Hybrid Parallel Multiple Sequence Alignment Based on Artificial Bee Colony on the Supercomputer JUQUEEN

Plamenka Borovska
Department of Informatics
Technical University of Sofia
Sofia, Bulgaria
e-mail: pborovska@tu-sofia.bg

Veska Gancheva
Department of Programming and Computer Technology
Technical University of Sofia
Sofia, Bulgaria
e-mail: vgan@tu-sofia.bg

Ivailo Georgiev
Institute of Microbiology
Bulgarian Academy of Science
Sofia, Bulgaria
e-mail: ivailo@microbio.bas.bg

Desislava Ivanova
Department of Informatics
Technical University of Sofia
Sofia, Bulgaria
e-mail: d_ivanova@tu-sofia.bg

***Abstract*—The paper focuses on performance investigation and improvement of multiple biological sequence alignment software MSA_BG on the BlueGene/Q supercomputer JUQUEEN. For this purpose, scientific experiments in the area of bioinformatics have been carried out, using as case study influenza virus sequences. The objectives of the investigation are code optimization, porting, scaling, profiling and performance evaluation of MSA_BG software. To this end we have developed hybrid MPI/OpenMP parallelization on the top of the MPI only code and we showcase the advantages of this approach through the results of benchmark tests, performed on JUQUEEN. The experimental results show that the hybrid parallel implementation provides considerably better performance than the original code.**

Keywords—artificial bee colony; BlueGene/Q; hybrid programming; high performance computing; multiple sequence alignment.

I. Introduction

The fundamental scientific studies are in revolution era by the big files and flows of data. One of the fields of the fundamental science, strongly dependent from the development of big data, is the field of molecular and computational biology [1]. In the biological sciences there are very well established practices of collecting data in the public and generally accessible data bases, which are used by the scientists from all over the world, working on concrete subjects. The development of the bioinformatics stimulates in high extent the methods for processing and analyzes of collected data. The technological progress, as well the next generation sequencing, yielded to exponential grow of size and number of experimental data, and as a result the well-known methods and technologies became not applicable to the new challenges of the big flows of data. Many scientific research teams are doing prognostics for the significance of big data, and most analyses for the period till 2025 list astronomy, molecular and computational biology, medicine and meteorology as directions of fundamental science, strongly dependent and influenced from the development of the big files and flows of data [2].

Multiple sequence alignment (MSA) is an important method for biological sequences analysis and involves more than two biological sequences, generally of the protein, DNA, or RNA type [3]. This method is computationally difficult and is classified as a NP-hard problem [4] [5] [6].

The innovative parallel algorithm MSA_BG for multiple alignment of biological sequences was proposed as a result of a previous study [7]. The MSA_BG algorithm is iterative and based on the concept of Artificial Bee Colony metaheuristics and the concept of algorithmic and architectural spaces correlation. The Artificial Bee Colony (ABC) algorithm is an optimization algorithm based on the intelligent foraging behavior of honey bee swarm [8]. In the ABC model, the colony consists of three groups of bees: employed bees, onlookers and scouts. The algorithmic framework of the designed parallel algorithm had already been constructed and the resulting parallel implementation used has been based on MPI only.

Within this study we investigate the parallel performance of the MSA_BG algorithm. Optimization is achieved by applying hybrid MPI & OpenMP code development on the initial MPI implementation. The application was ported on the JUQUEEN supercomputer [9] and numerous experiments have been conducted. Profiling and benchmark tests were performed in order to evaluate the performance of the application.

This paper is structured as follows. Section II explains the multiple sequence alignment method MSA_BG. Section III presents the experimental framework. Profiling results of the MPI only implementation are presented in section IV. The design of a hybrid MPI/OpenMP implementation is explained in Section V. The experiments, performance evaluation and results analysis are discussed in Section VI. We present the conclusion and future work aspects in Section VII.

