Parallel Genome Sequence Searching on SupercomputerBlueGene/P

Plamenka Borovska, Ognian Nakov, Veska Gancheva, Ivailo Georgiev

***Abstract*—**The aim of this paper is to investigate the efficiency of sequence alignment. The parallel computational model is based on mpiBlast algorithm, data decomposition and manager-worker algorithmic paradigm utilizing massage passing based on MPI and is verified on the basis of flat parallel program implementation on a supercomputer BlueGene/P.

***Keywords*—**Biocomputing, High Performance Computing, mpiBLAST, Parallel Performance, Searching of Sequences.

I. Introduction

BIOCOMPUTING and molecular biology are areas, demanding knowledge and skills for acquisition, storing, management, analysis, interpretation and dissemination of biological information. This requires the utilization of high performance computers and innovative software tools for the management of the vast information, as well as the deployment of innovative algorithmic techniques for the analysis, interpretation and prognostication of data in order to get to the insight of the design and validation of life-science experiments.

The recent whole genome sequencing technology made it possible to reveal the nucleotide sequence of more than 1500 viral, bacterial, plants and animal genomes after the year 2000. The world DNA databases are accessible for common use and usually contain information for more than one (up to several thousands) individual genomes for each species.

The huge sequences of biological data, being accumulated in data bases, require the development of efficient tools for genome sequences comparative analysis and use powerful supercomputers. Size of genomes varies considerably depending on the type of organism. The human genome (over 3.4 billion bases) is much larger than the flu (over 13 000 bases). The size of file stored the information of the sequenced human genome is 9 GB.

Manuscript received October 31, 2010. This work was supported by the National Science Fund, Bulgarian Ministry of Education and Science, Grant DO02-115/2008.

P. Borovska is head of the Department of Computer Systems in Technical University of Sofia, Bulgaria (e-mail: pborovska@tu-sofia.bg).

O. Nakov is dean of the Faculty of Computer Systems and Control in Technical University of Sofia, Bulgaria (e-mail: nakov@tu-sofia.bg).

V. Gancheva is with the Department of Programming and Computer Technologies in Technical University of Sofia, Bulgaria (e-mail: vgan@tu-sofia.bg).

I. Georgiev is PhD student in Department of Computer Systems, Technical University of Sofia, Bulgaria (e-mail: ivailo_georgiev@tu-sofia.bg)

The information, obtained as a result of the genome sequence analyses, has various applications. There exist various tools for analyzing such sequences. In spite of that, deploying these tools to the analysis of large sets genome sequences data is un-applicable for single-processor machines. The major shortcomings of the purely experimental approach to searching and analyzing imply the significant financial expenditures (tens of millions of euro) and the high prize of investigations. These problems may be solved presently only by means of utilizing the modern methods of information technologies. Therefore there is a need to increase the effectiveness of popular search tools, access, retrieval and analysis of sequences of parallel cluster computers.

An investigation of this type is of exclusive significance in the era of high performance computing, where the paradigm shifts from conventional computers to the more efficient computer clusters.

Bioinformatics algorithms used in processing wide range data require large computational resources. They are easily yielded to parallelization and well scalable

The goal of this paper is to investigate the efficiency of sequence alignment on supercomputer BlueGene/P. The parallel computational model is based on mpiBlast algorithm, data decomposition and manager-worker algorithmic paradigm utilizing massage passing based on MPI.

II. BLAST Algorithm and Parallelization

The Basic Local Alignment Search Tool (BLAST) program is by far the most widely used program to look at sequence alignments and similarities [1]. BLAST searches a database for sequences similar to your sequence (the “query” sequence) by using the 2-step approach and efficiently calculates local pairwise alignments between sequences, based on a sophisticated statistical model. BLAST is based on the premise of sequence similarity, that is, similarity between two DNA or protein can suggest similar structure or function based on a common ancestor. BLAST searches thus attempt to find sequences in the database that are “similar” to the query sequence, where similarity is calculated using methods that increase the likelihood of common structure or function.

