Available online at www.prace-ri.eu

Partnership for Advanced Computing in Europe

Massively Parallel Algorithm for Multiple Sequence Alignment Based on Artificial Bee Colony

Plamenka Borovska, Veska Gancheva

National Centre for Supercomputing Applications, Bulgaria

Abstract: ***In silico*** **biological sequence processing is a key task in molecular biology. This scientific area requires powerful computing resources for exploring large sets of biological data. Parallel** ***in silico*** **simulations based on methods and algorithms for analysis of biological data using high-performance distributed computing is essential for accelerating the research and reducing the investment. Multiple sequence alignment is a widely used method for biological sequence processing. The goal of this method is DNA and protein sequences alignment. This paper presents an innovative parallel algorithm MSA_BG for multiple alignment of biological sequences that is highly scalable and locality aware. The MSA_BG algorithm we describe is iterative and is based on the concept of Artificial Bee Colony metaheuristics and the concept of algorithmic and architectural spaces correlation. The metaphor of the ABC metaheuristics has been constructed and the functionalities of the agents have been defined. The conceptual parallel model of computation has been designed and the algorithmic framework of the designed parallel algorithm constructed. Experimental simulations on the basis of parallel implementation of MSA_BG algorithm for multiple sequences alignment on heterogeneouc compact computer cluster and supercomputer BlueGene/P have been carried out for the case study of the influenza virus variability investigation. The performance estimation and profiling analyses have shown that the parallel system is well balanced both in respect to the workload and machine size.**

Keywords**: Artificial Bee Colony, Bioinformatics, BlueGene/P, Computer Cluster, High Performance Computing, Multiple Sequence Alignment, Performance.**

1. Introduction

Multiple sequence alignment involves more than two biological sequences, generally protein, DNA, or RNA [1]. Multiple sequence alignment is computationally difficult and is classified as a NP-Hard problem [2] [3] [4]. ClustalW is a widely used multiple sequence alignment algorithm for DNA or proteins and implements a progressive method for multiple sequence alignment [5]. It calculates the best match for the selected sequences and lines them up so that the identities, similarities and differences can be seen. The basic algorithm behind ClustalW proceeds in three stages: pairwise alignment (PA), guide tree (GT) and multiple alignment (MA). Each of the phases produces intermediate data which is used as an input for the next one, but the calculations themselves are independent.

Several parallel ClustalW algorithms for multiple sequence alignment have been reported in recent years. ClustalW-MPI is a distributed and parallel implementation on distributed computer clusters and on traditional parallel computers and uses a scheduling strategy called fixed-size chunking where batches of tasks of one fixed size are allocated to available processors [6]. The ClustalW_MPI algorithm is not so effective in the case of highly parallel implementations. Analysis of the algorithm shows that it is characterized by significant computational imbalance, because it is based on the "master-slave" parallel algorithmic paradigm [7]. Increasing the input file size is limited by memory considerations. Thus, on JUQUEEN, where the memory available per core is 1Gb [8], the maximum number of input sequences is approximately 10000 (depending on the sequences length).

The aim of our work is to propose an innovative parallel algorithm MSA_BG for multiple sequence alignment based on the concept of Artificial Bee Colony (ABC) metaheuristics and the concept of algorithmic and architectural spaces correlation.

The ABC algorithm is an optimization algorithm based on the intelligent foraging behaviour of a honey bee swarm [9]. In the ABC model, the colony consists of three groups of bees: employed bees, onlookers and scouts.

The choice of the ABC algorithm is based on the fact that, in essence, it is a hybrid metaheuristics - a combination of methods based on populations (scouts generate a number of possible solutions simultaneously) and a method based on trajectories (employed bees perform the local searches around the decisions of the scouts, seeking to improve the decisions quality).

2. Multiple Sequence Alignment Algorithm MSA_BG based on Artificial Bee Colony Metaheuristics

The ABC algorithm is based on populations. The position of a food source represents a possible solution of the optimization problem and the amount of nectar represents the quality of the proposed solution. The number of scouts is equal to the number of solutions in the population. The first step is to generate randomly a partitioned initial population. After the initialization, the population repeats the cycle of seeking employed bees, onlookers and scouts. The scout modifies the positions of the sources in his memory and remembers the new position of a food source.

In case of the new source the nectar amount is greater than the previous, the scout remembers the position of the new source and forgets the old one (Figure 1.). Otherwise the scout keeps the position of current source in the memory. Once all the scouts complete the search process, they share information about the positions of food sources with onlookers through dance (Figure 2.). Each onlooker evaluates the information for the nectar according to the dance of scouts and then selects a food source according to the amount of food in the source. The onlooker compares quantities of nectar in the new source with that in already stored. If the amount of the nectar is greater in the new source, the bee remembers the new position and forgets the old one. The scouts determine which sources should be abandoned and randomly select new sources. The employed bees search around for decisions (Figure 3.).