II. Multiple Sequence Alignment Algorithm MSA_BG

A new highly scalable and locality aware parallel algorithm MSA_BG for multiple alignment of biological sequences is presented in [7]. MSA_BG is a parallel iterative algorithm with a regular computational and communication system based on data parallelism and replica code, which is executed on all computing nodes. The parallel paradigm is Single Program Multiple Data (SPMD) and data decomposition. The granularity is hybrid - coarse granular computing for each node (multithreaded process) that runs multithreading (fine granular) of the cores within the computing node. In the case of hybrid granularity in order to effectively use the resources of supercomputers it is appropriate to use hybrid parallel implementations. The parallel algorithm is designed according to the methodology for the synthesis of parallel algorithms, which is based on the correlation of the parameters of the algorithmic and architectural spaces. The conceptual model of the MSA_BG method for parallel multiple alignment of biological sequences on the basis of the ABC algorithm is shown in Fig. 1.

The allocation of computing resources is as follows:

The entire system simulates the behavior of a colony of beehives, and the number of hives is equal to the number of computing nodes. Each computing node simulates the behavior of a hive. Within a hive q swarms are included, where q is the number of segments of the system. The OpenMP threads simulate the behavior of many bees in the swarm. The swarms within a hive work on common lists of best temporary solutions and elite solutions. Each hive has a mother bee, which gets the best quality decisions of all swarms in the hive. The number of mother bees is equal to the number of MPI processes. The queen bee of the colony (MPI process rank 0) finds the elite solution of all the hives in colony.

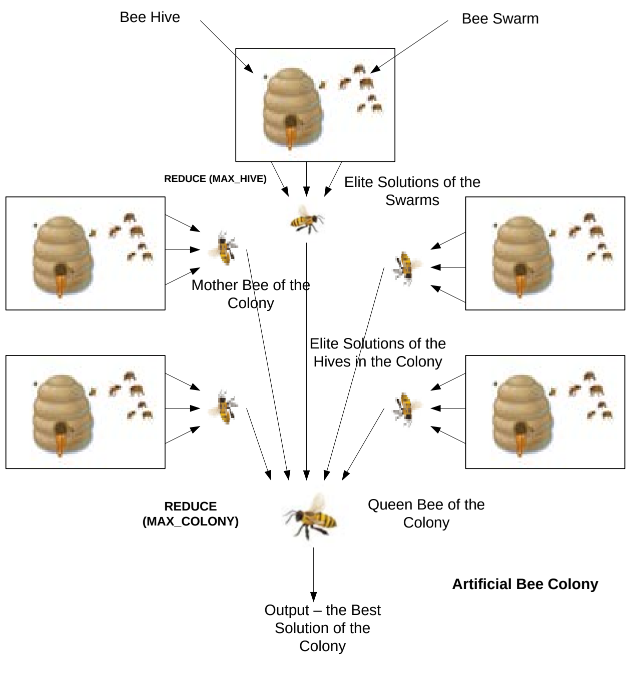


Fig. 1. Conceptual model of the MSA_BG method for parallel multiple alignment of biological sequences on the basis of ABC algorithm.

A description of the algorithmic framework for the parallel multiple sequence alignment in steps follows:

1. Parse input file - Read the sequences

Each MPI process reads the sequences from the input file and calculates the maximum sequence length. If the variety of sequence lengths is less than a limit, a number of variety gaps are added.

2. Alignment of the sequences

The sequences are aligned by adding gaps in random positions, so that their length is equal to the maximum sequence length. For each sequence (row) a dynamic data structure containing the indexes of the gaps in the sequences has been generated. Each MPI process iterates through the entire set of sequences. Therefore, the number of iterations is equal to the number of processes. This is a highly parallel task as there is no dependency between the sequences.

3. The sequence-favorite is created

Every MPI process iterates through each column on the array of the sequences in order to find the favorite nucleotides. The iterations are equal in number to the max sequence length. The calculations are independent for each genome. Therefore this is also a perfect task for parallel multithreaded execution.

4. The grades of the genomes are calculated

Each MPI process calculates the grade of every sequence in the working set. The grade calculation is independent for each sequence. The number of iterations is equal to the number of the sequences.

