ISSN:0975-9646

Parallel Implementation & Performance Evaluation of Blast Algorithm on Linux Cluster

Nisha Dhankher , O P Gupta

School of Electrical Engineering & IT
COAE&T, PAU
Ludhiana, India

Abstract-The aim of this paper is to investigate the performance of parallel implementation of BLAST algorithm on HPC platform using Infiniband. This paper described the optimized and extended version of mpiBLAST called mpiBLAST-PIO. Due to high non-search overhead, parallel-writing the results by the slaves evolved as the efficient solution to the problem.

Keywords: Bioinformatics, mpiBLAST, HPC, Infiniband, Cluster Computing

I. INTRODUCTION

Genomic sequence-search is a basic problem of computational biology that has greatly benefited from parallel and distributed computing. The most widely used sequence-search tool is BLAST. BLAST is a fast program that efficiently calculates local pairwise alignment based on approximation. Through sequence alignment (or sequence comparison) of two biological sequences, researchers can find evolutionary information about a new sequence. Similarities between newly discovered sequence and a known sequence can help in determining functions of the new sequence and find sibling species from common ancestor.

There are two types of sequence alignment problems: global and local. The global alignment algorithm finds the best match between the entire sequences whereas the local alignment algorithm finds the best match between parts of the sequences. The first algorithms devised for sequence-alignment were Needleman Wunsch (1979) and Smith Waterman (1981). These were based on dynamic programming and produce optimal solutions but had time complexity $O(n^2)$.

As a result, heuristic based BLAST algorithm was proposed by Altschul *et al* in 1990. BLAST searches a query sequence containing DNA or proteins against a database of known nucleotide or peptides sequences in linear time using a statistical model. BLAST heuristic search, first, breaks the query into words of length w (by default w=3) and compare them to each database sequence. The matching words (or seeds) are then extended in both the direction until the score of alignment drops below a threshold to form the High Scoring Segment Pair (HSP). BLAST2 uses 2 -hit alignment algorithm to find the top-scoring HSP's which are combined to form consistent local alignment. BLAST's final result consists of a series of local alignments, ordered by the similarity score along with an e-value. BLAST program has the capability to compare all possible combinations of query and database sequence types by translating them. BLAST search types are:

1. blastn: search nucleotide database using a nucleotide query.
2. blastp: searches protein database using a protein query.
3. blastx: search protein database using a translated nucleotide query.
4. tblastn: search translated nucleotide database using a protein query.
5. tblastx: search translated nucleotide database using a translated nucleotide query.

Recent advances in molecular biology techniques, has led to the exponential growth of sequence databases. Although CPU architectures are struggling to show better performance, traditional techniques to sequence homology searches using BLAST have proven to be slow to keep up with the current rate of sequence acquisition (Kent 2002). As BLAST is both computationally intensive and parallelizes well, many parallel and distributed approaches of parallelizing BLAST have been proposed by programmers.

The mpiBLAST Algorithm

mpiBLAST is a freely available open-source parallelization of National Centre for Biotechnology Information (NCBI) BLAST, which achieves super linear speedup by segmenting a BLAST database. It is designed to work on a computer cluster using MPI library and adopts a master-slave style. (Darling *et al* 2003) The mpiBLAST algorithm consists of three steps:

1. Segmenting and distributing the database,
2. Running mpiBLAST queries on each node,
3. Merging the results from each node into a single output.

Before mpiBLAST search, the database is formatted and segmented using a wrapper called mpiformatdb and placed at shared storage. mpiBLAST enables the master node to assign the query sequence and database fragment to each worker node. The worker nodes perform the BLAST search on queries and send the results to the master node. When one worker node complets its task, the master node assign a new fragment to it. This procedure is repeated until all the queries have been searched. The master node merge all the results and sorts them according to score. Results written in output file can be in any format including XML, HTML, simple text, ASN.1.

However, mpiBLAST suffers from non-search overheads with increasing number of processors and varying database sizes. So, Lin *et al* 2005 proposed pio-

BLAST that stands for parallel I/O BLAST and uses MPI-IO for efficient data access. MPI-IO enables multiple processors to read or write files simultaneously. (Correa and Silva 2011) One of pioBLAST's main updates was the caching of sequences by worker nodes as they find potential alignments in their partial results. Due to parallel writing of output, pioBLAST greatly improved the performance. As a result, some of the pioBLAST's enhancements were added to mpiBLAST, resulting in the development of mpiBLAST-PIO, which is the official version of mpiBLAST since release 1.6 (Lin H *et al* 2005).

mpiBLAST-PIO (Thorsen *et al* 2007, Borovska *et al* 2010) is 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.

This paper presents the experience in mapping and evaluating both the serial and parallel BLAST algorithm onto a infiniband based HPC.

