two hex - core Intel, Xeon 2.93 GHz processors. Operating System used on the system is Red Hat Enterprise Linux 5.6 All the experiments are performed with CUDA 5.5.

3.2 Experimental Data

Protein database of 2.38 GB was collected from National Center for Biotechnology Information (NCBI) site. The database was converted into FASTA format. Different sized protein sequences were used to perform the different experiments and gather results.

4. RESULTS AND DISCUSSION

All the four different GPU- implementations of protein BLAST, discussed above, were implemented on a common hardware platform, so that their performance can be analyzed and compared accurately. Experiments were performed with the query sequences of different lengths ranging from 100 characters 4000 characters to derive the inferences. The parameter taken for analysis and performance comparison of these implementations is the execution time. The results collected with the queries of varying length for all the four GPU implementations and standard NCBI-BLAST are shown in Table 3.

Table 3: Execution Time with different GPU implementations of protein BLAST and standard NCBI-BLAST for queries of different lengths

Query Length	Execution Time (Sec)				
	NCBI-BLAST	Imp -I (C.Ling)	Imp-II (S.Xiao)	ImpIII (W.Liu)	Imp -IV (P.D. Vouzis)
500	2023	880	316	206	547
1000	2084	906	326	213	563
1500	2101	910	328	214	568
2000	2203	960	344	229	600
2500	2286	994	357	233	618
3000	2327	1012	364	237	629
3500	2735	1189	427	279	739
4000	3739	1627	587	382	1011

Figure 2 shows the graph depicting the execution time of four different GPU-implementations of protein BLAST on the discussed hardware platform for different query sequences of varying length.

From Figure 2, it can be easily analyzed that implementation proposed by C. Ling is the slowest among all and W. Liu proposed the fastest GPU implementation of the protein BLAST.

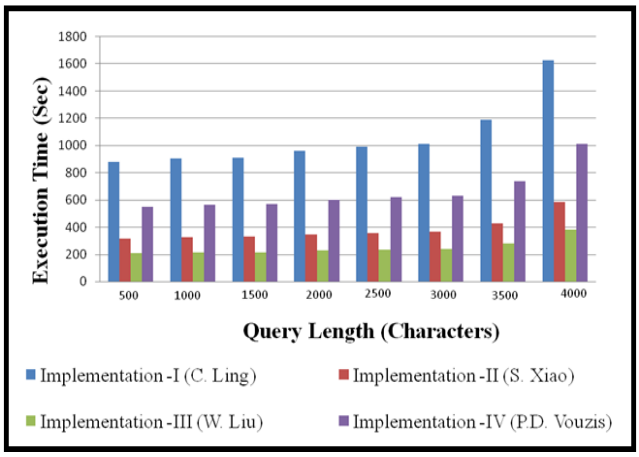

Figure 2: Query Length verses Execution Time of different GPU- implementations of protein BLAST

Execution time of different GPU implementations of protein BLAST, along with execution time of standard NCBI – BLAST is shown in Figure 3 with the help of line-graph. In Figure 3, results for more number of query sequences are taken for more accurate comparison of different implementations along with NCBI-BLAST.

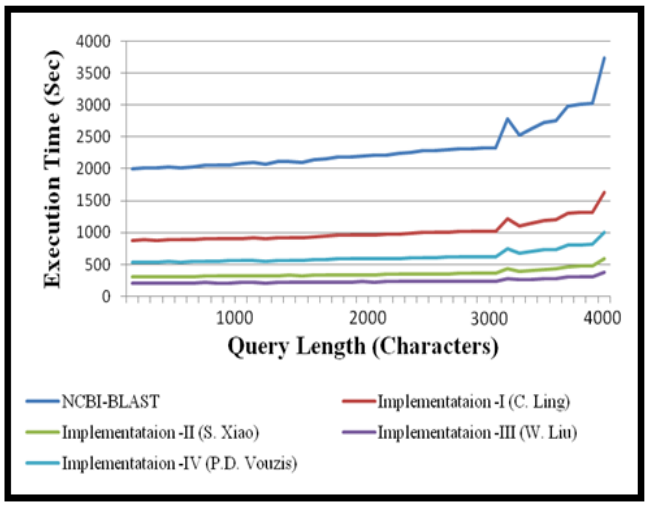

Figure 3: Execution time comparison of different GPU-implementations of protein BLAST with standard NCBI-BLAST

When all the four GPU versions of protein BLAST were implemented on common hardware platform, they have shown varying speedup as compare to standard NCBI-BLAST. Speedup obtained by each implementation is depicted with a line graph in Figure 4. Average speedup for different implementations varied from 2.3X to 9.8X in comparison to NCBI-BLAST as shown in Table 4.