5. The sequences are sorted in descending order

The scores of the working sets are stored in the list of solutions, which is sorted, in descending order by the total alignment scores. Each MPI process sorts its own working set of sequences.

6. The quality of the solutions is improved

Minor changes are made in the working set and the quality of the modified alignment is evaluated: the new sequence is compared with the sequence-favorite and a grade is calculated as described in step 4. In case of quality improvement, the new solution is accepted and is stored in the list of "best temporary solutions". Otherwise, the new alignment is ignored. This is an iterative process that can be executed in parallel.

The number of iterations, which corresponds to the number of attempts to improve the quality of each sequence alignment, is split as a division over the number of MPI processes. The total number of iterations is given as an input to the program. In this study, the benchmark tests have been performed using 1 million and 10 million iterations.

7. The total matrix grade is calculated

After the process of working set improvement is completed, the matrix grade is calculated as the sum of each sequence grade and is stored along with the process rank. An MPI collective communication through reduction and operation MAX is performed in order to locate the best matrix grade, as well as the process that found the best solution.

8. The best solution is written to the output file

The root process (rank 0) communicates with all MPI processes and informs the process with the best quality solution working set to save the aligned sequences to a file.

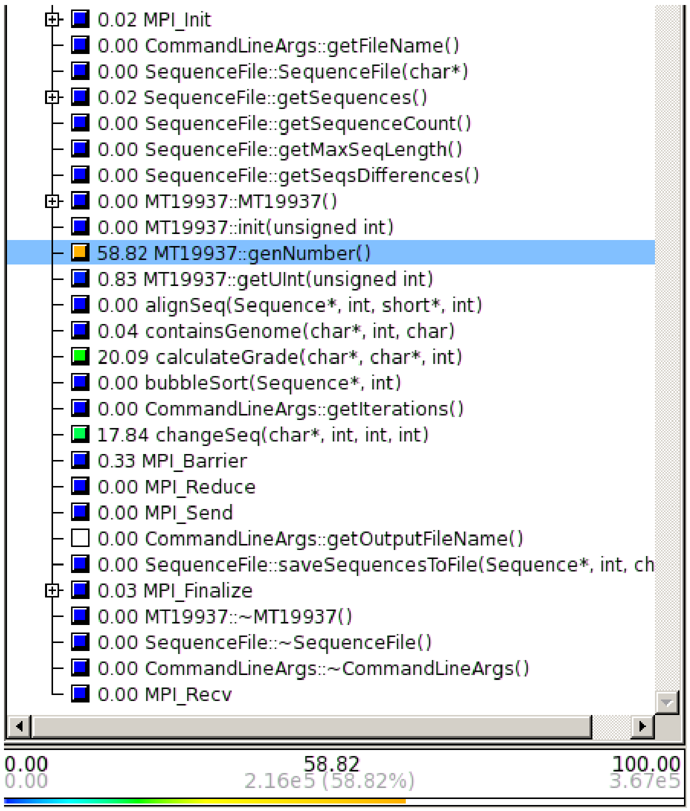
III. Experimental Framework

The benchmark tests were run on JUQUEEN, an IBM BlueGene/Q supercomputer. JUQUEEN consists of 28 racks on which 28,672 nodes are installed in total. The processor that is used is IBM PowerPC A2, 1.6 GHz. There are 16 cores per node supporting additional 4-way SMT each. Memory per node is 16 GB. Additionally JUQUEEN hosts 248 I/O nodes.

JUQUEEN provides interactive access and submission of batch jobs through two login nodes with processor architecture IBM Power 740. The front-end nodes have identical environments. The operating system is RedHat Linux (6.2) [10].

