Parallelization and Optimization of Multiple Biological Sequence Alignment Software Based on Social Behavior Model

PLAMENKA BOROVSKA
Department of Informatics
Technical University of Sofia
8 Kliment Ohridski boul., Sofia
BULGARIA
pborovska@tu-sofia.bg

VESKA GANCHEVA
Department of Programming and Computer Technologies
Technical University of Sofia
8 Kliment Ohridski boul., Sofia
BULGARIA
vgan@tu-sofia.bg

Abstract: - This activity is aimed to investigate and to improve the performance of the multiple sequence alignment software MSA_BG, for the case study of the influenza virus sequences. The objective is code optimization, porting, scaling and performance evaluation of the parallel multiple sequence alignment software MSA_BG for Intel Xeon Phi (the MIC architecture). For this purpose a parallel multithreaded optimization including OpenMP has been implemented and verified. The experimental results show that the hybrid parallel implementation utilizing MPI and OpenMP provides considerably better performance than the original code.

Key-Words: - Artificial Bee Colony, Bioinformatics, Hybrid Programming, High Performance Computing, Multiple Sequence Alignment, Parallel Programming, Performance.

1 Introduction

The biological sequence processing is essential for bioinformatics and life science. Scientists are now dependent on databases and access to the biological information. 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. This scientific area requires powerful computing resources for exploring the large sets of biological data. The parallel implementations of methods and algorithms for analysis of biological data using high-performance computing are important for accelerating the research.

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 [1]. This method is computationally difficult and is classified as a NP-hard problem [2] [3] [4].

The aim of the project is optimization and investigation of the parallel performance and efficiency of an innovative parallel method in the software MSA_BG for multiple alignments of biological sequences, which is highly scalable and locality-aware. The method is iterative and is based on the concept of Artificial Bee Colony metaheuristics and the concept of algorithmic and architectural space correlation.

The computational aspect of this project is to investigate the parallel performance with respect to efficiency and scaling of parallel multiple alignment software MSA_BG on the computer system EURORA [5], utilizing parallel OpenMP deployment and optimization of the MPI-based parallel implementation for the case study of influenza viral sequences comparison.

2 The Multiple Sequence Alignment Method MSA_BG Based on Artificial Bee Colony Metaheuristics

A new parallel highly scalable and locality-aware method MSA_BG for multiple alignments of biological sequences is presented in [6]. MSA_BG method is iterative and is based on the concept of Artificial Bee Colony (ABC) metaheuristics [7] and

the concept of algorithmic and architectural spaces correlation.

The granularity is hybrid - coarse granule computing for each node (multithreaded process) that runs multithreading (fine granule) of the cores within the computing node. It is appropriate to use a hybrid parallelization scheme in the case of a code which employs a hybrid granularity, in order to make efficiency use of supercomputer resources. The current parallel program implementation is based on MPI.

An overview of the computational algorithm is the following:

- *Scout 0 reads the sequences from an input file and stores them in the shared memory of the computational node (hive).* Scout bees in the swarms round certain subregions in the searching space and construct a potential solution. Once the scout bees obtain possible (feasible) solution, they return to the hive and begin to dance. The better the quality of a solution generated by a scout is, the higher is possibility to include it in the list of elite solutions. The food source is presented by possible sequence alignments. Scouts generate initial solutions through sequence alignment including gaps. The random generator that is used is based on the Mersenne Twister pseudo random generator that uses a 32-bit word length.
- *Onlookers watch the waggle dances, choose one of the possible solutions and evaluate it.* The quality of obtained solutions is determined by the grade of sequences' similarity. The higher the grade of the solution the better the quality of the obtained alignment, i.e., the criterion of optimality is a maximum similarity score. For the evaluation of the alignment quality, the following method is used.

An assessment by columns is done – in case of nucleotide sequences the numbers of symbols – A, G, C and T are counted. The numbers of symbols are compared and the nucleotides that occur mostly in the different columns are selected. Afterwards, the calculation of assessments in columns is formed, the so-called sequence-favorite (f_{ij}), which contains in each position the respective favorite nucleotide in the column (Table 1).