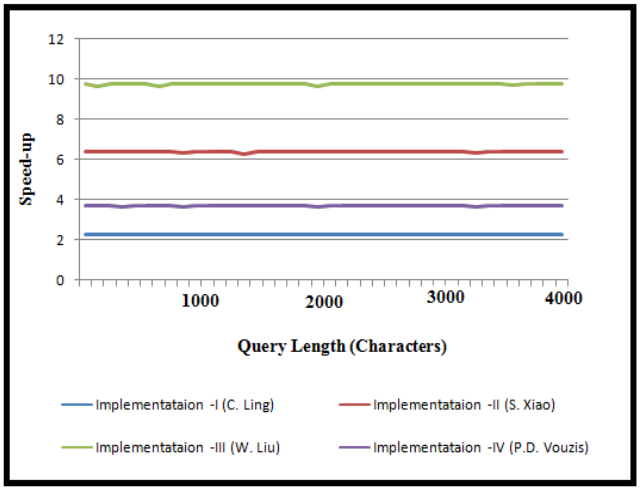

Figure 4: Speedup obtained by each GPU implementation of protein BLAST in comparison with standard NCBI-BLAST

Minimum speedup is shown by the implementation proposed by C. Ling and maximum speedup is obtained with the implementation proposed by W. Liu as depicted in Figure 4. But there is a major drawback of the implementation proposed by W. Liu, alignments generated did not match with standard NCBI-BLAST as it was highlighted by the authors.

Table 4: Average speedup obtained with different GPU implementations of protein BLAST in comparison with standard NCBI-BLAST

Implementation	Authors	Speed-up in comparison with standard NCBI-BLAST
I	C. Ling and K. Benkrid	2.3
II	S. Xiao, H. Lin and W. Feng	6.4
III	W. Liu, B. Schmidt and W. Muller-Witting	9.8
IV	P.D. Vouzis and N.V. Sahinidis	3.7

5. CONCLUSIONS

The prime objective of this research was to measure the performance of different GPU-implementations of protein BLAST on a common hardware platform and database. In this research exhaustive experimentation is done to obtain more accurate results. The parameter used for comparison is the execution time to process the query sequence against database. The experiments are performed by taking the queries of different length ranging from 100 characters to 4000 characters. The comparison is also done with standard NCBI-BLAST by calculating the speedup of different implementations. Average speedup, in comparison to standard NCBI- BLAST varied from 2.3X to 9.8X. Implementation proposed by Ling et. al., is 2.3 times faster than NCBI-BLAST. Whereas Liu. et. al. proposed fastest implementation with a speedup of 9.8.

6. FUTURE SCOPE

During analysis, its being observed that in all the implementation discussed, only one query sequence is processed on all the threads GPU at a time. But each streaming multiprocessor of the GPU can execute a different query on complete database or on a part of the database by following SIMT approach. So still a next level of parallelism need to be exploited for BLAST on GPUs for multiple query execution in parallel. Further BLAST can be implemented on GPU enabled High- Performance Cluster (HPC) for multiple query sequence processing to reduce execution time.

7. ACKNOWLEDGEMENT

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

8. REFERENCES

[1] Isaza, S., Sanchez, F., Cabarcas, F., Ramirez, A. and Gaydadjiev, G., "parameterizing multicore architectures for multiple sequence alignmnet", International *Conference on Computing Frontiers*, May3-5, 2011, Ischia, Italy.

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

[3] Altschul, S.F., Gish, W. , Miller, W., Myers, E.W. and Lipman, D.J. " Basic Local Alignmnet Search Tool", *J. Molecular Biology*, 1990, vol. 215, pp. 403-410.

[4] Dematte, L. and Prandi , D. , " GPU computing for system biology" , *Briefings in Bioinformatics*. 2010, vol. 11. No. 3, pp. 323-333.

[5] Kindratenko, V.V., Enos, J.J. , Shi, G., Showerman, M.T., Arnold, G.W., Stone, J.E., Phillips, J.C. and Hwu, W., " GPU Clusters for high performance computing" , *IEEE International Conference on Cluster Computing and Workshop*, Aug, 31-Sep,4, 2009, New Orleans, LA, pp. 1-8.

[6] Sharma , T.R., *" Genome Analysis and Bioinformatics*" , 2009, pp.67.

[7] Xiao, S., Lin, H. and Feng , W. " Accelerating Protein Sequence Search in Heterogeneous Systems" , IEEE Parallel and Distributed Processing Symposium, May, 2011, Anchorage, AK, pp. 112-1222.

[8] Ree, E. J. and B. S., "Parallelization Methods for the Distribution of High Throughput Bioinformatics Algorithms" , *Ph.D. Dissertation* , Texas University, 2011 .

[9] Liu, W., Schmidt, B. and Muller-Witting, W., " CUDA-BLASTP: accelerating BLASTP on CUDA-enabled graphics hardware*", IEEE/ACM Transaction on Computational Biology and Bioinformatics* , 2011, vol. 8, no. 6, pp. 1678-1684.

[10] Ling, C. and Benkrid, K., " Design and implementation of a CUDA-compatible GPU-based Core for gapped BLAST algorithm*", International Conference on Computational Science* , 2010 is available on Science Direct Procedia Computer Science, vol. 1, no 1,pp.495-504.

[11] Vouzis, P.D., and Sahinidis, N.V., "GPU-BLAST: using graphics processors to accelearte protein sequence alignment", *BIOINFORMATICS*, vol.27, no. 2, 2011, pp. 182-188.