IV. Profiling Results of the MPI only Implementation

Scalasca profiling tool for measuring and analyzing runtime behavior has been used for the initial profiling of the code [11].



```
0.02 MPI_Init
0.00 CommandLineArgs::getFileName()
0.00 SequenceFile::SequenceFile(char*)
0.02 SequenceFile::getSequences()
0.00 SequenceFile::getSequenceCount()
0.00 SequenceFile::getMaxSeqLength()
0.00 SequenceFile::getSeqsDifferences()
0.00 MT19937::MT19937()
0.00 MT19937::init(unsigned int)
58.82 MT19937::genNumber()
0.83 MT19937::getUInt(unsigned int)
0.00 alignSeq(Sequence*, int, short*, int)
0.04 containsGenome(char*, int, char)
20.09 calculateGrade(char*, char*, int)
0.00 bubbleSort(Sequence*, int)
0.00 CommandLineArgs::getIterations()
17.84 changeSeq(char*, int, int, int)
0.33 MPI_Barrier
0.00 MPI_Reduce
0.00 MPI_Send
0.00 CommandLineArgs::getOutputFileName()
0.00 SequenceFile::saveSequencesToFile(Sequence*, int, ch
0.03 MPI_Finalize
0.00 MT19937::~MT19937()
0.00 SequenceFile::~SequenceFile()
0.00 CommandLineArgs::~CommandLineArgs()
0.00 MPI_Recv

0.00                 58.82                    100.00
0.00           2.16e5 (58.82%)                3.67e5
```

Fig. 2. Scalasca profiling results containing execution time percentages using 1024 MPI processes on JUQUEEN and 107 iterations.

The tests have shown that the routines that consume the highest amount of time are *genNumber*, *calculateGrade* and *changeSeq*. *GenNumber* implements the Mersenne Twister pseudo random generator, while *changeSeq* attempts to improve the solution by shifting the retrieved sequence. *CalculateGrade* is used to evaluate the new alignment each time a new modification of the alignment is proposed by *changeSeq*. The total amount of time consumed by these routines is approximately 95% of the overall execution time (Fig. 2). The rate of time consumption by these 3 routines is very similar using 512/1024 MPI processes and 106/107 iterations as a termination condition.

Important parts of the application, considering I/O, are steps 1 and 8, as described in section 2. At step 1 all MPI processes read the sequences from a common input file. The function *getSequences* is called by every process in order to parse the sequences from the input file and store them in their own memory space as the initial working set. At step 8 the best quality solution is written to the output file. The MPI process which holds the best solution, calls *saveSequencesToFile* to save the final solution.

As a result the total time spent at the *getSequences* calls is proportional to the number of MPI processes, while the time for the *saveSequencesToFile* call stays almost constant for the same problem size. However, in all cases the percentage of time spent for I/O is below 0.2% of the overall execution time.

V. Hybrid MPI/OpenMP Implementation

The concept of hybrid parallelism that was implemented on MSA_BG is based on simple logic. Each MPI process forks multiple OpenMP threads to work in parallel with the sequences in the working set. The parallel hybrid MPI/OpenMP computational model of MSA_BG algorithm for multiple sequence alignment is presented in Fig. 3.

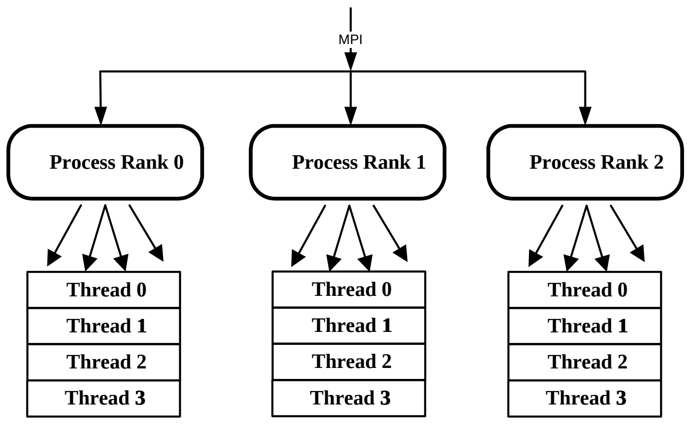


Fig. 3. Hybrid OpenMP/MPI parallel computational model of MSA_BG algorithm.

