IRACST – Engineering Science and Technology: An International Journal (ESTIJ), ISSN: 2250-3498, Vol.2, No. 3, June 2012

AGAligner – DNA Local Sequence Alignment Using Alchemi Grid

S.Vasantharathna, *A.Kunthavai, R. Karuppayya
Coimbatore Institute of Technology
Coimbatore, Tamilnadu, India

***Abstract*—This Sequence alignment is a popular bio informatics application that determines the degree of similarity between nucleotide or amino acid sequences that is assumed to have same ancestral relationships. Traditional accurate algorithm, such as Smith Waterman generates optimal alignment, either miss targets or its computational complexity is too high. So in this paper, an algorithm AGAligner is developed which provides a cross platform environment, cost-effective and more computing power to pair wise local sequence alignment. Grid is one of the most emerging technologies of cost effective computing paradigm for large class of data and compute intensive application which enables large-scale aggregation and sharing of computational data and other resources across institutional boundaries. This paper takes on the challenge of designing an optimal, fast and efficient algorithm for evaluating local-alignment using Alchemi Grid.**

Keywords- ***Sequence alignment, Bioinformatics, Local alignment, Alchemi Grid.***

I. INTRODUCTION

Bioinformatics and computational molecular biology are concerned with the use of computing and mathematical sciences as tools to advance traditional laboratory based biology. The need to process an exponentially growing amount of biological information for scientific advances and to understand its role in heredity, chemical processes within the cell, drug discovery, sequence alignment, evolutionary studies etc. have created new problems that are of interdisciplinary nature. Sequence alignment is one of the most important fundamental operations in bioinformatics. It has been successfully applied to predict the function, structure and evolution of biological sequences. It can reveal biological relationship among organisms, for example finding evolutionary information, determining causes and cures of diseases, or information about a new protein.

Sequence alignment is a basic operation of the DNA sequencing problem, mainly due to the large number of DNA sequences. 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 [1-3]. It is the basic step of many applications like detecting homology, finding protein structure and function, deciphering evolutionary relationships, etc. There exists several local sequence alignment programs that use well-known algorithms [4,5,8] or their heuristic versions [6,7,9,10].

However, most of the currently available alignments algorithms provides optimal or near optimal results and have long been limited by their computation speed. As the amount of sequence data has been exponentially growing and more genome sequencing projects are approaching their finale one after another, an urgent demand for an efficient alignment algorithm to perform large-scale sequence analysis is a key issue during the post genome era. It is certainly impractical to keep upgrading the computer hardware and increasing the equipment expenses, especially for some bioinformatics institutes with very limited computer resources. So this application requires cross platform, cost-effective and more computing power algorithm for sequence matching and searching a sequence in database. Grid [11] is one of the most emerging technologies of cost effective computing paradigm for large class of data and compute intensive application which enables large-scale aggregation and sharing of computational data and other resources across institutional boundaries. Tahir et al in Parallel Needleman-Wunch developed Needleman-Wunsch algorithm [12] to globally align two DNA Sequences using four processors, it reduces the time to O (N+M) using Alchemi Grid v 0.6.0[11].

This paper describes AGAligner, the parallel version of the Smith-Waterman algorithm for local sequence alignment problem. The Alchemi grid v 0.8.0 [13] has been used to the run the algorithm in grid environment.

II. SEQUENCE ALIGNMENT

Sequence alignment is a fundamental problem in Bioinformatics. It is a way of arranging the primary sequences of DNA, RNA, or protein to identify regions of similarity that may be a consequence of functional, structural, or evolutionary relationships between the sequences.

The alignments are simply the mathematical models whose behaviors can be modified using parameters. Different models exist like multiple sequence alignment which are designed to encapsulate a variety of physical characteristics of biological sequences.

A. Pair wise Sequence Alignment

Pair wise sequence alignments are used to find diagnostic patterns that characterize the two DNA families; to detect or demonstrate homology between new sequences and existing families of sequences.

Two general models view alignments in different ways: the first considers similarity across the full extent of the sequences (a global alignment); the second focuses on regions of similarity in parts of the sequences only (a local alignment). It is important to understand these distinctions, to appreciate that sequences are not uniformly similar, and there is no value in performing a global similarity on sequences that have only local similarity. So, finding local similarity may produce more biological meaning and sensitive result than finding optimal alignment over entire length of the sequence.