The sequences are compared to the sequence-favorite. The higher the similarity to the sequence-favorite, the higher is the grade of a sequence. A scoring matrix is built up which stores (in columns) the values of the evaluation function *S* for sequences (rows in the matrix). For the grade computation of a sequence (row) i in position j (column) the nucleotide a_{ij} and nucleotide–favorite f_{ij} are used:

$$S_{ij} = 0 \text{ in case } a_{ij} = \text{gap}$$
$$S_{ij} = 1 \text{ in case } a_{ij} = f_{ij}$$
$$S_{ij} = -1 \text{ in case } a_{ij} \neq f_{ij}$$

Table 1: Working set - the favourite sequence 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*

Table 2: 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 employed bees select one of the solutions and make attempts to improve it based on local search.* An approach for the modification of the aligned working set of sequences is used:

The column with counters of the scoring matrix S is reviewed and the row (sequence) with the lowest counter value is selected (sequence that differs most from the favourite). Using a random generator two indexes are selected: one for insertion of a gap (INS) and another for deletion of a gap (DEL) from the list of empty positions (DEL ≠ INS). The generated indexes are compared:

- If DEL > INS, then all characters in positions between INS and DEL are shifted one position to the right (shift_right).
- If DEL < INS the characters are shifted one position to the left (shift_left).

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.

- *Finally, the mother bee shall inform the queen bee of the colony and send her the quality of the best solution (elite solution).* The colony queen gets the quality of the elite decisions by hives' mother bees through collective communication reduction with the operation MAX and determines the

best one. Finally the queen bee sends messages to the mother bee holding the best solution to display the resultant sequences alignment.

The algorithmic framework for parallel multiple alignment of biological sequences in the MSA_BG method on the basis of the ABC methaheuristics is shown in [6].

```
  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
```

Fig. 1. Algorithmic framework for parallel multiple alignment of biological sequences in the MSA_BG method on the basic of the ABC methaheuristics.

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-favourite is created: Every MPI process iterates through each column on the array of the sequences in order to find the favourite nucleotides. The iterations are equal in number to the maximal 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-favourite 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 with the 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.

3 Profiling Results of the MPI Only Implementation

Scalasca has been used for the initial profiling of the code. Tests have shown that the functions 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 about 95% of the overall execution time (Fig. 2).

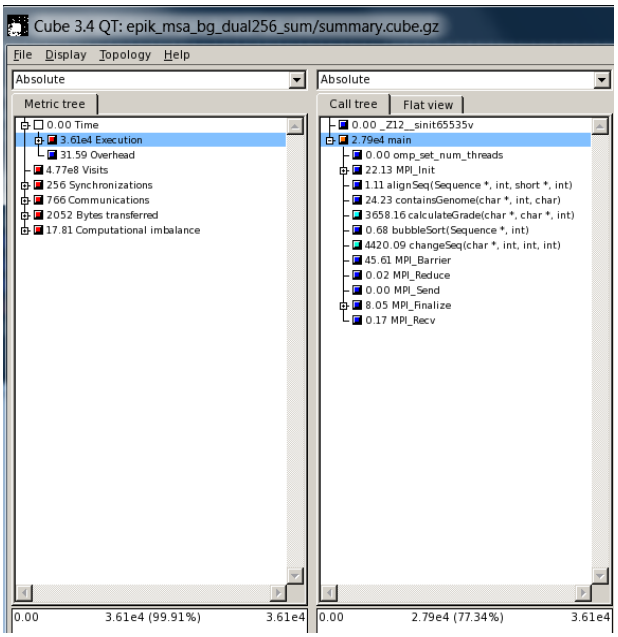

Fig. 2. Scalasca profiling results containing execution time percentages on BlueGene/P.