A Hybrid MPI/OpenMP version of the code has been implemented in order to exploit more efficiently the whole shared/distributed memory hierarchy of the JUQUEEN system.

The steps 2, 3, 4 and 6 of the algorithm described in section 4 are performed in iterations until every sequence is processed. The initial MPI implementation was modified by including

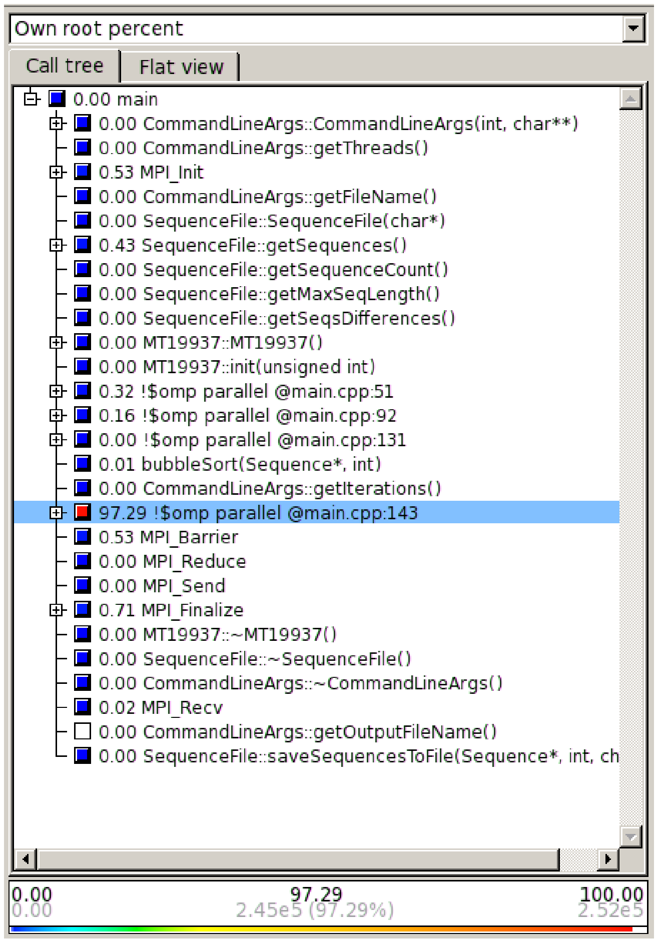
OpenMP directives in the source code of the MSA_BG application in steps 2, 3, 4 and 6.

Initial benchmark tests have shown that increasing the number of threads for the same number of MPI processes and nodes decreases the execution time significantly. Within the Table I we have measured the execution time of step 6 (the most time consuming part of the code) when adding 2, 4, 8 ad 16 OpenMP threads per MPI process. We also measure the relative speedup (normalized to using only 2 OpenMP threads per MPI process). These results correspond to using 512 processes allocating 4 MPI tasks per JUQUEEN node and for a total of 10^7 iterations.

TABLE I. RELATIVE SPEEDUP OF EXECUTION TIME OF STEP 6 USING OPENMP DIRECTIVES.

#OpenMP threads	**Time for step 6 (seconds)**	**Relative speedup**
2	241.33	1.00
4	122.65	1.96
8	34.04	7.08
16	27.57	8.75

