# Implementation and Evaluation of mpiBLAST-PIO on HPC Cluster

Nisha Dhankher
School of Electrical Engg. & IT
COAE&T, PAU
Ludhiana, India

O P Gupta
School of Electrical Engg. & IT
COAE&T, PAU
Ludhiana, India

## ABSTRACT
Due to exponential growth in the size of genomic databases, traditional techniques of sequence search proved to be slow. To address the above problem, an open source and parallel version of BLAST called mpiBLAST was developed by the programmers. In mpiBLAST, the master process distributes the database fragments among worker nodes to compute the sequence search in parallel. As merging and writing of the results is done sequentially by the master process, it would create performance bottleneck with increasing number of processors and varying database sizes. To handle this high non-search overhead, mpiBLAST-PIO was introduced. This paper describes the optimized and extended version of mpiBLAST called mpiBLAST-PIO. The goal of this research was to investigate the performance of parallel implementation of BLAST in comparison to sequential NCBI-BLAST by measuring Speedup and efficiency on HPC platform using Infiniband. Different options of mpiBLAST-PIO were activated that helped in understanding the optimal parameters for achieving highly scalable parallel BLAST implementation. The results found that parallel-writing of the results, can evolve as an efficient solution when high-performance parallel file system is available.

## General Terms
Algorithms, High Performance Computing, Bioinformatics

## Keywords
mpiBLAST-PIO, Parallel & Distributed Computing, High Performance Computing, Bioinformatics

## 1. INTRODUCTION
Today,genomic sequence search is one of the most important and basic problem in computational biology. Sequence comparison, also called sequence alignment refers to the procedure of comparing two or more biological sequences by searching for a series of characters that appear in the same order as in the input sequences. The goal of sequence comparison is to determine the regions of similarity between two genetic sequences. The similarities between newly discovered sequence and sequence of known functions can help in identifying functions of new sequence and find sibling species from a common ancestor.

The alignment of two sequences (pairwise alignment) requires different types of algorithms. The algorithms used can be dynamic programming based or heuristic based. Dynamic programming based algorithms generate optimal solutions but are computationally intensive so impractical for a large number of sequence alignments. E.g. Needleman–Wunsch algorithm and Smith–Waterman algorithm. To reduce the time complexity, heuristic based algorithms are used that generate a near-optimal solution. Heuristics are approximation algorithms. E.g. BLAST, FASTA.

### 1.1 BLAST Algorithm
Basic Local Alignment Search Tool (BLAST**)** proposed by Altschul *et al* searches a query sequence containing nucleotides (DNA) or peptides (amino acids) against a database of known nucleotide or peptides sequences. A scoring matrix is usually used to estimate the statistical probability of the match/mismatch at each position in the sequence. Penalties are also assigned on introducing and extending gaps in the alignment. The most widely used scoring matrices are Percent Accepted Mutations (PAM) and Blocks Substitution Matrix (BLOSUM). BLAST's final result consists of a series of local alignments, ordered by the similarity score along with an e-value. Databases in BLAST are in fact text file in FASTA format. Factors affecting the BLAST performance are query batch size, database size and search sequence length. BLAST program can search sequences against database, with on-the-fly translations. BLAST search types are:

i. blastn: search nucleotide database using a nucleotide query.
ii. blastp: searches protein database using a protein query.
iii. blastx: search protein database using a translated nucleotide query.
iv. tblastn: search translated nucleotide database using a protein query.
v. tblastx: search translated nucleotide database using a translated nucleotide query.

With the exponential growth in the size of sequence database's searching database that can not fit in the main memory became a serious performance issue. This prompted the researchers to develop a parallel algorithm to keep pace with the current rate of sequence acquisition. As BLAST is both computationally intensive and parallelizes well, many parallel and distributed approaches of parallelizing BLAST have been proposed.