II. Material and Methodology

Cluster Hardware

All the experiments were run on a HPC Linux cluster installed at Data Centre, SEEIT, PAU. The cluster is composed of 40 compute nodes, each with two hexa-core Intel, Xeon 2.93 GHz processors, 12 MB cache , 50 GB RAM means that there are 480 cores available for processing. 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.

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. 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 through ftp site of NCBI.

Experiment Data

9.38 GB nr database in compressed form was downloaded from the NCBI-BLAST website through ftp. The formatting and partitioning of the database into 24 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 was used for data exchange between parallel-processes which were scheduled to run by PBS.

III. Results and Tables

In this paper, mpiBLAST-PIO performance was evaluated by measuring speedup and efficiency in comparison to sequential NCBI BLAST version. The algorithm run on HPC was blastx that compared the nucleotide query sequences with the NR database. As mpiBLAST-1.6.0 allow to run both parallel write and master write, writing performance of mpiBLAST in high performance parallel file system was compared.

In master write, the master process is responsible for sorting the intermediate results, according to score and write the final output in the file sequentially. Master-write has two drawbacks. First, the result processing was serialized by the master. Second, the master memory may not be large enough to buffer all intermediate results and corresponding sequence data. To address the above problems, parallel-write was activated. In case of parallel-write, workers after searching their fragment, convert their intermediate results into the final output and send the final output metadata to the master. As the size of each result alignment output is known to master, it computes the offset ranges for each record and send the information to the workers. With the output offsets, workers write the local output records in parallel using the MPI-IO interface. By locally buffering output and parallel processing the results, mpiBLAST-PIO removes the performance bottleneck.

In this test, number of database fragments was fixed to 24 and number of workers was increased from 24 to 384 cores. The number of fragments was either equal to or an integral multiple of the number of cores.

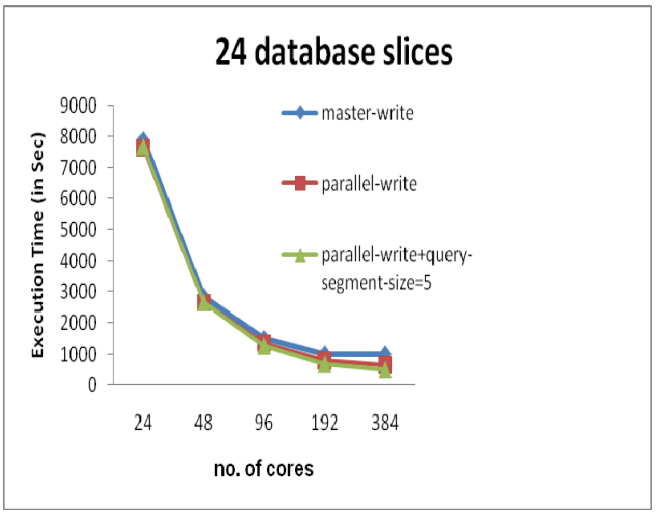

Figure1

In the above figure, performance of master-write, parallel-write and parallel-write along with query segmentation size set to 5, were compared. Figure shows that parallel write performed faster than the master write. It was also observed that query distribution along with database segmentation yields good results with increasing number of processes. In figure 2, Speedup of the mpiBLAST-PIO program was plotted against the number of cores. In this study, the Speedup was defined and evaluated as the ratio of time to run sequential algorithm NCBI-BLAST on single core to the time taken to run

parallel algorithm mpiBLAST-PIO. From this evaluation, it was concluded that mpiBLAST-PIO achieves super-linear speedup when number of slaves were increased up to 384. The experiment results demonstrated that maximum efficiency achieved was 51% when mpiBLAST-PIO was executed on 48 cores with 24 database fragments. After this point, efficiency starts decreasing.

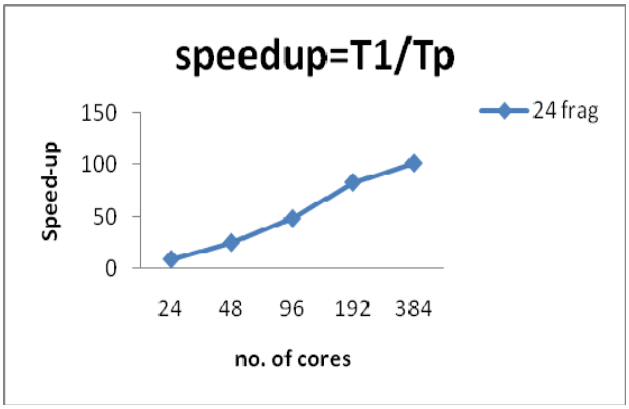

Figure 2

IV. Conclusion

This paper evaluated I/O and communication related optimizations for parallel sequence search using BLAST. Such optimizations include the use of parallel-write option for efficient handling of I/O. This paper showed that mpiBLAST-PIO gave performance gain over mpiBLAST with master write. Also, use of query distribution along with database segmentation resulted in reduced execution time. mpiBLAST-PIO gave optimum efficiency when number of cores was double the number of database fragments.