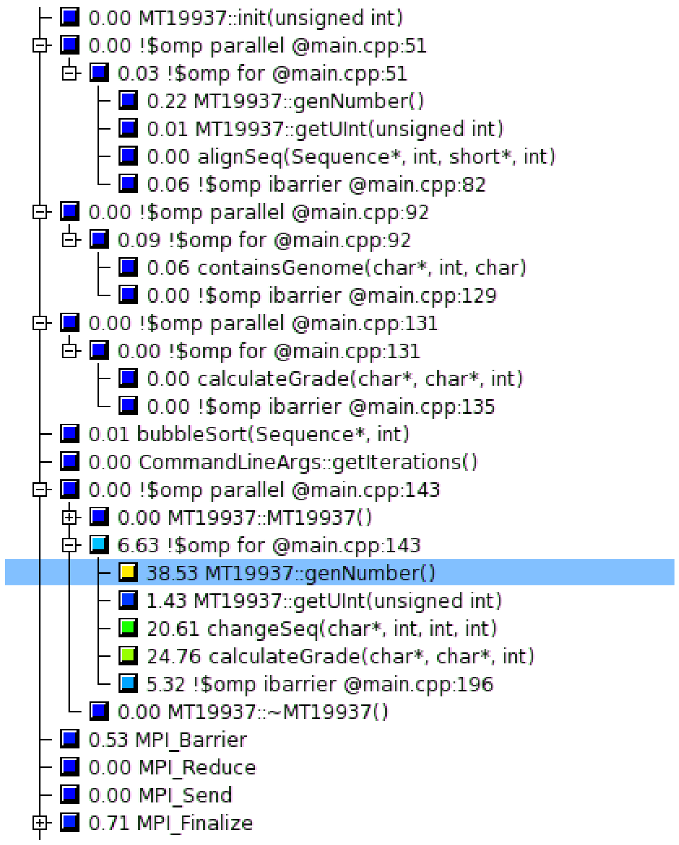
Scalasca profiling of the hybrid MPI/OpenMP implementation is shown in Fig. 4.



```
0.00 main
  0.00 CommandLineArgs::CommandLineArgs(int, char**)
  0.00 CommandLineArgs::getThreads()
  0.53 MPI_Init
  0.00 CommandLineArgs::getFileName()
  0.00 SequenceFile::SequenceFile(char*)
  0.43 SequenceFile::getSequences()
  0.00 SequenceFile::getSequenceCount()
  0.00 SequenceFile::getMaxSeqLength()
  0.00 SequenceFile::getSeqsDifferences()
  0.00 MT19937::MT19937()
  0.00 MT19937::init(unsigned int)
  0.32 !$omp parallel @main.cpp:51
  0.16 !$omp parallel @main.cpp:92
  0.00 !$omp parallel @main.cpp:131
  0.01 bubbleSort(Sequence*, int)
  0.00 CommandLineArgs::getIterations()
  97.29 !$omp parallel @main.cpp:143
  0.53 MPI_Barrier
  0.00 MPI_Reduce
  0.00 MPI_Send
  0.71 MPI_Finalize
  0.00 MT19937::~MT19937()
  0.00 SequenceFile::~SequenceFile()
  0.00 CommandLineArgs::~CommandLineArgs()
  0.02 MPI_Recv
  0.00 CommandLineArgs::getOutputFileName()
  0.00 SequenceFile::saveSequencesToFile(Sequence*, int, ch

0.00            97.29              100.00
0.00       2.45e5 (97.29%)         2.52e5
```

(a) Call tree - cpu time percentages.



```
0.00 MT19937::init(unsigned int)
0.00 !$omp parallel @main.cpp:51
  0.03 !$omp for @main.cpp:51
    0.22 MT19937::genNumber()
    0.01 MT19937::getUInt(unsigned int)
    0.00 alignSeq(Sequence*, int, short*, int)
    0.06 !$omp ibarrier @main.cpp:82
0.00 !$omp parallel @main.cpp:92
  0.09 !$omp for @main.cpp:92
    0.06 containsGenome(char*, int, char)
    0.00 !$omp ibarrier @main.cpp:129
0.00 !$omp parallel @main.cpp:131
  0.00 !$omp for @main.cpp:131
    0.00 calculateGrade(char*, char*, int)
    0.00 !$omp ibarrier @main.cpp:135
0.01 bubbleSort(Sequence*, int)
0.00 CommandLineArgs::getIterations()
0.00 !$omp parallel @main.cpp:143
  0.00 MT19937::MT19937()
  6.63 !$omp for @main.cpp:143
    38.53 MT19937::genNumber()
    1.43 MT19937::getUInt(unsigned int)
    20.61 changeSeq(char*, int, int, int)
    24.76 calculateGrade(char*, char*, int)
    5.32 !$omp ibarrier @main.cpp:196
  0.00 MT19937::~MT19937()
0.53 MPI_Barrier
0.00 MPI_Reduce
0.00 MPI_Send
0.71 MPI_Finalize
```

(b) OpenMP parallel regions in detail

Fig. 4. Scalasca screenshots corresponding to 1024 MPI processes with 16 OpenMP threads per process. The problem size corresponds to 107 iterations.