On each iteration of the cycle a decision is generated by the function lookForNectarHere(). The decision is evaluated by the function evaluatePosition(). Then it is determined whether it is good enough, based on the scout experience. If the source is good enough, the scout invites employed bees and begins dancing – danceToCallEmployed Bees ().

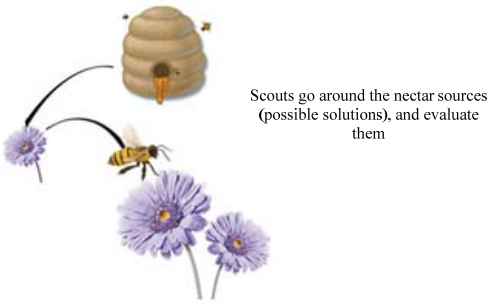

Figure 1. Scouts go around the search space.

Figure 2. The scout invites employed bees and begins dancing.

Figure 3. Employed bees search around for decisions.

The concept is to synthesize a parallel iterative algorithm with a regular computational and communication system based on data parallelisms 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 granule computing for each node (multithreaded process) that runs multithreading (fine granule) of the cores within the computing node. In the case of hybrid granularity, in order to use effectively the resources of supercomputers hybrid parallel implementations are employed.

A new parallel algorithm called MSA_BG, in which each process simulates the behaviour of a beehive and each hive contains multiple swarms is proposed. The design of the algorithm is based on the methodology for the synthesis of parallel algorithms, which is based on the correlation of the parameters of the algorithmic and architectural spaces. Here threads simulate the behaviour of many bees in the swarm.

Scout bees in the swarms surround certain subregions in the searching space and construct a potential solution. Once the scout bees obtain a possible (feasible) solution, they return to the hive and begin to dance in the hive. Onlookers watch the dance of employed bees, choose one of their solutions and evaluate them. The employed bees select one of the solutions and make attempts to improve it based on a local search. The quality of the solutions obtained is determined by the grade of sequences similarity. The greater this evaluation is, the higher the quality of the alignment, i.e., the criterion of optimality is a maximum similarity score.

If a scout generates a higher quality solution then the possibility of including it in the list of elite solutions will be increased. In the next step this solution is improved using a local search by the employed bees.

a. Metaphor

The food source is presented by possible sequence alignments. The quality of solutions (nectar amount) is the evaluation function of the similarity between the sequences for a particular alignment.

The proposed algorithm is based on the concept of swarms and hives. The allocation of computing supercomputer resources is as follows: the entire supersystem simulates the behaviour of a colony of beehives, and the number of hives is equal to the number of computing nodes. Each computing node simulates the behaviour of a hive. In a hive there are q swarms, where q is the segments number of the supersystem. The swarms within a hive work on common lists of best temporary solutions and elite solutions. Each hive has a queen bee, which gets the elite decision of all swarms on the hive. Scout 0 reads the sequences from the memory and stores it in sharing memory of the computational node (hive). Scouts generate initial solutions via sequence alignment including gaps.. Scouts then evaluate the quality of the alignment using the following method.

First, an assessment by columns is done – in the case of nucleotide sequences the numbers of symbols – A, G, C and T – are counted. The numbers of symbols are compared and the nucleotides, which occur mostly in the different columns, are selected. This is called the "nucleotide-favourite" sequence (f_{ij}). After the calculation of assessments in columns the so-called sequence-favourite contained in each position the respective favourite in the column is formed, i.e., the nucleotides form the so-called sequence-favourite. Compared sequences along with the sequence-favourite form the so-called working set sequences.

A scoring matrix is built up where for sequences (rows in the matrix) the values of the evaluation function S are stored (by columns). For sequence (row) i in position j (column): nucleotide a_{ij} and nucleotide–favourite f_{ij}:

$S_{ij} = 0$ in case of a_{ij} = gap

$S_{ij} = 1$ in case of $a_{ij} = f_{ij}$

$S_{ij} = -1$ in case of $a_{ij} \neq f_{ij}$

The elements of the scoring matrix are calculated for each sequence $S_{p,n+1}$, where n is the sequence length and p the number of aligned sequences.

The overall assessment of the quality of alignment S is calculated as:

$$S = \sum_{i=1}^{p} \sum_{j=1}^{n} S_{i,j} \quad (1)$$

The additional column (similarity counters) of the scoring matrix $S_{n,n+1}$ consists of the similarity evaluation for each sequence compared with the sequence-favourite, i.e.,

$$S_{i,n+1} = \sum_{j=1}^{n} S_{i,j} \quad (2)$$

The next example shows the working set. The sequence-favourite is marked in red.