Acknowledgement

We would like to express our gratitude to Sanjiv Tiwari of Locuz Enterprise and Inderjit Singh Yadav of Biotechnology department for helping us in the work. We would like thank our department teachers Amarjeet Singh, Arun who helped in maintaining the HPC system work smoothly.

References

i. Archuleta J, Balaji P, Coghlan S, Feng W, Foster I, Jha S, Katz D S, Lin H, Lusk E, Matsuoka S ,Reed D, Setubal J, Shinpaugh K, Thakur R and Warren A (2008) Distributed I/O with ParaMEDIC: Experiences with a Worldwide Supercomputer. *Proceedings of ISC'08.*

ii. Archuleta J, Feng W, Gardner M K, 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.

iii. Balaji P, Feng W, Archuleta J, Lin H, Kettimuthu R, Thakur R and Ma X (2008a) Semantics-based Distributed I/O for mpiBLAST. *PPoPP'08.* Pp 293-294.

iv. Balaji P, Feng W and Lin H (2008b) Semantic based Distributed I/O with the ParaMEDIC Framework. *HPDC'08*. pp 175-184. Boston, Massachusetts, USA.

v. Braun R C, Pedretti K T, Casavant T L, Scheetz T E, Birkett C L and Roberts C A (2001) Parallelization of local BLAST service on workstation clusters. *Future Generation Computer Systems* **17**:745-754.

v. Lin H, Balaji P, Feng W C, Ma X, Poole R and Sosa C (2008) Massively Parallel genomic sequence search on the Blue Gene/Parchitecture. *SC2008.* Austin, Texas, USA.

vi. Borovska P, Gancheva V and Markov S (2011) Parallel performance evaluation of sequence nucleotide alignment on the Supercomputer BlueGene/P. *Proceedings of the ECC'11.* Pp 462-467.

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

viii. Carey L, Darling A E and Feng W (2003) The Design, implementation and evaluation of mpiBLAST. *Proceedings of the 4th International Conference on Linux Clusters.*

ix. Chandramohan P, Geist A, Lin H, Ma X and Samatova N (2005) Efficient data access for Parallel BLAST. *19th IEEE International Parallel and Distributed Processing Symposium* **01:**72-82.

x. Correa J C and Silva G P (2011) Parallel BLAST analysis and performance evaluation. *Proceedings of the BICOB-2011.*

xi. Feng W (2003) Green Destiny + mpiBLAST = Bioinformagic. *10th International Conference on Parallel Computing: Bioinformatics Symposium*

xii. Feng W, Lin H, Ma X and Samatova N F (2011) Coordinating computation and I/O in massively parallel sequence search. *IEEE Transactions on Parallel & Distributed Systems* **22:**529-543.

xiii. Feng W C, Jiang K, Lin H, Peters A, Smith B, Sosa C P and Thorsen O (2007) Parallel genomic sequence search on a massively parallel system. *ACM Proceedings 4th International Conference on Computing Frontiers***.** Pp: 59-68. Ischia, Italy.

xiv. Kent W J (2002). Blat- The BLAST-Like Alignment Tool. *Genome Research* **12**:656-664.

xv. Kuo Y L and Yang C T (2003) Apply Parallel bioinformatics applications on Linux PC Clusters. *Tunghai Science.* Pp: 125-141.

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

xvii. Lifschitz S, Sousa D X D and Valduriez P (2008). BLAST parallelization on partitioned databases with primary fragments. *High Performance Computing for Computational Science- VECPAR* **5336:**544-554.

xviii. Mathog R D (2003) Parallel BLAST on split databases. *Oxford University Press* **19**:1865-1866.

xix. Mulhem M A, Sait S M and Shaikh R A (2011) Evaluating BLAST runtime using NAS based high performance clusters. *Proceedings of the CIMSIM'11.* Pp**:**51-56.

xx. 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* **9**:534 (Abstr).

xxi. Muralidhara B L (2013) Parallel two master method to improve BLAST algorithm's performance. *International Journal of Computer Applications* **63:**0975-8887.

xxii. Oehman C S and Baxter D J (2013) ScalaBLAST 2.0: rapid and robust BLAST calculations on multiprocessor systems. *Bioinformatics oxford* journals **29**:797-798 .

xxiii. Zomaya A Y (ed) (2006). *Parallel Computing For Bionformatics and Computational Biology*. pp 221-226. John Wiley & Sons Inc, New Jersey.

xxiii. mpiBLAST website, http://www.mpiblast.org

xxiv.National Centre for Bioinformatics website: http://www.ncbi.nlm.nih.gov/nuccore