The 6th step of the algorithm that improves the sequences alignments consumes almost the most significant part of the total execution time. In Figure 7(b) the four OpenMP parallel regions are displayed in detail. The routines genNumber, changeSeq and calculateGrade still consume a large proportion of the overall execution time. However, in this case they are executed in parallel by a number of OpenMP threads. This results in a significant decrease of the total execution time.

VI. PARALLEL PERFORMANCE EVALUATION OF HYBRID IMPLEMENTATION AND RESULTS ANALYSIS

Similarity searching between RNA segments of various influenza viruses strains obtained from Genbank [12] has been carried out based on the hybrid MPI/OpenMP version of MSA_BG algorithm. Some experiments using various numbers of cores have been conducted. The experimental results in Table II, Fig. 5 and Fig. 6 show that the parallel program implementation for multiple sequence alignment scales well as the number of the cores increases.

TABLE II. EXECUTION TIME AND ACCELERATION OF MSA_BG ALGORITHM USING VARIOUS NUMBERS OF CORES AND VARIOUS SEQUENCES

Length of seq.	**Number of seq.**	**Input file (MB)**	**Execution time, minutes**			**Acceleration, %**	
			128 cores	***256 cores***	***512 cores***	***256 cores***	***512 cores***
2340	3780	8,78	11,33	6,51	3,85	42,54	66,02
1750	5992	10,52	14,58	8,45	4,67	42,04	67,97
1460	6128	8,90	13,35	7,82	4,12	41,42	69,14
890	3996	3,58	7,12	4,14	2,16	41,85	69,66
2340	5850	13,61	17,2	10,06	5,64	41,51	67,21
1750	8999	15,77	22,01	12,47	7,23	43,34	67,15
1460	9662	14,05	20,55	12,04	6,45	41,41	68,61
890	6150	5,50	10,26	6,1	3,42	40,55	66,67

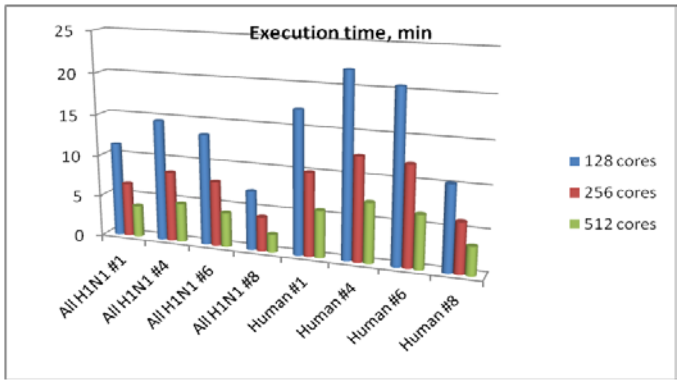


Fig. 5. Execution time of MSA_BG algorithm as a function of number of cores.

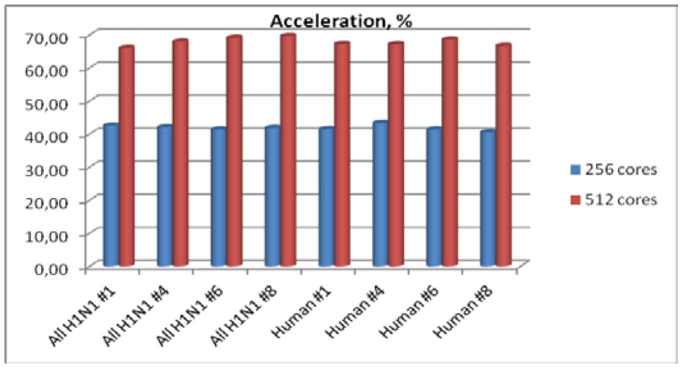


Fig. 6. Acceleration of MSA_BG algorithm with respect to 128 cores.