4 Experimental Framework

The experimental framework of the investigations is based on the computer system Eurora (Intel Xeon Phi) at CINECA. EURORA is a Cluster made of 65 nodes of different types [5]:

Compute Nodes: There are 64 16-core compute cards (nodes). Half of the nodes contain 2 Intel(R) Xeon(R) SandyBridge eight-core E5-2658 processors, with a clock rate of about 2 GHz, while the other half of the cards contain 2 Intel(R) Xeon(R) SandyBridge eight-core E5-2687W processors, with a clock of about 3 GHz. 58 compute nodes have 16GB of memory, but the safely allocatable memory on the node is 14 GB (see PBS resources: memory allocation). The remaining 6 nodes (with processors at 3 GHz clock rate) have 32 GB RAM. The EURORA cores are capable of 8 floating point operations per cycle. 32 compute cards have two nVIDIAK20 (Kepler) GPU cards and 32 compute cards have two Intel Xeon Phi accelerators. Login nodes: The Login node has 2 Intel(R) Xeon(R) esa-core Westmere E5645 processors at 2.4 GHz Intel Xeon (Esa-Core Westmere) E5645 2.4 GHz. All the nodes are interconnected through a custom Infiniband network, allowing for a low latency/high bandwidth interconnection.

Intel Xeon Phi is the first Intel Many Integrated Core (Intel MIC) architecture product. Each card consists of 60 physical cores (@1.1 Ghz) and each core is able to handle up to 4 threads using hyperthreading. Each core has one Vector Processing Unit able to deliver for each clock cycle: 8 Fused Multiply and Add (FMA) floating point operations in double precision or 16 Fused Multiply and Add (FMA) floating point operations in single precision. The Phi has a peak performance of 1056 GFlops in double precision and 2112 Gflops in single precision. Each Phi coprocessor has a RAM memory of 8 GB, and a peak bandwidth of 352 GB/s.

5 Hybrid Parallel MSA_BG Method Implementation

The concept of hybrid parallelism that was implemented in the MSA_BG software 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 method for multiple sequence alignment is presented in Fig. 3.

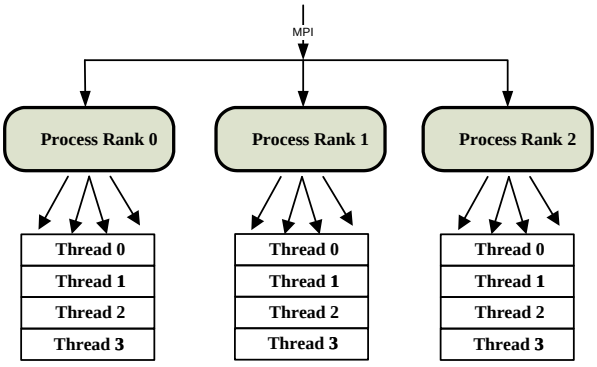

Fig. 3. Hybrid OpenMP/MPI parallel computational model of MSA_BG method.

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 EURORA system. The software is written entirely in C++ without any external dependencies to third party libraries.

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 behaviour 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 behaviour 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.

The steps 2, 3, 4 and 6 of the algorithm described in section 2 are performed in iterations until every sequence is processed. In each For Loop a *parallel for* OpenMP directive is placed. This results in having each OpenMP thread assigned to a different sequence every time until the work that needs to be done is completed.

Operations executed on the MIC accelerators:

1) Initial alignment of the biological sequences. This is a highly parallel task as there is no dependency between the sequences at this stage.

2) Creating the sequence favorite. The favorite sequence is created based on independent calculations on every genome. Again it is a perfect task for parallel multithreaded execution.

3) Calculating the grades of every genome.

4) Improving the working set of the sequences. It is an iterative process that can be executed in parallel.

6 Parallel Performance Evaluation and Results Analysis

The objective of the experiments is to estimate experimentally parallel performance parameters of the hybrid MPI/OpenMP parallel program implementation on the basis of the MSA_BG method. For scalability testing, the influenza virus protein data set obtained from Genbank [9] is used. Some experiments using various numbers of cores and 10000000 iterations have been conducted. The executions are performed on the Intel MIC architecture in native mode using different MPI/OpenMP configurations with a maximum of 240 threads on the MIC. The results are shown in Table 3 and Fig. 4.