Main approaches of Genomic Sequence Search Parallelization are:

i. Hardware Parallelization: It is implemented during the sequence alignment stage by representing search space as a matrix. e.g. Bioscan. Tera-BLAST.

ii. Query Segmentation: In this, the entire database is replicated on each compute node local storage system and individual nodes concurrently search subsets of query against whole database. But if the

database to be searched is larger than core memory than query segmentation search suffer from excessive disk I/O.

iii. Database Segmentation: Database segmentation permits each node to search a smaller portion of the database (one that fits in the main memory), eliminating disk I/O and vastly improving BLAST performance.E.g.TurboBLAST, mpiBLAST, Bioinformagic, BeoBLAST and parallelBLAST.

## 1.2 Parallelizing BLAST using MPI

### *1.2.1 mpiBLAST Algorithm*

mpiBLASTis an open-source parallelization National Center for Biotechnology Information (NCBI) BLAST based on database segmentation. It is designed to work on a computer cluster using MPI library and adopts a master-slave style. mpiBLAST provides a tool called mpiformatdb, a wrapper, which integrates the database formatting and partitioning of the tasks. The master is responsible for broadcasting query sequences, assigning database fragments to worker nodes, and merging search results from worker nodes. The workers process the queries and send back the results to master. Whenever one of the slaves completes the task and reports to be idle, the master assigns a new fragment to it. Once the master receives results from all the workers for a query sequence, it calls the standard NCBI BLAST output function to format and print results to an output file in any format including XML, HTML, tab-delimited text, and ASN.1.

As more slaves are added for computation, it becomes difficult for the master to handle all the output results. To address this problem pioBLAST was introduced which greatly improved the performance by parallel- writing of the output results [8]. One of pioBLAST's main updates was the caching of sequences by worker nodes as they find potential alignments in their partial results [3]. As a result, some of its enhancements were added to mpiBLAST which are available since release of version 1.6. mpiBLAST-PIO is an optimized and extended version of parallel and distributed-memory version BLAST. The extensions include a virtual file-manager, a "multiple master" runtime model, efficient fragment distribution and intelligent load balancing[2, 16].

In this paper, the parallel computation model is based on mpiBLAST-PIO algorithm (latest version), database segmentation, query sequence distribution and master-worker paradigm using MPI-IO. This paper discusses and analyzes the parameters of mpiBLAST-PIO on HPC Cluster to estimate the performance of parallel version of BLAST as compared to the serial NCBI-BLAST.

# 2. MATERIAL & METHODOLOGY2.1 Cluster Hardware

All the experiments were run on a HPC Linux cluster installed at Data Centre of SEEIT, PAU. The cluster is composed of 40 compute nodes, each with two hexa-cores Intel, Xeon 2.93 GHz processors (total 480 processing cores), 12 MB cache, 50 GB RAM. Two head nodes, each with two quad-core processors and 32 GB RAM are used to manage the cluster. The intercommunication network between the computing nodes consists of 40 Gbps Infiniband network, allowing for highly efficient message passing. The cluster consists of two 10 Gbps Ethernet switches and five Infiniband Host channel adapters that supports 4 × QDR.

## 2.2 Software

The Operating System running on the nodes is RHEL Server 5.6 with the 2.6.18-238.el5, 2.6.18-238.el5xen kernel. The cluster includes IBRIX parallel File System, the software component of the IBRIX is combined with the HP X9000 series of storage systems. There are three MPI implementations available in HPC cluster: OpenMPI, Intel MPI and MPICH2. Among these Intel-MPI was chosen for the experiment. To manage the MPI jobs, PBS-PROFESSIONAL 12.0.1 job-scheduler was used. All the mpiBLAST jobs were submitted through PBS-Scripts. The latest version of mpiBLAST-1.6.0 available at mpiBLAST website was compiled and installed. NCBI-BLAST was compiled from version 2.2.20, downloaded from the ftp site of NCBI.

## 2.3 Experiment Data

9.38 GB (in compressed form) nr database was downloaded from the NCBI-BLAST website. Database used in BLAST was in text file FASTA format. The formatting and partitioning of the database into 24 segments, 48 segments, 96 segments and 192 segments of approximately equal size was done by the command 'mpiformatdb'. In this experiment, 200 nucleotide sequences of BADH were used as query file of size 240 KB. The computational model was based on data parallelism, utilizing master-worker paradigm and MPI-IO was used for data exchange between parallel-processes which were scheduled to run by PBS.