VII. CONCLUSION

The parallel software MSA_BG for multiple sequence alignment has been ported and tuned on the Blue Gene/Q supercomputer JUQUEEN. Hybrid MPI/OpenMP parallelization was implemented and evaluated experimentally. Parallel performance was investigated and optimized through benchmark tests and profiling.

The implementation of hybrid MPI/OpenMP parallelization on MSA_BG reduces up to a good factor the overall runtime of the MPI only version of the application as it allows us to fully occupy the CPU capacity of a JUQUEEN node with threads. It should also be noted that runs using the hybrid implementation resulted in better quality of sequence alignments due to additional randomness that was produced by the Mersenne Twister generator.

The performance estimation and analyses show that the hybrid parallel program implementation of MSA_BG algorithm scales well as the number of the cores increases and is well balanced both in respect to the workload and machine size. The optimized code is universal and can be applied for other similar research projects and experiments in the field of bioinformatics. MSA_BG software allows researchers to conduct their experiments and perform simulations with very large amounts of data.

ACKNOWLEDGMENT

This work was supported by the National Science Fund, Bulgarian Ministry of Education and Science, Grant DN 07/24. Thanks the project PRACE for the access to the JUQUEEN system.

REFERENCES

[1] European Big Data Value Partnership, Strategic Research and Innovation Agenda, January 2016.

[2] Stephens ZD, Lee SY, Faghri F, Campbell RH, Zhai C, Efron MJ, Iyer R, Schatz MC, Sinha S, Robinson GE.Big Data: Astronomical or Genomical?, PLoS Biol. 2015 Jul 7;13(7):e1002195. doi: 10.1371/journal.pbio.1002195, eCollection 2015.

[3] H. Carrillo and D. Lipman, "The multiple sequence alignment problem in biology," SIAM Journal of Applied Mathematics, vol. 48, no. 5, 1988, pp. 1073-1082.

[4] L. Wang and T. Jiang, "On the complexity of multiple sequence alignment," Journal of Computational Biology, vol. 1, no. 4, 1994, pp. 337–348, doi:10.1089/cmb.1994.1.337.

[5] W. Just, "Computational complexity of multiple sequence alignment with SP-score," Journal of Computational Biology, vol. 8, no. 6, 2001, pp. 615–23.

[6] S. Sze, Y. Lu, and Q. Yang, "A polynomial time solvable formulation of multiple sequence alignment," Journal of Computational Biology, vol. 13, no. 2, 2006, pp. 309–319, doi:10.1089/cmb.2006.13.309.

[7] P. Borovska, V. Gancheva, N. Landzhev, "Massively parallel algorithm for multiple biological sequences alignment", 36th International Conference on Telecommunications and Signal Processing (TSP), 2-4 July, 2013, Rome, Italy, pp. 638 - 642.

[8] D. Karaboga, "An idea based on honey bee swarm for numerical optimization," Technical Report-TR06, Erciyes University, Engineering Faculty, Computer Engineering Department, 2005, http://mf.erciyes.edu.tr/abc/pub/tr06_2005.pdf

[9] JUQUEEN – Configuration, http://www.fz-juelich.de/ias/jsc/EN/Expertise/Supercomputers/JUQUEEN/Configuration/Configuration_node.html

[10] JUQUEEN – Logging on to JUQUEEN, http://www.fz-juelich.de/ias/jsc/EN/Expertise/Supercomputers/JUQUEEN/UserInfo/LogonJuqueen.html.

[11] Scalasca, http://www.scalasca.org/

[12] GenBank, http://www.ncbi.nlm.nih.gov/Genbank/