IRACST - International Journal of Computer Science and Information Technology & Security (IJCSITS), ISSN: 2249-9555
Vol. 5, No3, June 2015

A Study to the effect of task granulation for the DNA multiple sequence alignment on Grid Computing

Mohamed Assal, Ahmed Said, Dina Mohamed and Nouran Osama
Faculty of Computer Science
MTI University
Cairo, Egypt

***Abstract*—DNA Multiple sequence alignment is widespread bioinformatics application that determines the similarity between a new sequence with other exist sequences. Along with the growth in the heterogeneous biological data, many research groups designed tools to analyze them. Integration of these biological data and tools is becoming one of the major topics in the field of bioinformatics. Grid computing provides the ability to perform high performance computing by taking advantage of many computers geographically distributed and connected by a network. Task granulation can greatly affect the processing time of the multiple sequence alignment on the grid. This paper shows a DNA multiple sequence alignment framework using the GDS Grid. Finally, the paper study the effect of the task granulation on the processing time and its effect on the computation to communication ratio.**

Keywords-- DNA Computing, Computational biology, Distributed computing, Grid computing, Bioinformatics

I. INTRODUCTION

Computational biology and bioinformatics is mainly concerned with the analysis and processing of biologic data related to the humans and other living orgasm. Laboratories all over the world producing data at huge rates. Consequently, there is a need to process the exponentially growing amount of biological data for scientific advances and to solve the data management problems. On the other hand, computational biology aims to solve these biological problems by utilizing computers to test and evaluate hypotheses and theories.

Research in the field of bioinformatics has grown significantly in the recent years as demands for more computing power increased. The solutions to these demands usually involve using parallel and/or distributed techniques. Grid Computing is an evolving technology to provide high performance computing in a virtual environment composed of a large number of computers connected through network[1].

Multiple sequences alignment involves more than two biological sequences, generally protein, DNA, or RNA. Multiple sequence alignment is computationally intensive problem and classified as a NP-Hard problem [2] [3].

Sequence alignment is a common bioinformatics application used to determine the degree of similarity between two sequences. Sequence alignment is the elementary operation of the DNA sequencing problem, due to the large number of DNA sequences applications. Sequences can be aligned across their entire length (global alignment) or only in certain regions (local alignment). Local sequence alignment plays a major role in the analysis of DNA and protein sequences [4] [5].

Several global and local sequence alignment tools use well-known algorithms such as The popular Smith-Waterman and Needleman-Wunsch used for local and global sequence alignment respectively [6] [7].

This paper describes framework implemented both the Smith-Waterman and Needleman-Wunsch algorithms along with multiple scoring matrices on the Grid Developing System (GDS) [8]. The paper also study the task granulation effect on the overall processing time of the grid framework.

II. SEQUENCE ALIGNMENT

The data of sequence are partitioned into DNA sequences and protein sequences. Each DNA sequence consists of four types of base A, T, C and G and each protein sequence is made up of 20 types of amino acid, hence any sequence can be represented as a string over specific alphabet.

DNA sequence alignment is a representation of the similarity between two or moresections of genetic code. It is used to compare these sections in a quantitative way.Biologists use the comparisons to discover evolutionary divergence, the origins ofdisease, and ways to apply genetic codes from one organism into another[9].

A. Pair wise Sequence Alignment

Pair wise sequence alignments are used to find diagnosticpatterns that characterize the two DNA families; to detect ordemonstrate homology between new sequences and existingfamilies of sequences.

Two general models view alignments in different ways:the first considers similarity across the full extent of thesequences (a global alignment); the second focuses onregions of similarity in parts of the sequences only (a localalignment). It is important to understand these distinctions, toappreciate that sequences are not uniformly similar, and thereis no value in performing a global similarity on sequencesthat have only local similarity. Therefore, finding local similaritymay produce more biological meaning and sensitive resultthan finding optimal alignment over entire length of thesequence[10].