B. Local Sequence Alignment

In many biological applications, two DNA sequences may not be highly similar in their entire length, but may contain regions that are highly similar, because only some internal sections of those strings may be related. When comparing such DNA sequences, local alignment becomes critical because local similarity finds out highly conserved regions in the DNA sequence and it is the preferred choice for biological applications. The reason for choosing local alignment algorithm is it highlights conserved regions between two sequences and it yields more homological information.

Let Σ= {A, G, C, T}, $\Gamma = \Sigma \cup \{\text{'-'}\}$ and S_1, S_2 are two sequences over Σ with length n_1 and n_2 respectively. Let us assume that local pair wise alignment of S_1 and S_2 is A and B and starts at i^{th} character in S_1 and j^{th} character in S_2 over Σ with the following criteria:

- $Ai = Bj$, $Ai+m = Bj+m \in \Sigma$
- $Ak, Bl \in \Gamma$ where $i < k < i+m$, $j < l < j+m$ and m- length of the alignment
- $Ai+k = Bj+k \in \Sigma$ OR if $Ai+k$ is {'-'} then $Bj+k \in \Sigma$ OR if $Bj+k$ is {'-'} then $Ai+k \in \Sigma$ where $2 < k < m - 1$

AB for S1 = GAATTCAGTTA and S2 = GGATT GAT is shown in figure 1 which satisfies the above criteria. If match = 2, mismatch = -1 and gap = -1 then the alignment score of AB is 7 as shown in figure 1.

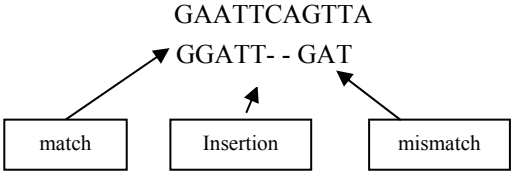

Figure 1. Sequence Alignment Example

In this paper an optimally fast computation solution is achieved through a parallel version of Smith Waterman algorithm by making use of the Alchemi Grid as the processing engine through which the time complexity in DNA alignment has been reduced to O (n+m).

III. ALCHEMI GRID

A grid is a type of parallel and distributed system that enables the sharing, selection, and aggregation of geographically distributed "autonomous" 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 are distinguished from traditional client server applications by their simultaneous use of large numbers of resources, dynamic resource requirements, use of resources from multiple administrative domains, complex communication structures and stringent performance requirements.

Alchemi is a .NET-based Enterprise Grid framework that allows seamless aggregation of the computing power of networked desktop computers into a virtual supercomputer for executing resource intensive applications. The Alchemi 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.

Alchemi Grids are constructed using three types of distributed components (or nodes). These are Alchemi Manager, Alchemi Executor and Alchemi Owner according to their roles with respect to a grid application.

A. Alchemi Manager

Manager manages the execution of grid applications and provides services associated with managing thread execution. It is deployed as an executable. An optional sub-component of the Manager is the Cross Platform Manager, which is deployed as a web service.

B. Alchemi Executor

The Executor is the individual system that executes individual grid threads and provides services associated with executing threads. It is deployed as an executable. An Executor can be configured to be dedicated i.e. the Manager initiates thread execution directly or non-dedicated that is that thread execution is initiated by the Executor on a volunteer basis via a screen saver or some other user-defined options.

C. Alchemi Owner

The Owner owns an application and provides services associated with the ownership of an application (and its constituent threads). The Owner is implicitly created by the Alchemi API.

Alchemi's layered architecture for a desktop grid computing environment is shown in Figure 2 Alchemi follows the master-worker parallel computing paradigm in which a central component dispatches independent units of

parallel execution to workers and manages them. In Alchemi, this unit of parallel execution is termed 'grid thread' and contains the instructions to be executed on a grid node, while the central component is termed 'Manager'.