# 3. RESULTS & DISCUSSIONS

All the graphs presented in this paper describe the performance of blastx (querying a nucleotide sequence against a protein database), using nucleotide BADH query file against the protein nr database on High Performance Linux Cluster. As a general goal, parallel performance parameters (execution time, Speedup, efficiency) were estimated experimentally.

## 3.1Parallel-write v/s Master-write

As a significant portion of the non-search fraction of the total Blast runtime is dependent on the writing of the results, test was conducted to compare the writing performance of mpiBLAST on a high performance parallel file system. In case of master-write, the master process receives the sequence data from the slaves that they have cached in their buffers and writes the output file sequentially. When parallel-write is activated, the slaves become responsible for writing the output file in parallel, in the offsets designated by the master.

Figure 1 shows the execution time taken by the master-write as compared to parallel-write option when run with 24 database fragments. The graph in figure 1 depicts that parallel-write is faster than master-write as number of processes increases. Parallel-writing of the results by the slaves can evolve as an efficient solution to the problem of I/O. The graph below gives a clear picture that with increase in the number of cores the difference in execution time of master-write and parallel-write is significant. Parallel-write outperform master-write when mpiBLAST was executed up to 384 cores i.e. 64 processors.

**Table 1: Execution time (in sec) of mpiBLAST with different writing options**

| No. of cores | Master write | Parallel write |
|---|---|---|
| 24 cores | 7916.64 | 7674.17 |
| 48 cores | 2840.03 | 2646.40 |
| 96 cores | 1501.87 | 1352.45 |
| 192 cores | 1005.02 | 783.264 |

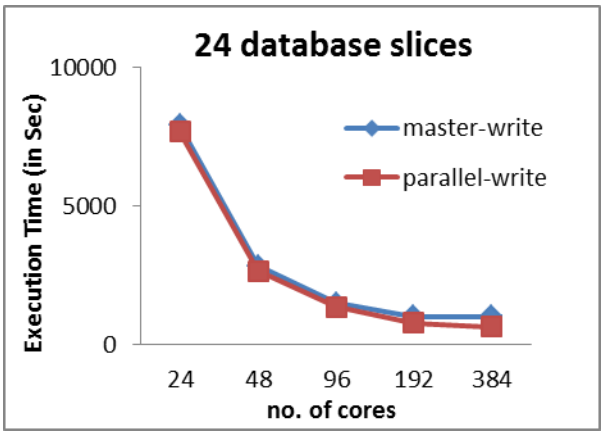


**Fig 1: Comparison of writing performance of mpiBLAST**

## 3.2 Load Balancing

In this test, the behavior of mpiBLAST regarding the distribution of its fragments among slaves was observed and the effects of different rates between the number of processes and the number of fragments were analyzed. In this study, the sequence-search was performed on 24 cores up to 192 cores and the number of fragments was raised progressively from 24 to 192 to find the best fragment number in which database be divided to achieve improved performance. In each case, the number of fragments was either equal to or an integral multiple of the number of slaves.

From the figure 2 below, it can be observed that segmenting database into 24 fragments was an adequate option. It was observed that when the number of fragments increased, both the search and non-search time increased. As the size of the fragment to be searched per processor became very small, the result combination step became greater than the actual search time.

**Table 2: Execution time (in sec) of mpiBLAST parallel-write on different database fragments.**

| No.of fragments | 24 cores | 48 cores | 96 cores | 192 cores |
|---|---|---|---|---|
| 24 fragments | 7674.17 | 7690.24 | 7741.61 | 7828.48 |
| 48 fragments | 2646.40 | 3920.56 | 4003.93 | 4741.76 |
| 96 fragments | 1352.45 | 1572.115 | 1983.3 | 2508.38 |
| 192 fragments | 783.264 | 791.789 | 838.224 | 1053.5 |