B. Global Sequence Alignment

The idea of aligning two sequences (of possibly different sizes) is to write one on top of the other, and break them into smaller pieces by inserting spaces in one or the other so that identical Subsequences are eventually aligned in a one-to-one correspondence - naturally, spaces are not inserted in both sequences at the same position. In the end, the sequences end up with the same size. The following example illustrates aglobal alignment between the sequences A=”ACAAGACAGCGT” and B=”AGAACAAGGCGT”.

```
A  =  A  C  A  A  G  A  C  A  G  -  C  G  T
      |     |  |     |     |  |     |  |  |
B  =  A  G  A  A  C  A  -  A  G  G  C  G  T
```

Figure 1. Global Alignment of two sequences

The objective is to match identical subsequences as far as possible. In the example, nine matches are highlighted with vertical bars. However, if the sequences are not identical, mismatches are likely to occur as different letters are aligned together. Two mismatches can be identified in the example: a “C” of A aligned with a “G” of B, and a “G” of A aligned with a “C” of B. The insertion of spaces produced gaps in the sequences. They were important to allow a good alignment between the last three characters of both sequences [11].

An alignment can be seen as a way of transforming one sequence into the other. From this point of view, a mismatch is regarded as a substitution of characters. A gap in the first sequence is considered an insertion of a character from the second sequence into the first one, whereas a gap in the second sequence is considered a deletion of a character of the first sequence. In the previous example, A can be converted into B in four steps: 1) substitute the first “C” for a “G”; 2) substitute the first “G” for a “C”; 3) delete the second “C”; and 4) insert a “G” before the last three characters.

Once the alignment is produced, a score can be assigned to each pair of aligned letters, called aligned pair, according to a chosen scoring scheme. We usually reward matches and penalize mismatches and gaps. The overall score of the alignment can then be computed by adding up the score of each pair of letters. For instance, using a scoring scheme that gives a $+1$ value to matches and -1 to mismatches and gaps, the alignment of the two sequences in Figure 1 scores $9 * (1) + 2 * (-1) + 2 * (-1) = 5$.

The similarity of two sequences can be defined as the best score among all possible alignments between them. Note that it depends on the choice of scoring scheme. In the next sections, the problem of finding the best alignment of two sequences (an alignment that gives the highest score) will be addressed[11].

C. *Local Sequence Alignment*

The previous section described a type of alignment know as global alignment since we are interested in the best match covering the two sequences in their entirety. Frequently, though, biologists are interested in short regions of local similarity. A local alignment is one that looks for best alignments between “pieces”, or more precisely, substrings of both sequences.

Local alignment searches for segments of the two sequences that match well. There is no attempt to force entire sequences into an alignment, just those parts that appear to have good similarity, according to some criterion are considered [12].

The following example illustrates a local alignment between the sequences A=”ACAAGACAGCGT” and B=”AGAACAAGGCGT”.

```
A  =  A  C  A  A  G  A  C  A  G  C  G  T
                              |  |  |  |
B  =  A  G  A  A  C  A  A  G  G  C  G  T
```

Figure 2. Local Alignment of two sequences

Most commonly used algorithm for local alignment is Smith-Waterman algorithm [12].

D. *Substitution Matrices*

In the previous example, fixed scores were given for matches, mismatches and gap penalties. However, biologists frequently use scoring schemes that take into account physicochemical properties or evolutionary knowledge of the sequences being aligned. This is common when protein sequences are compared.

For instance, for some reason one might want to penalize the mismatch of an aspartic acid (D) with leucine (L) more heavily than a mismatch between the same aspartic acid with, say, histidine (H). Similarly, one may want to reward a match of two cysteine (C) better than two alanine (A) [11].

This type of scoring schemes is called *alphabet-weight* scoring schemes, and is usually implemented by a substitution matrix. Currently, two types of amino acid substitution matrices are being largely used by biologists for practical protein sequence alignment: PAM and BLOSUM. They were developed from different concepts but have the same structure. In fact, they are a series of matrices with varied degrees of sensibility.

The PAM matrices (acronym for point accepted mutations) are extrapolated from data obtained from very similar sequences to reflect an amount of evolution producing on average one mutation per hundred amino acids. The BLOSUM matrices (acronym for blocks substitution matrix), in contrast, were developed to detect more distant relationships [13]. In particular, BLOSUM50 and BLOSUM62 are being widely used for pairwise alignment and database searching [14].