A	**G**	**T**	**C**	**A**	**A**	**T**
A	A	T	C	G	A	T
A	G	T	C	A	T	T
A	G	-	G	A	A	G

The next example illustrates the calculation of scoring matrix S. The scoring column of the counters is marked in red and consists of similarity scores for each sequence and the sequence-favourite.

1	-1	1	1	-1	1	1	**3**
1	1	1	1	1	-1	1	**6**
1	1	0	-1	1	1	-1	**2**

The scouts write down the scores of the working sets in the list of working solutions that is sorted in descending order by the total alignment scores. For each sequence (row) dynamic data structure containing the indexes of the gaps in the sequence is generated.

The onlookers select the best quality working set and perform a local search – they make minor changes in the working set and evaluate the quality of the modified alignment. In case of quality improvement, the modification is accepted and the working set is stored in the list of "best temporary solutions". Otherwise, the new alignment is ignored.

The following approach for modification of the aligned working set of sequences is proposed. The column with counters of the scoring matrix S is reviewed and the row (sequence) with the lowest counter value (sequence that differs most from the favourite) selected. Using a random generator an index for inserting a gap (INS) and an index for deleting a gap (DEL) are selected from the list of empty positions (to keep unchanged the length of the sequence) ensuring that INS ≠ DEL. Both indexes are compared. If DEL>INS, then all characters

in positions between INS and DEL are shifted one position to the right (shift_right), otherwise if DEL>INS they are shifted one position to the left (shift_right).

An example for the sequence is shown below:

```
A-TGC-GGTA-CCGT-G
```

The list of gaps indexes is: 1, 5, 10, 15.

Based on a random generator, the position for insertion INS=4 and the position for deletion DEL=15 are selected. In this case, DEL>INS, the operation is shift_right.

```
A-TGC-GGTA-CCGT-G
A-TG-C-GGTA-CCGTG
```

The value of similarity counter for the sequence compared with the sequence-favourite is calculated when the modification of the working set is accepted and is stored in the list of temporary best solutions; otherwise, the modification is ignored.

In case of inserting position INS=12, and deleting position DEL=5, DEL<INS and the operation is shift_left.

```
A-TGC-GGTA-CCGT-G
A-TGCGGTA-C-CGT-G
```

After a number of modifications, employed bees suspend the processing of the current working set and write down the best solution in the list of elite solutions that is sorted in descending order.

The condition for termination of the parallel algorithm is the number of iterations; then, the mother bee shall inform the queen bee of the colony and sent her the quality of the best solution (elite solution).

b. Algorithmic framework for parallel multiple alignment

The algorithmic framework for parallel multiple alignment of biological sequences MSA_BG on the basis of ABC algorithm is shown in Figure 4.

```
Parallel For
For m=1, Q          // For every hive
   For r=1, q       // For every swarm (node)
      Input_sequences //Scout 0
      Parallel Sections
         Parallel For k=0,15    /*Scout_bees
            Generate_ Random_Alignment
            Evaluate_Score_By Columns
            Construct_Favorite_Sequence
            Evaluate_Column of Counters
            Save_Working Set_in_Optimization list
         End Parallel For
         Parallel For k=0,15    /*Onlookers_bees
            Select_Random_Working_Set_Out_of_Optimization_List
            Select_Sequence_of_MIN_Counter
            Select_INS(Random)
            Select_DEL, such that DEL≠INS (RANDOM)
            If DEL>INS then shift_right else shift_left
            Evaluate_Column of Counters
            If higher_quality then Save_In_Optimization_List
            else skip
         End Parallel For
      End Parallel For
   Sort_Optimization_List_by_Quality   // Scout 0
   Reduce_Elite_Alignment_to_Mother_of_Hive  // Scout 0
   Reduce_MAX_ Elite_Alignment_of_Hive  //Mother of Hive
   End Parallel For
   Reduce_MAX_ Elite_Alignment _of_Colony //Queen of Colony
   Output_Best_Alignment_Obtained_So_Far        //Queen of Colony
```

Figure 4. Algorithmic framework for parallel multiple alignment of biological sequences MSA_BG on the basic of ABC algorithm.

The conceptual model of the MSA_BG method for parallel multiple alignment of biological sequences on the basis of ABC algorithm is shown in Figure 5.