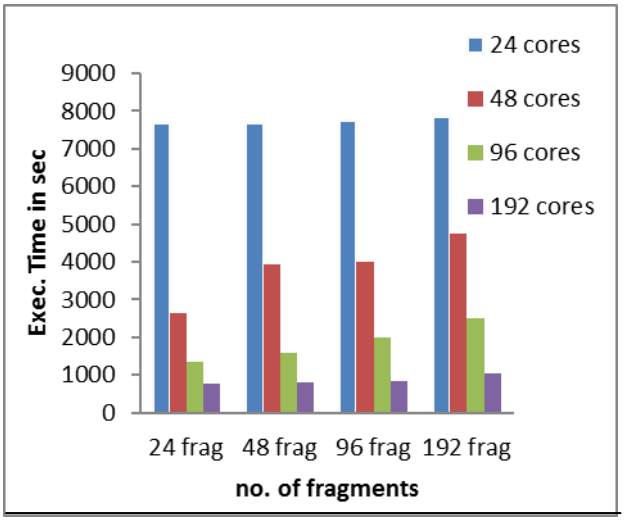


**Fig 2: Execution time for different fragment size**

## 3.3 Query-Distribution

mpiBLAST-PIO provides the ability to segment query sequence file and distribute the sequences among processes by using the option --use-segment-size available in mpiBLAST-1.6 .In mpiBLAST-PIO, the master node reads the query file, counts the number of sequences and distributes them to the workers and then writes to a temporary file for each worker. In this test, the master process fetches five sequences at a time. This experiment was conducted to compare the performance of mpiBLAST with parallel-write and mpiBLAST parallel-write along with query-segmentation. The experiment was executed on 24, 48, 96, 192 number of fragments and number of processes were increased from 24 cores to 192 cores. The graph shown below indicates the results obtained after running the above experiment.