Substitution matrices allow for the possibility of giving a positive score for a mismatch, what is sometimes called an approximate or partial match. For instance, the BLOSUM62 matrix returns a score of +2 for the substitution of a lysine (K) for an arginine (R) [11].

III. GRID COMPUTING

A grid is a type of parallel and distributed system that enables the sharing, selection, and aggregation of geographically distributed resources dynamically at runtime depending on their availability, capability, performance, cost, and users' quality of-service requirements. At the basic level, a grid can be viewed as an aggregation of multiple machines (each with one or more CPUs) abstracted to behave as one "virtual" machine with multiple CPUs.

Grid applications distinguished from traditional client server applications by their simultaneous use of large numbers of resources, use of resources from multiple administrative domains, complex communication topologies and severe performance requirements.

The GDS is a .NET based computational grid environment implemented in MTI University [8]. The GDS Grid allow the seamless aggregation of the computing power of multiple distributed machines connected through network into a virtual super computer. The GDS grid computing framework was conceived with the aim of making grid construction and development of grid software as easy as possible without sacrificing flexibility, scalability, reliability and extensibility [8].

GDS has practical capabilities of connecting up to 4096 workstations. In addition, GDS is hardware scalable in which workstations could be easily replaced with a high-end server through the GDS plug and play agent feature. The GDS will automatically utilize the new powerful resources in the new connected agents [15].

GDS Grids are constructed using two types of distributed components (or nodes). These are GDS Coordinator and GDS Agent according to their roles with respect to a grid application.

A. GDS Coordinator

Coordinator manages the execution of grid applications and provides services associated with managing thread execution. The coordinator distribute the tasks to agents according to certain scheduling policy. The default GDS scheduling policy distribute tasks to grid agents based on their available physical cores. The coordinator considers each physical core of each agent as a separate execution unit [8].

GDS separates the coordinator into several components, each component responsible for a specific task. Figure. 3 shows the different GDS Coordinator components [15]. GDS expose an Application Programming Interface (API) for both the Problem Decomposition and the Result aggregation components for the grid application developer to write the application to utilize the GDS grid.

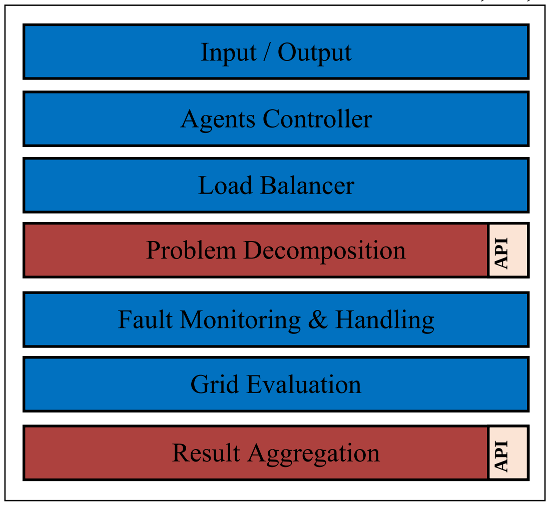

Figure 3. GDS Coordinator components

B. GDS Agent

The agent represent the worker unit in the GDS Grid. The agent registers itself with a GDS coordinator and waits to receive grid tasks to process [8]. Again, GDS expose an API for the grid application developer to utilize the agent's resources. Figure. 4 shows the different GDS agent components[15].

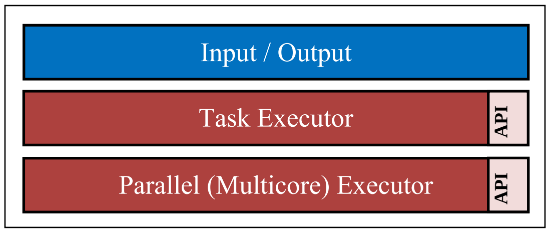

Figure 4. GDS Agent components

IV. THE IMPLEMENTED FRAMEWORK

The implemented DNA multiple sequence alignment framework is a tool developed using the GDS grid environment. The framework uses one GDS Coordinator and several agentsto align the DNA sequences.

The framework supports both the Global sequence alignment using Smith waterman and Local sequence alignment using Needleman Wunch algorithms along with 3 different substitution scoring matrices (BLOSUM62, PAM250 and Gonnet160).