There are the following several of BLAST:

- **blastn** compares a nucleotide query sequence against a nucleotide sequence database;
- **blastp** compares an amino acid query sequence against a protein sequence database;
- **blastx** compares the six-frame conceptual translation products of a nucleotide query sequence (both strands) against a protein sequence database. This translation is the simple conversion of a nucleotide string into six separate strings of aminoacids (one for each possible reading frame);
- **tblastn** compares a protein query sequence against a nucleotide sequence database dynamically translated in all six reading frames (both strands);
- **tblastx** compares the six-frame translation of a nucleotide query sequence against the six-frame dynamic translations of a nucleotide sequence database. This program is doing 36 comparisons (6x6) for each comparison between the query sequence and any of the target sequences in the database. This will of course reflect on the speed of the program, making this one the slowest of the pack. However, this simultaneous translation into protein of both the query (nucleotide) and the target database (also nucleotide), allows us to find more distantly related sequences.

BLAST [1] utilize heuristics approach for increasing the performance of the alignment searching by first looking for ungapped homologies, i.e. similarities due exclusively to mutations and not insertions or deletions.

A new criterion for triggering the extension of word hits, combined with a new heuristic for generating gapped alignments, yields a gapped BLAST program that runs at approximately three times the speed of the original [2]. In addition, a method is introduced for automatically combining statistically significant alignments produced by BLAST into a position-specific score matrix, and searching the database using this matrix. The resulting Position-Specific Iterated BLAST (PSI-BLAST) program runs at approximately the same speed per iteration as gapped BLAST, but in many cases is much more sensitive to weak but biologically relevant sequence similarities. PSI-BLAST is used to uncover several new and interesting members of the BRCT superfamily.

As the number of DNA and protein sequences in databases increases, it is increasingly important to be able to create comparison and sequence alignments for very large sequences.

There is an increasing use of massively parallel system, computer cluster or network of personal computers as distributed memory parallel machines for solving large scale computational problems. Such they are usually programmed with the help of a message passing API like Message Passing Interface (MPI).

Recent advances in parallelization of biological sequence search applications have enabled bioinformatics researchers to utilize high-performance computing platforms and, as a result, greatly reduce the execution time of their sequence database searches. However, existing parallel sequence search tools have been focusing mostly on parallelizing the sequence alignment.

Several parallel BLAST algorithms for alignment search have been reported in recent years.

mpiBLAST [3] was developed at Los Alamos National Laboratories. This work introduced the database fragmentation strategy in the context of BLAST. It is an open source project that uses the standard message passing protocol (MPI) for its parallel BLAST implementation. It works on a wide range of clusters and supercomputers and has gained popularity amongst members of the bioinformatics community needing a high performance BLAST.

Scientifics have been using mpiBLAST, a popular, parallel, bioinformatics package that runs on a computer cluster, for their research activities [4], [5], [6], [7].

A set of techniques for efficient and flexible data handling in parallel sequence search applications are presented in [8] and demonstrate optimizations through improving mpiBLAST, an open-source parallel BLAST tool. These optimization techniques aim at enabling flexible database partitioning, reducing I/O by caching small auxiliary files and results, enabling parallel I/O on shared files, and performing scalable result processing protocols. As a result is reducing mpiBLAST users' operational overhead by removing the requirement of prepartitioning databases.

The paper [9] presents experiences in mapping and optimizing genomic sequence search onto the massively parallel IBM Blue Gene/P (BG/P) platform. Specifically, mpiBLAST has been optimized on numerous supercomputing environments. The authors identify several critical performance issues and propose and study different approaches for mapping sequence-search and parallel I/O tasks on such massively parallel architectures.

The focus of the paper [10] is to explain research involved with running a parallel implementation of the widely used BLAST algorithm on thousands of processors, available on supercomputers like the IBM Blue Gene/L. Their work involved optimally splitting up the set of queries as well as the database. They also found solutions to reduce the I/O thereby delivering a fast, high throughput BLAST.

mpiBLAST-PIO [11] is modified and extended version of parallel and distributed-memory version BLAST. It maps to a massively parallel system, specifically IBM Blue Gene/L. The extensions include a virtual file manager, a "multiple master" runtime model, efficient fragment distribution, and intelligent load balancing.