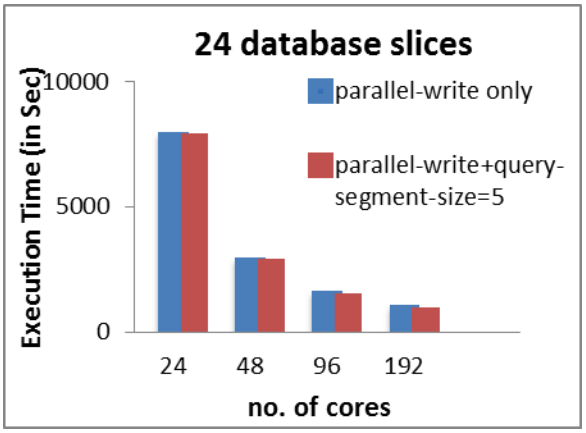


**Fig 3: parallel-write v/s parallel write + query-distribution time with 24 fragments**

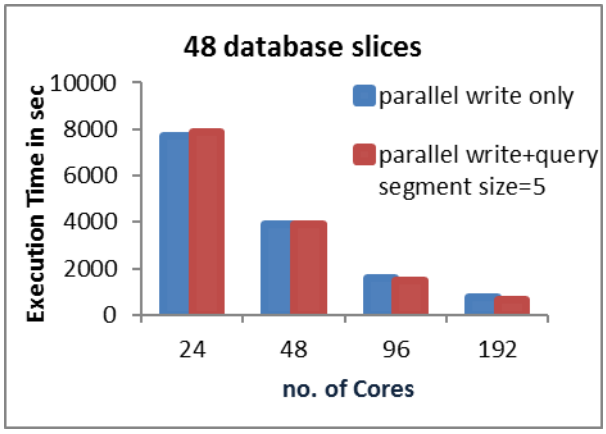


**Fig 4: parallel-write v/s parallel write + query-distribution time with 48 fragments**

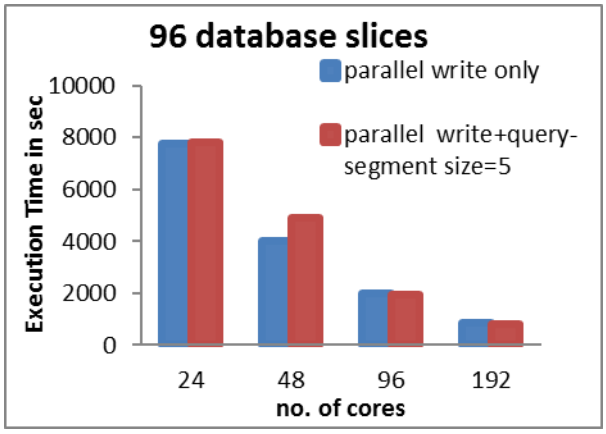


**Fig 5: parallel-write v/s parallel write + query-distribution time with 96 fragments**

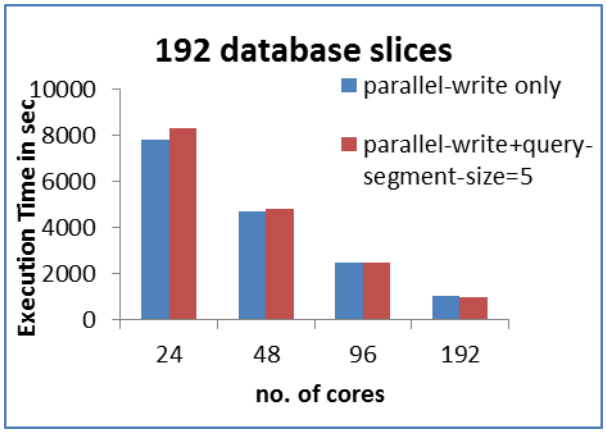


**Fig 6: parallel-write v/s parallel write + query-distribution time with 192 fragments**

After analyzing the above given four graphs, it was found that query-segmentation improved the performance only when the number of cores became equal to or greater than the number of fragments. The test results conclude that the execution time of query-distribution was more in comparison to the parallel-write till the number of cores was less than number of fragments after this point execution time started decreasing.

## 3.4 Multiple Masters

By activating the --partition-size flag of mpiBLAST-PIO, performance of hierarchical scheduling with multiple masters33 was evaluated. A second level of management was introduced, in which the number of workers can be limited for the master by creating groups of nodes containing one master for each group working on separate query sequences. So as to prevent the groups from waiting for the queries from the SuperMaster, a minimum number of queries to distribute among the group masters were set using the option –query-segment-size. In this experiment, the query-segment-size was set to 5. Values of the tables given below are plotted in graph Figure 7 for analysis.

**Table 3: Execution time (in sec) of mpiBLAST-PIO with multiple masters on 24 cores**

| No. of database fragments | partition-size= 12 | partition-size= 24 |
|---|---|---|
| 24 fragments | 6948.69 | 7913.56 |

**Table 4: Execution time (in sec) of mpiBLAST-PIO with multiple masters on 48 cores**

| No. of database fragments | partition-size= 24 | partition-size= 48 |
|---|---|---|
| 24 fragments | 2646.39 | 3005.39 |

**Table 5: Execution time (in sec) of mpiBLAST-PIO with multiple masters on 96 cores**

| No. of database fragments | partition-size= 48 | partition-size= 96 |
|---|---|---|
| 24 fragments | 1209.01 | 1387.65 |

**Table 6: Execution time (in sec) of mpiBLAST-PIO with multiple masters on 192 cores**

| No. of database fragments | partition-size= 96 | partition-size= 192 |
|---|---|---|
| 24 fragments | 684.809 | 687.195 |

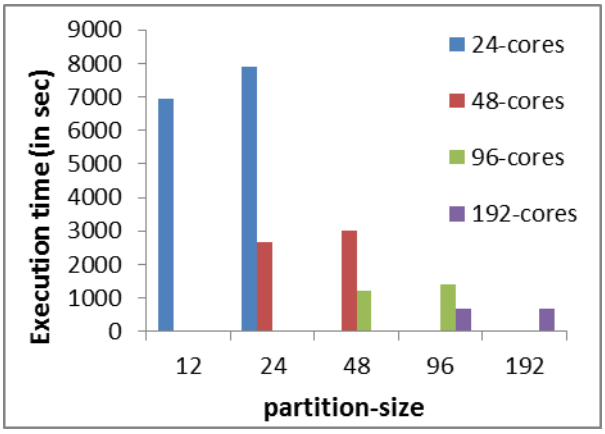


**Fig 7: execution time of mpiBLAST when number of database fragments =24**

After analyzing the figure 7, it was observed that the performance of mpiBLAST-PIO was improved when the partition-size was set to half the number of cores. The execution time increased when partition-size was equal to number of cores (MPI processes) as larger partition sizes overburden the master process with increasing loads of scheduling and output coordination, and incurs higher parallel overhead.