In addition, the framework aims to study and overcome the effect of the computation to communication ratio. Therefore, to overcome the computation to communication ratio, the workload of each grid task must has heavier computation. The idea is to bundle more than one sequence from the dataset along with the unknown sequence. That is, the enhanced framework adds an option to choose the sequence bundle from

1 up to 4 sequences. For example 2 sequences bundle, means each grid task will contain 2 sequences from the dataset and the unknown sequence and 4 sequences bundle, means each grid task has 4 sequences from the dataset and the unknown sequence.

The framework has the ability to limit the dataset working space to a specific class in case that the dataset is classified into different classes.

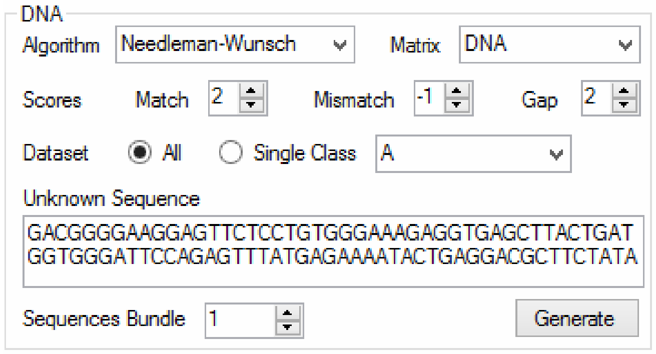
DNA
Algorithm: Needleman-Wunsch
Matrix: DNA
Scores — Match: 2, Mismatch: -1, Gap: 2
Dataset: [✓ All] Single Class A
Unknown Sequence:
GACGGGGAAGGAGTTCTCCTGTGGGAAAGAGGTGAGCTTACTGAT
GGTGGGATTCCAGAGTTTATGAGAAAATACTGAGGACGCTTCTATA
Sequences Bundle: 1

Figure 5. The framework user interface

The framework is been based on pairwise comparison to query large databases to find the maximum complete or partial alignment over the grid computing. Figure 5 shows the implemented framework user interface.

The GDS Coordintor use the problem settings and dataset supplied by the grid user to generator DNA sequence alignment task. Next, the coordinator divides the tasks equally according to GDS default load balancing policyto the connected agents. The agents start to execute the tasks one by one and send the results to the coordinator. If any failure happened during the execution, the coordinator handle this failure and recover from it. For example if one of the agent disconnected for any reason, the coordinator will reassigned the tasks assigned to the disconnected agent toother available agentsaccording to the load balancing policy. At last, the coordinator selects the sequence with maximum matching score.

V. Experiments and Discussions

The implemented framwork tested against 3 different datasets TABLE I. shows the 3 different datasets properties. Each dataset tested using the Smithwaterman algorithm and 3 bundles configurtions (1:1, 1:2 and 1:4). The experiments done using a GDS grid of 9 workstations (8 as agents and 1 as a coordinator). TABLE II. shows each workstation specifications.

TABLE I. Different Datasets Properties

ıset		Size	Sequence le (nucleotid
t		800	600
d		1600	600
d		19200	2000

TABLE II. Workstation Specification

	Specification
Operating System	Microsoft Windows Enterprise Service Pack 1
Processor	Intel Core i7-2600
Number of Cores	4 Physical & 4 Virtual
Actual Clock	3.701 GHz.
Memory	4 GB

The first experiment was against the 1st dataset and using the 3 different bundle configurations (1:1, 1:2 and 1:4). The expirement compare the sequential time (single core) incremental against up to 8 agents. TABLE III. Shows the results of this expirment. Figure 6. Shows the speedup of the expirment incremental up to the 8 agents.

The second expirement done against the 2nd dataset and the 3 different bundle configurations. TABLE IV. Shows the expirement reulsts and Figure 7. Shows its speed up.

Finally, The third was against the 3rd dataset and the 3 different bundle configurations. TABLE V. shows the expirement results and Figure 8. Shows its speed up ratio.