[12] National Center for Biotechnology Information: http://www.ncbi.nlm.nih.gov/

[13] Altschul, S.F., Madden, T.L., Schaffer, A.A. , Zhang, J., Zhang, Z., Miller, W., and Lipman, D.J., "Gapped BLAST and PSI-BLAST: a new generation of protein database search programs", *J. Nucleic Acid Research*, 1997, vol. 25, no. 17, pp. 3389-3402.

[14] Luebke, D., "CUDA: Scalable parallel programming for high performance computing", *Proc. 5th IEEE International Symposium on Biomedical Imaging, Paris*, 2008, pp. 836-838.

[15] Liu, Y., Maskell, D.L., and Schmidt, B.," CUDASW++: optimizing Smith-Waterman sequence database searches for CUDA-enabled graphics processing units", *BMC Research Note*,2009, vol. 2, no. 73, pp. 1-10.

[16] Nickolls, J. and Dally, W.J. " The GPU Computing Era", *J. IEEE Computer Society Micro*, 2010, vol. 30, no. 2, pp. 56-69.

[17] McClanahan, C., "History and Evolution of GPU Architecture", 2010, A paper Survey. http://mcclanahoochie.com/blog/wpcontent/uploads/2011/03/gpu-hist-paper.pdf

[18] Fenstermacher, D.,"Introduction to Bioinformatics", *J. of American Society for International Science and Technology*, 2005, vol. 56, no. 5, pp. 440-446.

[19] Albayraktaroglu, K., Jaleel, A., Wu, X., Franklin, M., Jacob, B., Tseng, C.W. and Yeung, D. " BioBench: A Benchmark Suite of Bioinformatics Applications", *Proc. IEEE International Symposium on Performance Analysis of Systems and Soft- wares, Austin*, TX,2005, pp. 2-9.

[20] Cohen, J. "Bioinformatics: An Introduction to Computer Scientists", *ACM J. Computing Surveys*, 2004, vol. 36, no. 2, pp. 122-158.

[21] Baxevanisand D.A., and Ouellette, B.F., *"BIOINFORMATICS A Practical Guide to the analysis of Genes and Proteins,"* John Wiley and Sons INC., U.K., 2006, pp. 82-102.

[22] Pang, B., Zhao, N., Becchi, M., Korkin, D. and Shyu, C. " Accelerating large-scale protein structure alignments with graphics processing units", *BMC Research Notes*, 2012.

[23] Lin, C., Hung, C. , and Huang, J., "Efficient GPU-Based Algorithm for Aligning Huge Sequence Database", *IEEE International conference on High Performance Computing and Communications, 10th IEEE International Conference on Embedded and Ubiquitous Computing, Zhangjiajie,* 2013, pp. 1758-1762.

[24] Lee, S., Lin, C. and Hung, C.L. " GPU-Based Cloud Service for Smith-Waterman Algorithm using Frequency Filtration Scheme", *BioMed Research International*, Research Article, vol. 2013, pp. 1-8.

[25] Zhu, X. , Li, K., Salah, A., Shi, L. and Li, K., " Parallel Implementation of MAFFT on CUDA-Enabled Graphics Hardware", *IEEE/ACM Transaction on Computational Biology and Bioinformatics*, 2015, vol. 12, no. 1, pp. 205-218.

[26] Zhang, J., Wang, H., Lin, H. and Feng, W., "cuBLASTP: Fine Grained Parallelization of Protein Sequence Search on a GPU", *IEEE 28th Parallel and Distributed Processing Symposium*, Phoenix, AZ, 2014, pp. 251-260.

[27] Gupta, OP. and Rani, S., "Accelerating Molecular Sequence Analysis using Distributed Computing Environment", *International Journal of Scientific and Engineering Research*, 2013, vol. 4, no. 10, pp. 262-266.

9. AUTHOR PROFILE

Sita Rani is pursuing her Ph.D. in Computer Science and Engineering from I.K.G. Punjab Technical University, Kapurthala. She received her B.Tech (Computer Science and Engineering) and M.Tech (Computer Science and Engineering) in the years 2002 and 2008 respectively from Guru Nanak Dev Engineering College, Ludhiana. Currently, she is working as Associate Professor cum Head of Department at Ludhiana Group of Colleges, Chaukimann. Her current research interests are Software Engineering, Data Structures, Parallel Computing and Bioinformatics.

OP Gupta, Ph.D. (Computer Science & Engineering) is an alumni of PAU, Ludhiana, Thapar University, Patiala and GNDU, Amritsar has demonstrated his intellectual, interpersonal and managerial skills in various domains. He is the winner of PAU Meritorious Teacher Award for 2009-2010. Having vast industrial experience of working in IT industry with the role of Project Manager, currently he is an Associate Professor of Computer Science and Deputy Director, School of Information Technology at PAU, Ludhiana. He is also approved Ph.D supervisor of I.K.G. Punjab Technical University, Kapurthala. His areas of interests include Parallel and Distributed Computing, Grid Computing, Bioinformatics, Network Testing and Network Management.