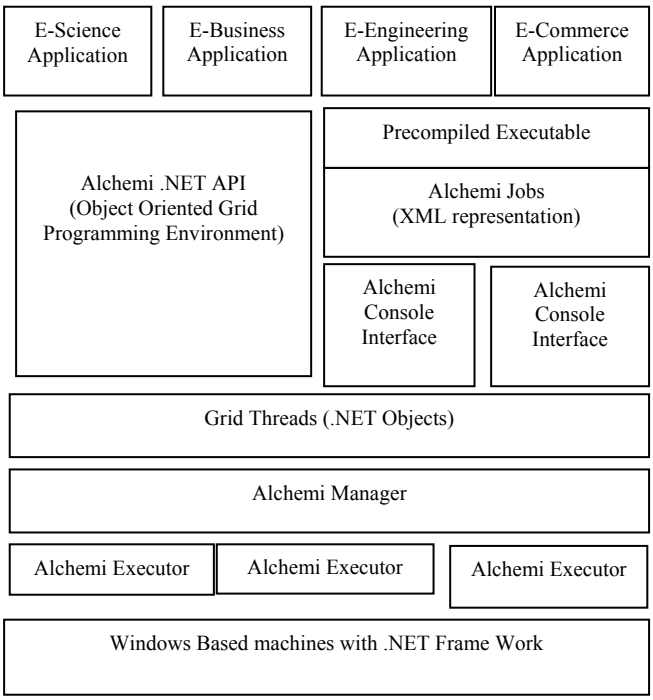

Figure 1 Alchemi Framework

IV. AGALIGNER

The developed AGAligner is a tool that uses one Alchemi manager and two or more executor for its functioning. Alchemi manager controls the executors connected with it. Alchemi manager and executors are prepared for the grid setup. Once the manager is started, the executors can connect to the manager. The whole process of sequence alignment is divided into several threads. Threads are a unit of execution of each cell. Three matrices namely scoring matrix, pointer matrix and thread distribution matrix are used. The thread distribution matrix stores the order of distribution of threads to executors. The scoring matrix stores the DNA sequences and their calculated scores and the pointer matrix stores the cell from which score was derived (i.e. top, left or diagonal). The threads are distributed to the executors. The CPU which receives the thread performs the scoring and after which corresponding values are stored in the score and pointer matrix. The score matrix obtained in the above step is back propagated along the path from which it was derived (with the help of pointer matrix) to produce the required alignment. There are four steps in AGAligner.

- Connection Establishment
- Initialization
- Matrix Fill
- Alignment

A. Connection Establishment

Alchemi manager is responsible for controlling various activities across the grid. Once the manager is started, the executors can connect with the grid. The manager determines the number of executors available and distributes the threads evenly across the connected executors. It monitors the resource utilization in each executor. Once the thread operations in each thread are completed, the results are sent back to the manager.

B. Initialization

In this step two DNA sequences are represented in a 2-dimensional array. Initialization matrix is created with m+1 row and n+ 1 column where m and n are the length of the two DNA sequences. The values in the first row and column are filled using the scoring functions shown in equation (1) and (2).

$$M(i,0) = 0, \quad 0 \le i \le m \tag{1}$$

$$M(0,j) = 0, 0 \le j \le n \tag{2}$$

C. Matrix Fill

Thread distribution matrix in the Alchemi manager is used to determine the order of distribution of threads for its execution. Each thread contains the cell positions (row and column) of the score matrix whose value are to be calculated in the executor. Thread distribution matrix is filled with the row and a column position which determines the order in which the threads should be distributed amongst the available CPUs. The threads are then distributed to different CPUs. The values in each cell score matrix M (i,j) are determined as in equation 3.

$$M[i,j] = \text{Max}\{ M[i-1,j-1] + \text{sub}(A[i], B[j]);\ M[i-1,j] + \text{del}(A[i]);\ M[i,j-1] + \text{ins}(Bj) \} \tag{3}$$

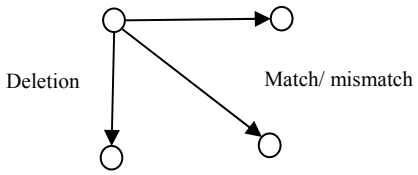

Figure 3 Sequence Alignment Path

Figure 4 shows the data dependencies in Smith-Waterman local alignment algorithm. As mentioned in figure 3 there are three possible alignments to choose from when calculating one element : alignment of the symbol in the row

considered with gap – horizontal arrow, alignment between the symbols in the row and column considered with match or mismatch - diagonal arrow, and alignment of the symbol in the column considered with a gap - vertical arrow. This means that rows or columns can't be computed in parallel. The only elements on each successive anti-diagonal labeled dashed line in Figure 4 are processed in parallel. These data dependencies present a serious challenge for sufficient parallel execution.