TABLE III. First Expirement Results

Number of Cores	**Processing Time (in minutes)**		
	1:1 Bundle	**1:2 Bundle**	**1:4 Bundle**
1 (Sequential)	4.38891	4.38891	4.38891
4	1.479	1.1556	1.10101
8	0.747	0.573	0.551
12	0.50674	0.40681	0.39049
16	0.49873	0.30803	0.30567
20	0.30313	0.25708	0.24979
24	0.2645	0.2152	0.2213
28	0.35199	0.20645	0.1883
32	0.23146	0.16398	0.1728

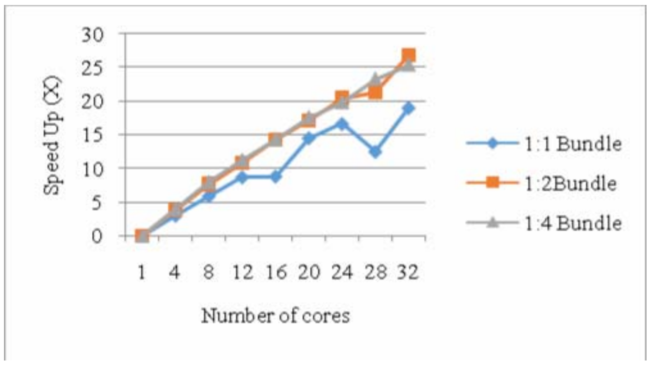

Figure 6. Speedup of the first expirment

TABLE IV. SECOND EXPIREMENT RESULTS

Number of Cores	**Processing Time (in minutes)**		
	1:1 Bundle	**1:2 Bundle**	**1:4 Bundle**
1 (Sequential)	8.77782	8.77782	8.77782
4	2.958	2.3112	2.20202
8	1.494	1.146	1.102
12	1.01348	0.81362	0.78098
16	0.99746	0.61606	0.61134
20	0.60626	0.51416	0.49958
24	0.529	0.4304	0.4426
28	0.70398	0.4129	0.3766
32	0.46292	0.32796	0.3456

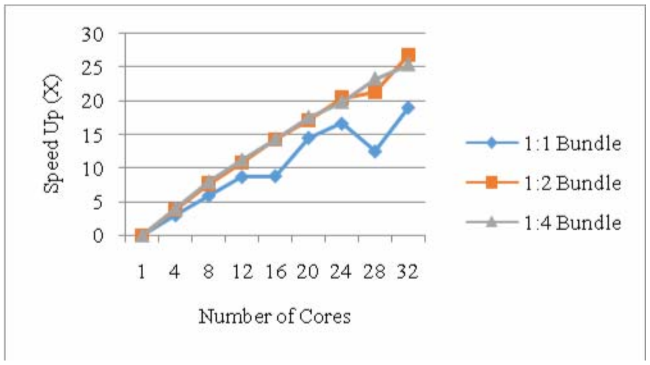

Figure 7. Speedup of the second expirment

TABLE V. THIRD EXPIREMENT RESULTS

Number of Cores	**Processing Time (in minutes)**		
	1:1 Bundle	**1:2 Bundle**	**1:4 Bundle**
1 (Sequential)	182.3112	182.3112	182.3112
4	63.96264	50.50008	46.76274
8	32.05206	25.45002	22.8267
12	21.54222	16.63398	15.40116
16	16.20594	12.66912	11.73402
20	13.1517	10.56132	9.48636
24	12.65958	9.4365	7.94646
28	11.52792	8.5212	7.13142
32	11.68182	7.55838	6.04134

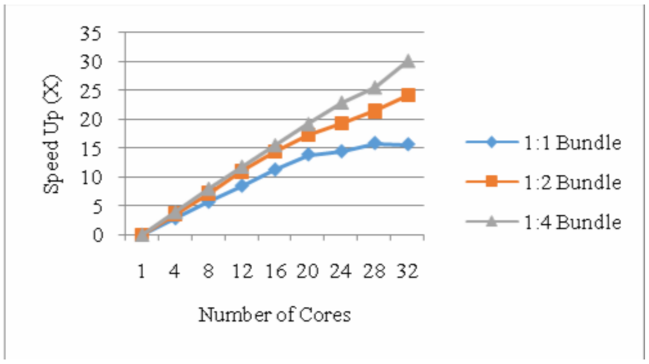

Figure 8. Speedup of the third expirment

The pervious expirements show that time decerses and the speedup increases as the number of cores increases. In addition, in the case of the 1:1 bundle expirement the speedup ratio suffer from the local saturation problem. Making the task more computational heavier through the using of task bundles (1:2 and 1:4) solved the local saturation problem. Expirements with 1:4 bundle achieved the best speed up ratio and didn't suffer from the local saturation problem.