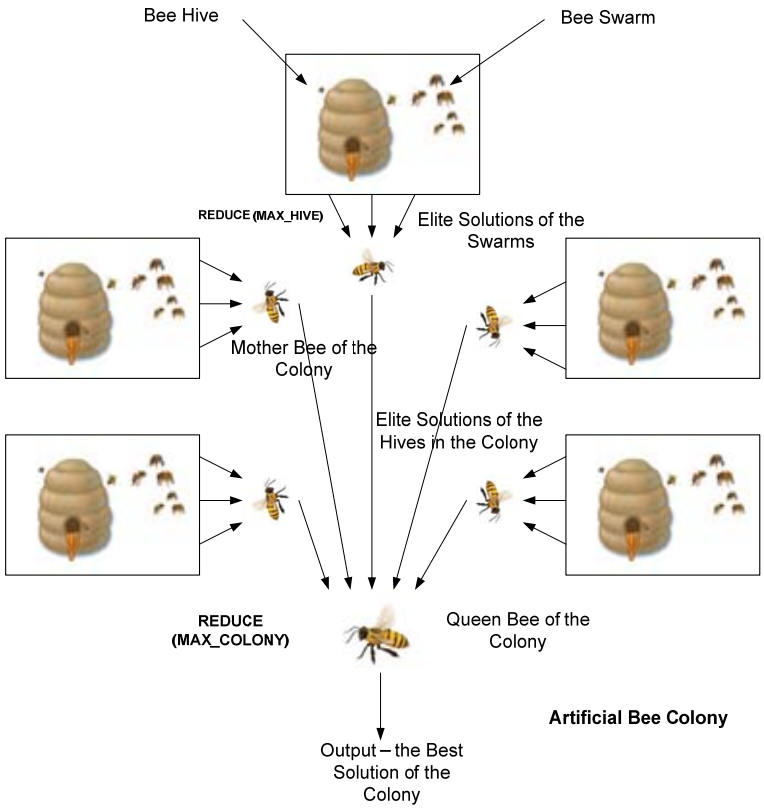

Figure 5. Conceptual model of the MSA_BG method for parallel multiple alignment of biological sequences on the basis of ABC algorithm.

The colony queen through collective communication reduction and the MAX operation gets the quality of the elite decisions by the mother bees of the hives. Thereafter the colony queen determines and sends the best one.

3. Experimental Frameworks

a. Supercomputer IBM Blue Gene/P

The experimental framework includes an 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. The smallest partition size available currently, is 128 compute nodes (512 processor cores). The maximum LINPACK performance achieved is Rmax= 23.42 Tflops while the theoretical peak performance is Rpeak= 27.85 Tflops. Furthermore, the cabinets with computing nodes of the supercomputing system include the following major components:

1. Front-End Node: 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 servers 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.

b. Heterogeneous Computer Cluster

The experimental framework is based on a heterogeneous computer cluster of 10 nodes including 8 servers AMD Opteron 64 Dual Core Processor 1.8 GHz, RAM: 2GB 800 MHz, HDD: 2x160GB Hitachi SATA in RAID 0 and 2 servers CPU: 2x Intel Xeon E5405 Quad Core Processor 2 GHz, RAM: 4GB 800MHz, HDD: 2x 146 GB Hitachi 10000 RPM in RAID 0. All nodes are interconnected via gigabit Ethernet switch. The operating system is 64 bit Scientific Linux 5.3. Message passing is based on the MPICH2 1.1.1p2 distribution of the MPI standard. The resource manager is SLURM.

4. Parallel Performance Evaluation and Results Analysis

a. Experimental Simulations on Supercomputer IBM Blue Gene/P

The objective of the experiments is to estimate experimentally parallel performance parameters and make profiling of the program implementation on the basis of MSA_BG algorithm. Similarity searching between RNA segments of various influenza viruses A/H1N1strains obtained from Genbank [10] has been carried out based on the parallel program MPI implementation of MSA_BG algorithm for multiple sequence alignment on supercomputer BlueGene/P.

Some experiments using various numbers of cores and 10 000 000 iterations on the BlueGene/P supercomputer have been conducted. The experimental results for the execution time are shown in TABLE I. Figure 6.

The speedup is evaluated as a ratio of the execution time on 32 cores to the execution time on 64, 256, and 512 cores respectively. The experimental results for the speedup of MSA_BG algorithm on the BlueGene/P using various numbers of cores and 10 000 000 iterations are shown in TABLE II. and Figure 7.

The experimental results show that the parallel program implementation for multiple sequence alignment scales well as the number of the cores increases.