ScalaBLAST [12] accommodates very large databases and scales linearly to as many as thousands of processors on both distributed memory and shared memory architectures. ScalaBLAST relies on a collection of techniques – distributing the target database over available memory, multilevel parallelism to exploit concurrency, parallel I/O.

III. Experimental Framework

Our experimental framework is based on a Bulgarian IBM Blue Gene/P supercomputer, consisting of two racks, 2048 PowerPC 450 based compute nodes, 8192 processor cores and a total of 4 TB random access memory. Double-precision, dual pipe floating-point core acceleration is available on each core. Sixteen I/O nodes are connected via fibre optics to a 10 Gb/s Ethernet switch. When the system is enhanced with a new 10Gb/s switch in the near future, 16 more I/O nodes will be connected. The smallest partition size, available currently, is 128 compute nodes (512 processor cores). Maximum LINPACK performance achieved is Rmax= 23.42 Tflops. Theoretical peak performance is Rpeak= 27.85 Tflops. Energy efficiency is 371.67 MFlops/W. Furthermore cupboards with computing nodes supercomputing system include the following major components:

1. Front-End Node (bgfen): server to which users have access and which put out its tasks. The architecture is PowerPC 64 and Operation System - SuSE Linux Enterprise Server 10 (SLES 10);
2. Service Node (SN): server that manages the overall operation of the system
3. Two file server by which FEN and computing nodes have to access to the shared disk array with 12TB.

The Blue Gene/P architecture supports a distributed memory, message-passing programming model. Message passing is based on the MPICH2 distribution of the MPI standard.

IV. Parallel Computational Model

The suggested parallel computational model for sequence alignment based on mpiBlast and the experimental framework is shown in Fig.1. It is based on the parallel programming paradigm master-worker and utilizes data decomposition of the database between the slave processes.

mpiBLAST works by initially dividing up the database into multiple fragments so that each processor has a separate smaller fragment to work on. The searching of a fragment is independent of any other fragment lending a very parallelizable solution.

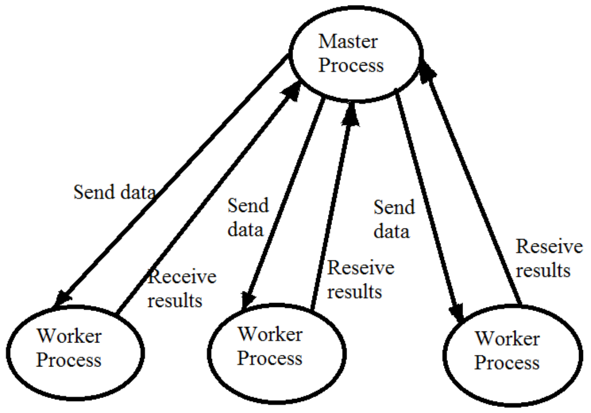


Fig. 1 parallel computational model for sequence alignment

The algorithm is implemented in two main steps. First, the database is segmented and stored on the shared memory. Second, mpiBLAST query runs on each node. If one node does not have a fragment of the database to search, a fragment form shared memory is copying. The tasks of each node are determine by an algorithm that minimizes the number of copies during each search.

On the Fig. 2 is shown the database segmentation and the query segmentation. In the case of database segmentation each node searches into the entire request against a fragment of the database. In the case of query segmentation, each node searches into the entire database using only part of the query.

The segmentation of the database is done through direct implementation of BLAST algorithm as implemented in the