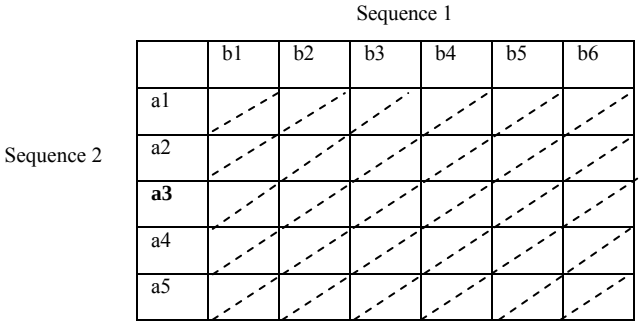

Figure 2 Data Dependency in Smith Waterman

2,2							
3,2	2,3						
4,2	3,3	3,4					
5,2	4,3	3,2	2,5				
6,2	5,3	4,4	3,5	2,6			
7,2	6,3	5,4	4,5	3,6	2,7		
8,2	7,3	6,4	5,5	4,6	3,7	2,8	
9,2	8,3	7,4	6,5	5,6	4,7	3,8	2,9
9,3	8,4	7,5	6,6	5,7	4,8	3,9	
9,4	8,5	7,6	6,7	5,8	4,9		
9,5	8,6	7,7	6,8	5,9			
9,6	8,7	7,8	6,9				
9,7	8,8	7,9					
9,8	8,9						
9,9							

Figure 3 TDM Matrix

The TDM matrix shown in Figure 5, explains that in first iteration only 2,2 will be calculated, in 2nd iteration 3,2 and 2,3 will be calculated in parallel and in third iteration 4,2 , 3,3 and 2,4 will be calculated in parallel and so on. The cell 2,2 and 9,9 in parallel cannot be calculated in parallel because 9,9's result will not be available until 9,8 and 8,9 are not present. This Data matrix clearly indicates that more CPUs means more parallel assigning, which will result in parallel calculation, biggest parallel calculation that can be performed in this step required eight CPUs for 8th iteration. Which means if less than 8 processors are available (let's assume 5) are available then 5 values will be calculated in parallel and the remaining 3 values will be calculated in parallel to complete eighth iteration. Alignment score and the alignment are derived from the pointer matrix. The algorithm for developed AGAligner is shown in figure 6.

```
//Input: two DNA Sequence , Output: local sequence alignment
//TDM : stores the order in which the alignment takes place ; x, y :
position in the matrix that has
// to be evaluated for score; A: score matrix, w(x,y):weight assigned based
on indel ; m, n- DNA
//Sequence length.
 AGALIGN()
{   Call Init();
    Call TDM ()
    Distribute the parallel tasks of alignment to all the CPU's available;
   Call ALIGNER(x,y) for all x,y where 0<x<m and 0<y<n :
   Pointer matrix obtained is back propagated to obtain the alignment.
}
Function INIT ()
{   Input the two DNA sequences;
   Initialize the score matrix using eqn (1) & (2) ;
}
 Function TDM ()
{     while (till half of init_matrix is filled)
      row no =current row to be filled
      column no =first column to be filled
      fill init_matrix using  row number-- and column number ++
     while(till the remaining half of init_matrix is filled)
     row no =final row to be filled
     column no =current column to be filled
     fill the init_matrix using  row number -- and column number ++
}
Function Aligner(x,y)
{    Calculate the score:
      Diagonal= A(x-1,y-1)+w(ai,bi)
      Left=A(x-1,y-1)+w(ai,_)
     Up= A(x-1,y-1)+w(_,bi)
     Where w(a,b) corresponds to the score for the current setup.
     Update score matrix and pointer matrix
     Return (max(diagonal, left, up))
}
```

Figure 6 AGAligner Algorithm

V. RESULTS AND DISCUSSION

The developed AGAligner takes two different real DNA sequences Oryza Sativa Japonica Group GSS and EST genomic sequences [14] and the compared with JAligner [15] which is a JAVA implementation of Smith waterman algorithm as shown in figure 7.