TABLE I. EXECUTION TIME OF MSA_BG ALGORITHM ON SUPERCOMPUTER BLUEGENE/P USING VARIOUS NUMBER OF CORES AND 10 000 000 ITERATIONS

Cores	Execution time
32	19.38 min
64	10.22 min
128	5.30 min
256	3.15 min
512	2.16 min

TABLE II. SPEEDUP OF MSA_BG ALGORITHM ON SUPERCOMPUTER BLUEGENE/P USING VARIOUS NUMBER OF CORES AND 10 000 000 ITERATIONS

Cores	Speedup
64	1.90
128	3.66
256	6.15
512	8.97

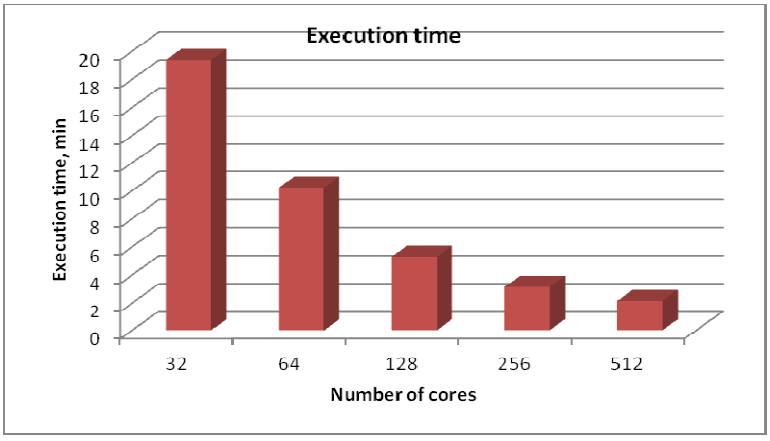

Figure 6. Execution time of MSA_BG algorithm on BlueGene/P as a function of number of cores for 10 000 000 iterations.

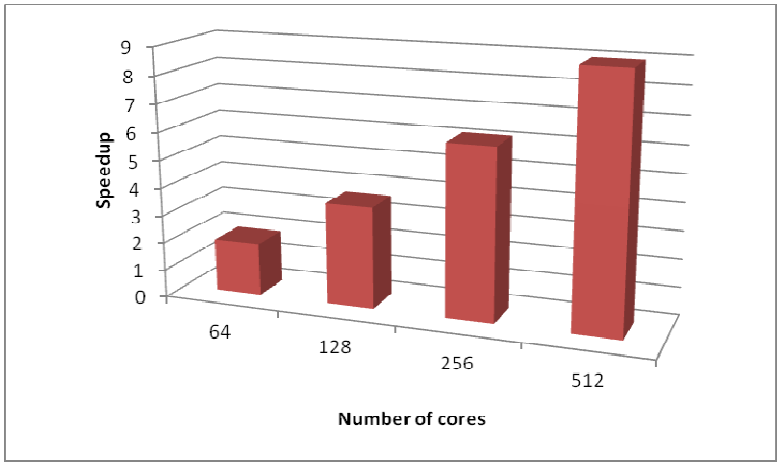

Figure 7. Speedup of MSA_BG algorithm on BlueGene/P using various number of cores for 10 000 000 iterations.

In order profile the parallel program implementation we have used the software tool SCALASCA [11] that supports performance optimization of parallel programs by measuring and analyzing their runtime behaviour. The analysis identifies potential performance bottlenecks – in particular those concerning communication and synchronization – and offers guidance in exploring their causes. Processes loading profiles during the execution of the algorithm MSA_BG for multiple sequences alignment on BlueGene/P is shown in Figure 8. The processes communication profile informs us that only the process rank 0 sends messages (function MPI_send()) (Figure 9.), while all the other ranks receive messages (function MPI_Recv()) (Figure 10.).