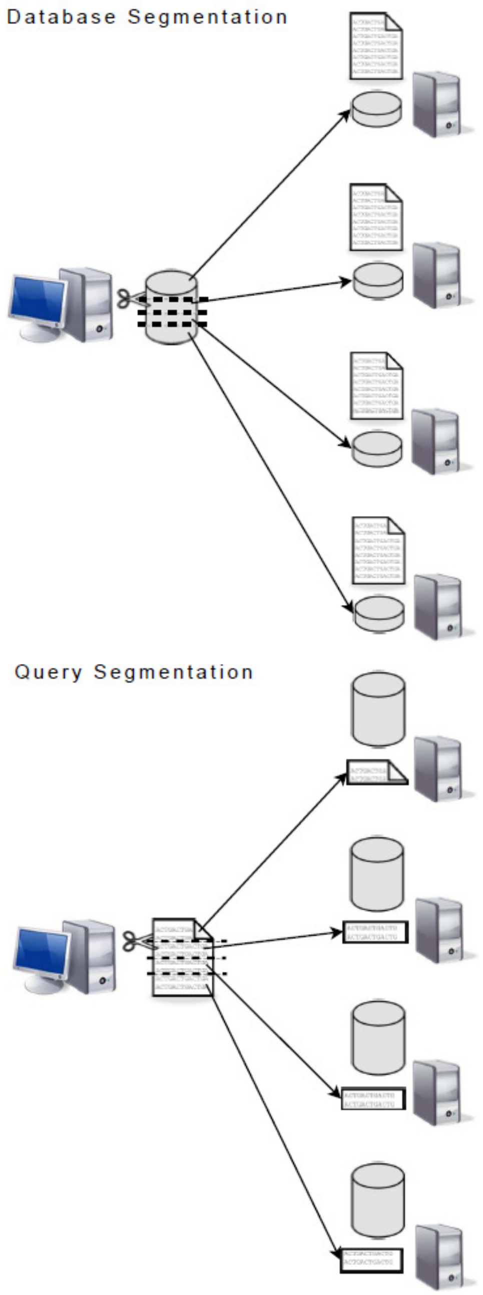


Fig. 2 database segmentation and query segmentation

NCBI library [14]. Each process worker report to the "master" which fragment of the database is already in the local memory. Then the process "master" (the one with rank 0), read sequences from the disk and sends them to all processes in the communication group. Where the query is complete, each process reports to the "master" that is empty. Then the "master" sends to "worker" fragment of the database to searching. This process is repeated until the entire fragmented database is searched.

The master process is responsible for the following tasks:

- reads the and send to the worker processes;
- broadcasts to the other nodes;
- decides which fragments each worker should search;
- sends a fragment of database to each worker process;
- receives the results from each worker;
- merge partial results from all workers;
- sorts them by score, and writes the output file;
- sends termination message when the task is fully processed;

Each worker processes performs the following operations:

- receives part of database from the master process;
- processes the query comparing it with the fragment of database;
- sends the comparison result to the master process;
- terminates when a termination message is received.

The formatting and dividing the database into many small fragments of approximately equal size is done by standard NCBI command, called mpiformatdb. Our database, obtained from GenBank [15] is fragmented into 64 segments. Upon successful completion of formatdb, formatted fragments are placed in shared memory.

The processing stage of the computational model requires each worker process independently to compare the query and part of the fragmented database. The communication stage requires the master process to distribute data to the worker processes, gather the results and send new parts of the database to the worker processes. Communication between the master and the worker processes is performed using synchronous communication routines of the message passing interface MPI.

The interaction phase comprises distributing data among the parallel processes, gathering the results and sending new partitions of data to the parallel processes. Each interaction phase deploys synchronous blocking communications implemented by means of the MPI functions *MPI_Send()* and *MPI_Recv()*.

V. Experimental Results and Analysis

A number of experiments have been carried out based on parallel program MPI-based implementations of the suggested parallel computational model for sequence alignment. We have used real data sets:.

➢ Database of nucleotide sequences: human_genomic

➢ Query of protein sequence: alpha_1_interferon_[Homo_sapiens]

➢ Query of nucleotide sequence: alcoholgene

Using formulas are:

Speedup = $T_{(1)}/T_{(p)}$

Parallel efficiency = Speedup(p)/p

TABLE I
SERIAL EXECUTION TIME

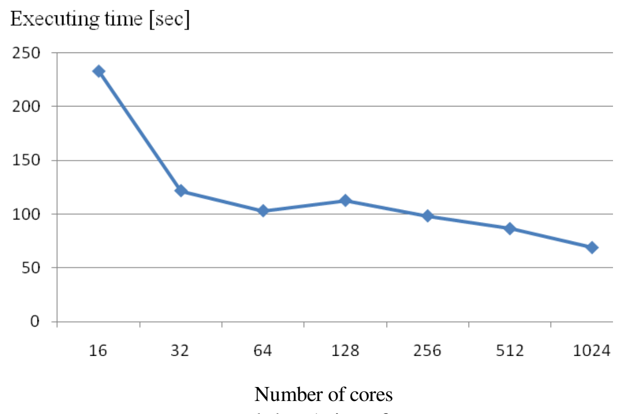
alpha_1_interferon Time [sec]	alcoholgene Time [sec]
3682,8	298058,24

The times in sec for the serial execution are shown in the Table 1.

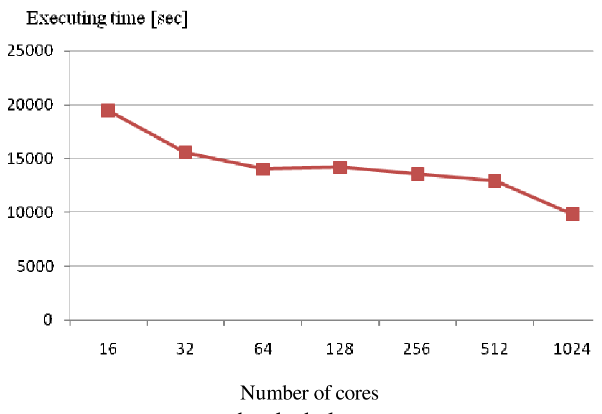
The parallel execution times for various sizes of queries and numbers of cores are shown in the Table 2 and Fig. 3.

TABLE 2
PARALLEL EXECUTION TIME

Query/ Number of cores	alpha_1_ interferon Time [sec]	alcoholgene Tine [sec]
16	232,65	19437,34
32	121,77	15562,76
64	103,16	14002,98
128	112,7	14167,54
256	98,35	13543,2
512	86,55	12912,2
1024	68,93	9833,54



a. alpha_1_interferon



b. alcoholgene

Fig. 3 execution time for various sizes of queries and number of cores

It is seen that the program is running much faster to increase the number of processors. alcoholgene is very larger query than alpha_1_interferon. This significant difference is reflected in the times to perform the tasks.

Interestingly, what is noticeable is the slight deviation from consistency of values in the case of implementing the program on 64 cores. In this case the program runs faster than in the case of 128 cores. This is due to the fragments of the database are also 64.

The speedup is defined and evaluated as the ratio of the time for executing the parallel code on a single processor to the time for parallel execution. The level of parallelization is higher if the speedup is greater.

Table 3 and Fig. 4 show the results obtained for the speedups for the various sizes of queries and number of cores.

TABLE 3
SPEEDUP

Query/ Number of cores	alpha_1_ interferon Speedup	alcoholgene Speedup
16	15,83	15,33
32	30,24	19,15
64	35,70	21,29
128	32,68	21,04
256	37,45	22,01
512	42,55	23,08
1024	53,43	30,31

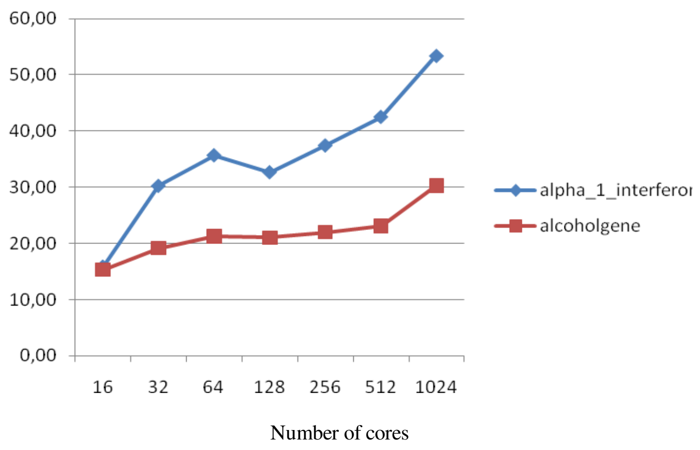


Fig. 4 scalability of the parallel system in respect of the speedup for various sizes of queries and numbers of cores

Obviously, the parallel system show good scalability in respect to both the number of cores and the workload size.