Obviously, the gird system shows good scalability with respect to both the grid size and the workload size. In the case of small number of tasks the speedup obtained is not sufficient. Increasing the number of tasks results in speedup improvement because of the better utilization of the cores.

VI. CONCLUSION

The paper presented a grid DNA multiple sequence alignment framework implemented on the GDS Grid environemnt.

From the expirements its obvious, that Grid computing can be a good solution for the challenges faced in bioinformatics field. Since bioinformatics demands more computing power, integration of distributed, huge and complex data as well as applications of heterogeneous networks, Grid computing environments can be a right choice. The integration of a platform dedicated to biology into GDS grid opens up new opportunities in terms of computing resources and data storage.

The expirements proved that increasing the task workload achieve better speedup, better computation to communication ratio and a good solution to the local saturation problem.

REFERENCES

[1] I. Foster, "What is the Grid? A Three Point Checklist," [Online]. Available: http://www.mcs.anl.gov/~itf/Articles/WhatIsTheGrid.pdf. [Accessed 1 May 2015].

[2] L. W. a. T. Jiang, "On the complexity of multiple sequence alignment," *Journal of Computational Biology*, vol. 1, no. 4, p. 337–348, 1994.

[3] Y. L. a. Q. Y. S. Sze, "A polynomial time solvable formulation of multiple sequence alignment," *Journal of Computational Biology,* vol. 13, no. 2, p. 309–319, 2006.

[4] W. G. W. M. E. W. M. D. J. L. S.F. Altschul, "Basic Local Alignment Search Tool," *Journal of Molecular Biology,* vol. 215, no. 3, pp. 403-410, 1990.

[5] D. L. W. R. Pearson, "Improved Tools for Biological Sequence Comparison," in *Proceedings of The National Academy of Sciences - PNAS*, USA, 1988.

[6] M. W. T.F. Smith, "Identification of Common Molecular Subsequences," *Journal of Molecular Biology,* vol. 147, pp. 195-197, 1981.

[7] S. N. a. C. Wunsch, "A general method applicable to the search for similarities in the amino acid sequence of two proteins," *Journal of Molecular Biology,* vol. 48, no. 3, pp. 443-453, 1970.

[8] M. H. A. a. M. B. A. Said, "An Enhanced framework for Grid Computing Developing System (EGDS)," *Managerial Research Journal, Consultancy Research & Development Center,* 2012.

[9] M. Lehman, "Experiments with Algorithms for DNA Sequence Alignment," Simpson College, Indianola, Iowa.

[10] S. Vasantharathna, A. Kunthavai and R. Karuppayya, "AGAligner – DNA Local Sequence Alignment Using Alchemi Grid," *IRACST – Engineering Science and Technology: An International Journal (ESTIJ), ISSN: 2250-3498,* vol. 2, no. 3, 2012.

[11] S. A. d. C. Junior, "Sequence Alignment Algorithms," King's College London, University of London, London, 2003.

[12] T. Naveed, I. S. Siddiqui and S. Ahmed, "Parallel Needleman-Wunsch Algorithm for Grid," Bahria University, Islamabad, Pakistan, 2004.

[13] D. Gusfield, Algorithms on Strings, Trees, and Sequences, Cambridge University Press, 1997.

[14] R. Durbin, S. Eddy, A. Krogh and G. Mitchison, Biological Sequence Analysis, Probabilistic Models of Proteins and Nucleic Acids, Cambridge University Press, 1998.

[15] A. Said, Design and Building of a Framework for Grid Computing Developing System, Cairo: Arab Academy for Science, Technology & Maritime Transport, December 2012.

[16] E. Orabi, M. Assal, M. Abdel Azim and Y. Kamal, "Designing and Building a Framework for DNA Sequence Alignment Using Grid Computing," *International Journal of Advanced Computer Science and Applications,* vol. 5, no. 9, pp. 83-88, 2014.