## 3.5 Speedup

The Speed-up is defined and evaluated as the ratio of the time for executing the sequential code of NCBI-BLAST on single core to the time of execution of parallel algorithm mpiBLAST-PIO with parallel-write enabled.

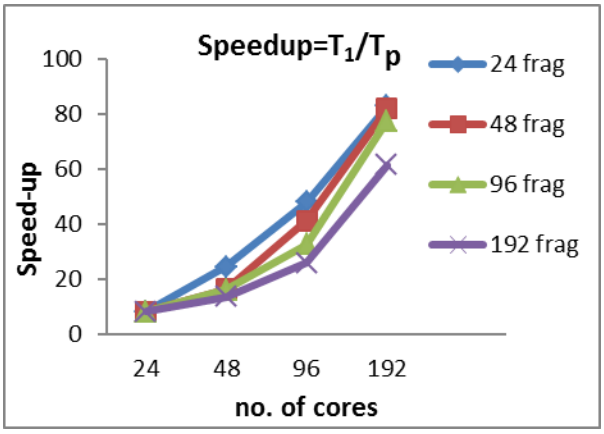


**Fig 8: Speedup**

With this graph in figure 8, it is possible to observe that execution of mpiBLAST-PIO on 24 fragments was super-linear. The performance gain significantly improved when number of processes were raised from 96 to 192 (32 number of processes). The diagram presents that mpiBLAST scaled well up to 192 processes.

## 3.6 Efficiency

**Table 7: Efficiency**

| p | 24 frag | 48 frag | 96 frag | 192 frag |
|---|---|---|---|---|
| 24 cores | 0.352204 | 0.351468 | 0.349136 | 0.345262 |
| 48 cores | 0.510669 | 0.344705 | 0.337528 | 0.285008 |
| 96 cores | 0.499626 | 0.429815 | 0.340704 | 0.269384 |
| 192 cores | 0.431348 | 0.4267038 | 0.4030657 | 0.3207018 |

The above given Table 7 shows the maximum efficiency achieved, in case of 24 database fragments searched on 48 numbers of cores. Afterwards, it started decreasing. In case of searching 48 database fragments, best efficiency was achieved on 96 cores. In 192 fragments, efficiency improved up to 192 cores. From the table, it can be concluded that performance was enhanced when number of fragments to be searched were executed on double the number of processing cores.

# 4. CONCLUSION

In this research, several experiments of mpiBLAST-PIO were performed with different options activated, to achieve high performance parallel BLAST implementation. As a general goal, parallel performance parameters like execution time, speed-up and efficiency were estimated experimentally. Tests were conducted to compare the writing performance of mpiBLAST using IBRIX parallel file-system. The results showed that, option –use-parallel-write provided performance gain as compared to master-write. This study also investigated that large number of database fragments executed on different number of processors degraded the performance of sequence-search, where number of fragments were equal to or integral multiple of the number of slaves. In this study, extensive performance evaluation was carried out on hierarchical architecture using different partition-size on different number of processes. Different techniques of handling I/O by using MPI I/O interface, efficient database distribution, query-segmentation, load-balancing and multiple-masters strategy showed improvements. This study demonstrated that mpiBLAST-PIO scaled well up to 192 MPI processes (32 processors). In this research, maximum efficiency achieved was 51% when 24 fragments were searched on 48 cores.

# 5. ACKNOWLEDGMENTS

Authors are indebted to express deep gratitude to Mr Sanjiv Tiwari of Locuz Enterprise and Mr Inderjit Singh Yadav of Biotechnology department of PAU for their support. We would like to thank the teachers of our department viz Mr Amarjeet Singh, Mr Arun Kumar who helped in maintaining the HPC system work smoothly.

# 6. REFERENCES

[1] Borovska P, Gancheva V and Markov S 2011. Parallel performance evaluation of sequence nucleotide alignment on the Supercomputer BlueGene/P. In Proceedings of the European Computing Conference, Wisconsin, USA. Pp 462-467.