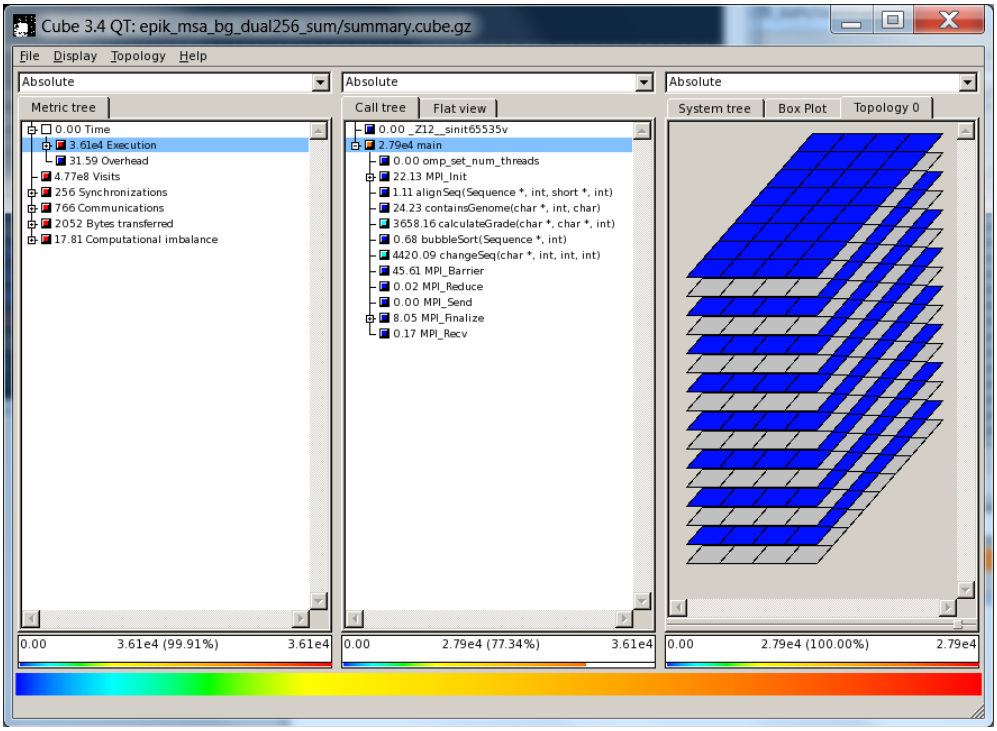

Figure 8. Processes loading profiles during the execution of the MSA_BG algorithm for multiple sequences alignment on BlueGene/P.

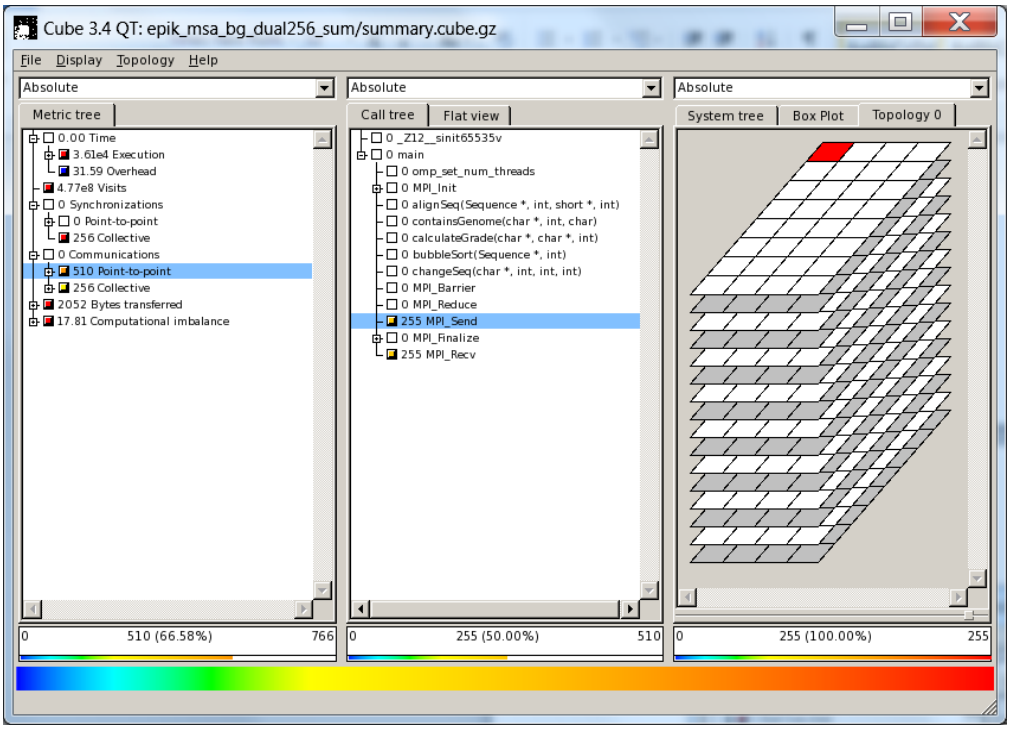

Figure 9. Processes communication profile – send, during the execution of the algorithm MSA_BG for multiple sequences alignment on BlueGene/P.