Table 3: Experimental results of hybrid MPI/OpenMP software MSA_BG on the Intel Xeon Phi architecture.

MPI Nodes	Threads	Iterations	Execution Time (Sec)
2	15	10000000	6145.65
2	30	10000000	3093.87
2	60	10000000	1550.55
2	120	10000000	1121.47
2	240	10000000	1090.5

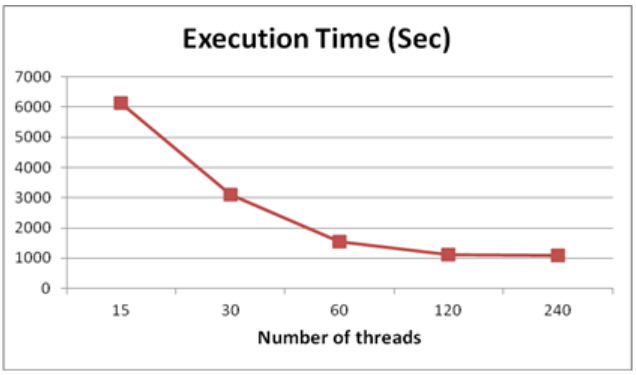

Fig. 4. Execution time of hybrid MPI/OpenMP software MSA_BG as a function of number of threads.

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. The experimental results show that the parallel program implementation for multiple sequence alignment scales well as the number of the cores increases (Fig. 5.).

7 Conclusion

The parallel software for multiple sequence alignment MSA_BG has been optimized by implementing a hybrid MPI/OpenMP parallelization scheme. The MSA_BG code was ported and is available for the Intel MIC architecture (EURORA system). It is best suited to work in homogenous

native Xeon Phi MPI mode in a combination of MPI tasks and threads.

Parallel performance evaluation and profiling of the multiple sequence alignment software MSA_BG utilizing MPI and OpenMP on the computer system EURORA have been investigated experimentally. Parallel performance parameters such execution time and speedup have been measured.

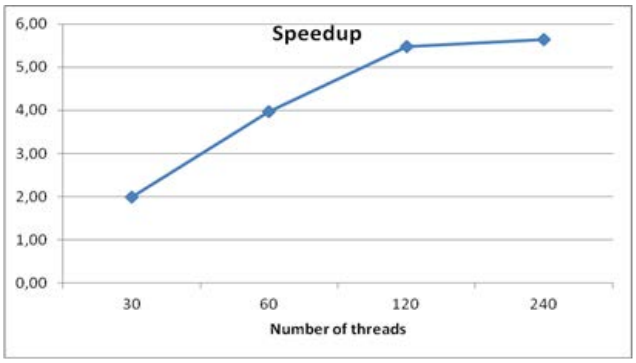

Fig. 5. Speedup of hybrid MPI/OpenMP software MSA_BG with respect to 15 threads.

The performance measurement and analyses show that the hybrid parallel program implementation utilizing MPI and OpenMP scales well when the number of cores increases and is well balanced both in respect to the workload and machine size.

The optimized code is universal and can be applies for other similar research projects and experiments in the field of bioinformatics and will allow researchers to conduct their experiments on even more powerful supercomputers. They will be able to perform simulations with very large amounts of data.

Acknowledgements

The paper is supported by the Research Project "Intelligent Method for Adaptive In-silico Knowledge Discovery and Decision Making Based on Analysis of Big Data Streams for Scientific Research" funded by the Bulgarian National Science Foundation, Bulgarian Ministry of Education and Science, Competition for financial support of Fundamental Research (2016) under the thematic priority: Technical Science, contract № ДН07/24 - 15.12.2016.

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] EURORA – Configuration, http://www.hpc.cineca.it/content/eurora-user-guide#systemarchitecture.

[6] 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.

[7] 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

[8] P. Borovska, V. Gancheva, Massively Parallel Algorithm for Multiple Sequence Alignment Based on Artificial Bee Colony, white paper, http://www.prace-ri.eu/IMG/pdf/wp114.pdf

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