[2] Borovska P, Nakov O, Gancheva V and Georgiev I 2010. Parallel genome sequence searching on supercomputer BlueGene/P. In Proceedings of ECS'10/ ECCTD'10/ ECCOM'10/ ECCS'10. Pp: 27-31.

[3] Correa J C and Silva G P 2011. Parallel BLAST analysis and performance evaluation. In Proceedings of the BICOB-2011, University of Houston, New Orleans, Louisiana, USA

[4] Darling A E, Carey L and Feng W 2003. The Design, implementation and evaluation of mpiBLAST. ClusterWorld Conference & Expo and the 4th International Conference on Linux Clusters: The HPC Revolution 2003.

[5] Feng W 2003. Green Destiny + mpiBLAST = Bioinformagic. 10th InternationalConference on Parallel Computing: Bioinformatics Symposium.

[6] Gardner M K, Feng W, Archuleta J, Lin H and Ma X 2006. Parallel genomic sequence searching on an Ad-Hoc grid: Experiences, Lessons Learned and Implications. SC'06 Proceedings of ACM/IEEE conference on supercomputing, Tampa, Florida, USA.

[7] Kent W J 2002. "Blat- The BLAST-Like Alignment Tool", Genome Research. Volume no. 12 Pp: 656-664.

[8] Lin H, Ma X, Chandramohan P, Geist A and Samatova N 2005. Efficient data access for Parallel BLAST. 19th IEEE International Parallel and Distributed Processing Symposium, April 3-8, 2005 in Denver, Colorado. Volume no. 01 Pp: 72-82.

[9] Lin H, Ma X, Feng W and Samatova N F 2011. "Coordinating computation and I/O in massively parallel sequence search". In IEEE Transactions on Parallel & Distributed SystemsVolume no. 22 Pp: 529-543

[10] Mathog R D 2003. "Parallel BLAST on split databases", Oxford University Press. Volume no. 19 Pp: 1865-1866.Brown, L. D., Hua, H., and Gao, C. 2003. A widget framework for augmented interaction in SCAPE.

[11] Mulhem M A and Shaikh R A 2013. "Performance modelling of parallel BLAST using Intel and PGI compilers on an infiniband-based HPC cluster", International Journal of Bioinformatics Research and Applications, Volume no. 9, pp 534 (Abstr).

[12] Muralidhara B L 2013. "Parallel two master method to improve BLAST algorithm's performance", International Journal of Computer Applications, Volume no. 63 pp: 0975-8887.

[13] Pedretti K T, Braun R C, Casavant T L, Scheetz T E, Birkett C L and Roberts C A 2001. Parallelization of local BLAST service on workstation clusters. In Future Generation Computer Systems. Volume no. 17 pp : 745-754.

[14] Rangwala H, Lantz E, Musselman R, Pinnow K, Smith B and Wallenfelt B 2005. Massively Parallel BLAST for the Blue Gene/L. High Availability and Performance Computing Conference.

[15] Sait S M, Mulhem M A and Shaikh R A 2011. Evaluating BLAST runtime using NAS based high performance clusters. In Proceedings of the CIMSIM'11, Langkawi, Malaysia Pp**:** 51-56.

[16] Sosa C P, Thorsen O, Smith B, Jiang K, Lin H, Peters A and Feng W C 2007. Parallel genomic sequence search on a massively parallel system. CF'07,Ischia, Italy.Pp:59-68

[17] Sousa D X D, Lifschitz S and Valduriez P 2008. BLAST parallelization on partitioned databases with primary fragments. High Performance Computing for Computational Science- VECPAR 2008, Toulouse, France Volume no. 5336 pp: 544-554.

[18] Yang C T and Kuo Y L 2003. "Apply Parallel bioinformatics applications on Linux PC Clusters", Tunghai Science. Pp: 125-141.

[19] Zomaya A Y (ed) 2006. Parallel Computing For Bionformatics and Computational Biology, John Wiley & Sons Inc, New Jersey. Pp 221-226.

[20] mpiBLAST website, http://www.mpiblast.org

[21] National Centre for Bioinformatics website: http://www.ncbi.nlm.nih.gov.