VI. CONCLUSION

In this paper we have performed investigation of the efficiency of sequence alignment. Parallel computational model based on mpiBlast algorithm, data decomposition and manager-worker algorithmic paradigm utilizing massage passing has been suggested. The proposed parallel model has been verified by flat parallel program implementations on a supercomputer BlueGene/P.

Conclusions based on the tests and analyses are that mpiBlast needs of large numbers of parallel processors.

Parallel mpiBLAST technique distributes data into worker nodes and sends the query for execution to all of them. After all the workers completed their part of the job, master completes the results.

Parallel performance parameters (execution time, speedup, scalability) have been estimated experimentally. The performance estimation and scalability analyses show that the suggested model has good scalability both in respect to the workload and number of cores.

REFERENCES

[1] S. Altschul, W. Gisha, W. Millerb, E. Meyersc, and D. Lipmana, “Basic local alignment search tool”, *Journal of Molecular Biology*, 215(3), 1990.
[2] S. Altschul, T. Madden, A. Schaffer, J. Zhang, Z. Zhang, W. Miller, and D. Lipman, “Gapped BLAST and PSIBLAST: a new generation of protein database search programs”, *Nucleic Acids Research*, 25:3389–3402, 1997.
[3] A. Darling, L. Carey, and W. Feng, “The design, implementation, and evaluation of mpiBLAST”, In Proceedings of the Cluster World Conference and Expo, in conjunction with the 4th International Conference on Linux Clusters: The HPC Revolution, 2003.
[4] W. Feng, “Green destiny + mpiblast = bioinfomagic”, In International Conference on Parallel Computing (ParCo), 2003.
[5] A. Zomaya (ed.), *Parallel Computing for Bioinformatics and Computational Biology: Models, Enabling Technologies, and Case Studies*, Wiley, 2006.
[6] R. Braun, K. Pedretti, T. Casavant, T. Scheetz, C. Birkett, and A. Roberts, “Three complementary approaches to parallelization of local BLAST service on workstation clusters”, Fifth International Conference on Parallel Computing Technologies (PaCT), Vol. 1662, Lecture Notes in Computer Science (LNCS), 1999.
[7] R. Costa and S. Lifschitz, “Database allocation strategies for parallel BLAST evaluation on clusters”, *Distributed Parallel Databases*, 13, 99–127, 2003.
[8] H. Lin, X. Ma, P. Chandramohan, A. Geist, and N. Samatova, “Efficient data access for parallel BLAST”, 19th Int. Parallel & Distributed Processing Symposium, April 2005.
[9] H. Lin, P. Balaji, R. Poole, C. Sosa, X. Ma, and W. Feng, “Massively parallel genomic sequence search on the Blue Gene/P architecture”, Proceedings of the ACM/IEEE conference on Supercomputing, 2008.
[10] H. Rangwala, E. Lantz, R. Musselman, K. Pinnow, B. Smith, and B. Wallenfelt, “Massively parallel blast for the Blue Gene/L. In High Availability and Performance Computing Workshop, 2005.
[11] O. Thorsen, K. Jiang, A. Peters, B. Smith, H. Lin, W. Feng, and C. Sosa, “Parallel genomic sequence-search on a massively parallel system,” in ACM International Conference on Computing Frontiers, New York, NY, USA, 2007.
[12] C. Oehmen and J. Nieplocha, “ScalaBLAST: A scalable implementation of BLAST for high-performance data-intensive bioinformatics analysis,” *IEEE Trans. Parallel Distrib. Syst.*, vol. 17, no. 8, pp. 740–749, 2006.
[13] *MPI: Message-passing interface standard*, Message Passing Interface Forum, 1995.
[14] National Center for Biotechnology Information, NCBI BLAST, http://www.ncbi.nih.gov/BLAST/
[15] GenBank http://www.ncbi.nlm.nih.gov/genbank/