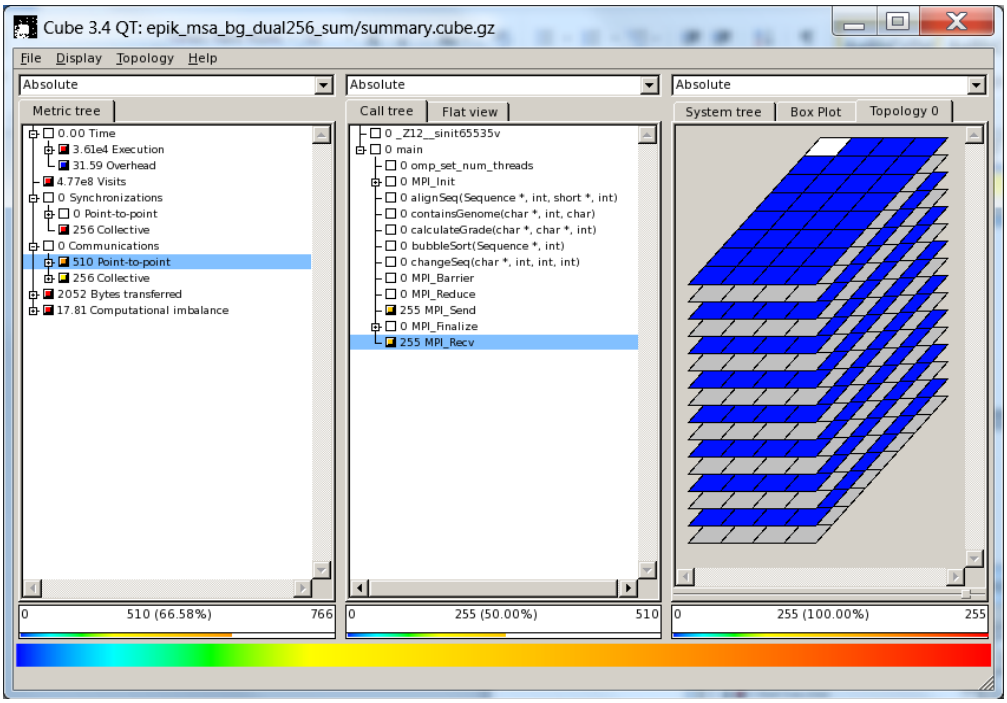

Figure 10. Processes communication profile – receive, during the execution of the algorithm MSA_BG for multiple sequences alignment on BlueGene/P.

b. Experimental Simulations on the Heterogeneous Computer Cluster

The objective of the experiments was to estimate parallel performance parameters of the program implementation on the basis of the MSA_BG algorithm. Similarity searching between RNA segments of various influenza viruses (A/H1N1strains) obtained from Genbank [10] has been carried out based on the MPI-based implementation of the MSA_BG algorithm for multiple sequence alignment on the heterogeneous computer cluster.

The experiments were conducted using various numbers of cores and various numbers of iterations. Experimental results are shown in TABLE III.

TABLE III. EXECUTION TIME OF MSA_BG ALGORITHM USING VARIOUS NUMBERS OF CORES AND VARIOUS NUMBERS OF ITERATIONS

Execution time					
Cores	***Iterations***				
	1000	***10000***	***100000***	***1000000***	***10000000***
1	4 sec	41 sec	6.42 min	62 min	624 min
2	2 sec	20 sec	3.22 min	31 min	312 min
8	1 sec	6 sec	0.51 min	8 min	82 min
16	1 sec	3 sec	0.26 min	4 min	42 min
28	1 sec	2 sec	0.15 min	2 min	24 min

The speedup is evaluated as a ratio of the sequential execution time to the parallel execution time. The experimental results for the speedup of MSA_BG algorithm on using various numbers of cores and various numbers of iterations are shown in TABLE IV. and Figure 11.

TABLE IV. SPEEDUP OF MSA_BG ALGORITHM USING VARIOUS NUMBERS OF CORES AND VARIOUS NUMBERS OF ITERATIONS

Speedup					
Cores	***Iterations***				
	1000	***10000***	***100000***	***1000000***	***10000000***
2	2	2,05	1,99	2,00	2,00
8	4	6,83	12,59	7,75	7,61
16	4	13,67	24,69	15,50	14,86
28	4	20,50	42,80	31,00	26,00

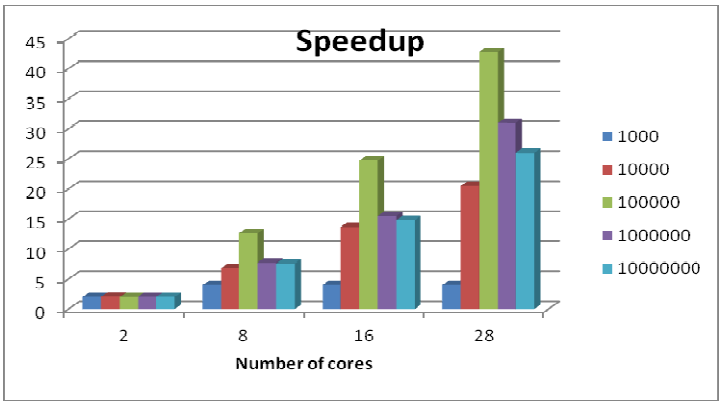

Figure 11. Speedup of MSA_BG algorithm as a function of the number of cores and iterations.

The efficiency is evaluated as a ratio of the achieved speedup to the number of cores. The results for the efficiency of the MSA_BG algorithm gains in the case of varying number of cores and number of iterations are shown in TABLE V. and Figure 12.

TABLE V. EFFICIENCY OF MSA_BG ALGORITHM USING VARIOUS NUMBERS OF CORES AND VARIOUS NUMBERS OF ITERATIONS

Efficiency					
Cores	***Iterations***				
	1000	***10000***	***100000***	***1000000***	***10000000***
2	1,00	1,03	1,00	1,00	1,00
8	0,50	0,85	1,57	0,97	0,95
16	0,25	0,85	1,54	0,97	0,93
28	0,14	0,73	1,53	1,11	0,93

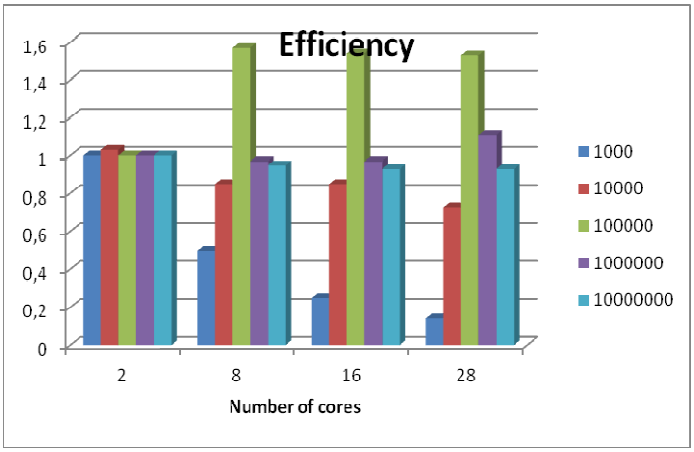

Figure 12. Parallel efficiency of MSA_BG algorithm using various numbers of cores and various numbers of iterations.

The parallel system shows good scalability with respect to both the cluster size and the workload size. In the case of small number of iterations the speedup obtained is poor. Increasing the number of iterations results in an improved speedup because of the better utilization of the cores.

5. Conclusion and Future Works

An innovative parallel algorithm MSA_BG for multiple alignment of biological sequences that is highly scalable and locality aware has been designed. The MSA_BG algorithm is iterative and is based on the concept of Artificial Bee Colony metaheuristics and the concept of algorithmic and architectural spaces correlation. The metaphor of the ABC metaheuristics has been constructed and the functionalities of the agents have been defined. The conceptual parallel computationional model has been designed and the algorithmic framework of the parallel algorithm has been constructed.

MSA_BG algorithm has a hierarchical structure, which enables the principle of locality (independent calculations) to be observed, and allows a very high scalability (hives and swarms), so high efficiency implementations for petaflop supercomputers could be expected.

To demonstrate the efficiency of the algorithm we have examined a case study investigating viral nucleotide sequences and finding out consensus motifs and variable domains in the different segments of the influenza virus. In the case study we have evaluated the parallel performance by profiling the multiple sequence alignment on a BG/P supercomputer and a computer cluster.

Parallel performance parameters such execution time, accelerating time and profiling have been measured and reported. The performance estimation and profiling analyses have shown that the parallel system is well balanced both in respect to the workload and machine size.

Acknowledgements

This work was financially supported by the PRACE-2IP project funded in part by the EUs 7th Framework Programme (FP7/2007-2013) under grant agreement no. RI-283493.

References

[1] 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.

[2] 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.

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

[4] 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.

[5] J. Thompson, D. Higgins, and T. Gibson, "ClustalW: Improving the Sensitivity of Progressive Multiple Sequence Alignment through Sequence Weighting, Position-Specific Gap Penalties and Weight Matrix Choice," Nucleic Acids Research, vol. 22, no. 22, 1994, pp. 4673-4680.

[6] K. Li, "ClustalW-MPI: ClustalW Analysis Using Distributed and Parallel Computing," Journal of Bioinformatics, vol.19, no. 12, 2003, pp. 1585–1586.

[7] P. Borovska, V. Gancheva, and S. H. Ko, "Scaling of Parallel Multiple Sequence Alignment on the Supercomputer Juqueen," the 7th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, 12-14 September 2013, Berlin, Germany.

[8] JUQUEEN – Configuration, http://www.fz-juelich.de/ias/jsc/ EN/Expertise/Supercomputers/JUQUEEN/Configuration/ Configuration_node.html, retrieved: 30.06.2013.

[9] D. Karaboga, "An Idea Based On Honey Bee Swarm for Numerical Optimization," Technical Report-TR06, Erciyes University, Engineering Faculty, Computer Engineering Department, 2005, mf.erciyes.edu.tr/abc/pub/tr06_2005.pdf, retrieved: 20.05.2013.

[10] GenBank, http://www.ncbi.nlm.nih.gov/Genbank/, retrieved: 20.06.2013.

[11] M. Geimer, F. Wolf, B. Wylie, B. Mohr, "A scalable tool architecture for diagnosing wait states in massively-parallel applications," Parallel Computing, vol. 35, no. 7, pp. 375–388, 2009. http://www.scalasca.org/