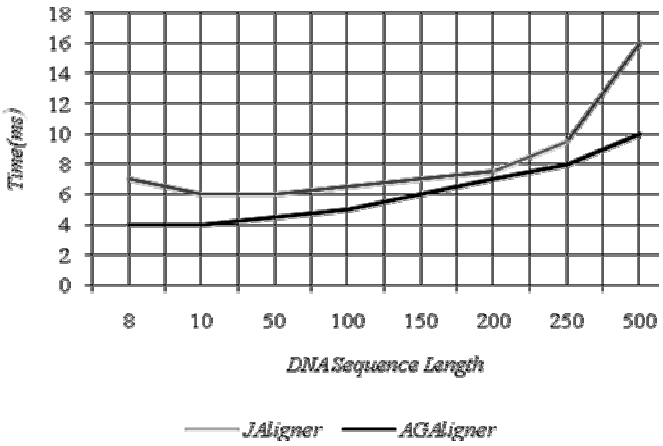

Figure 7 Comparison Figure 7 Comparison chart between Jaligner and AGAligner

Experimental result shows that the time complexity of traditional Smith Waterman algorithm is O (m*n) where developed AGAligner is O (m + n) where 'm' and 'n' are the length of the input sequences.

The figure 8, 9, 10 shows the resource usage of Alchemi executor for various sequence lengths. The graphs are drawn using CPU power percentage in Y-axis and time variation in milliseconds on the X-axis.

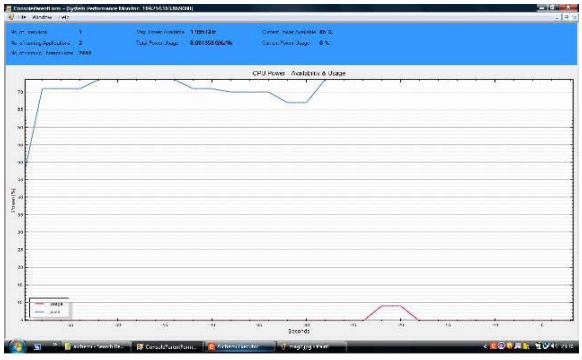

Figure 8 CPU usage graph for DNA sequence of length 50

From figure 8 it is inferred that the time duration taken by the executor for calculating the score of DNA sequences of length 50 is 10ms and the CPU Usage for the same is 15%. The fig 9 shows the CPU usage of the executor when a DNA sequence of length 100. The time duration taken by the executor for calculating the scores is 12ms and the CPU Usage for the same is 20%.

The figure 10 shows the CPU usage of the executor when a DNA sequence of length 500 is given as input to AGAligner. The time duration taken by the executor for calculating the scores is 15ms and the CPU Usage for the same is 25%.

From the figure 8, 9, 10 it is justified that the Alchemi Executor's CPU usage depends on the length of the DNA sequence that is the longer the sequence, higher is the CPU usage.

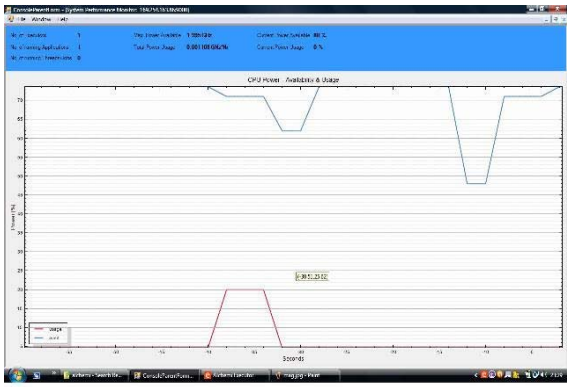

Figure 9 CPU usage graph for DNA sequence of length 100

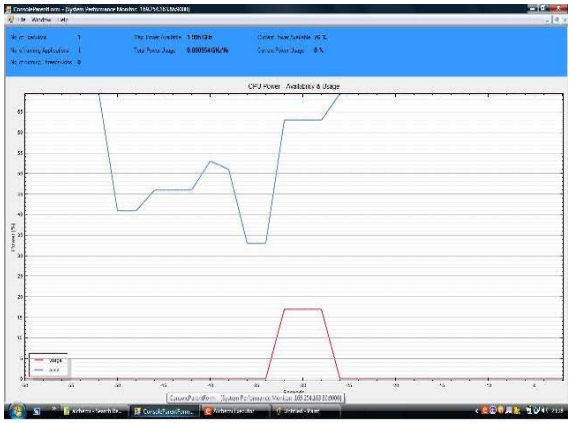

Figure 10 CPU usage graph for DNA sequence of length 500

VI. CONCLUSION

AGAligner algorithm for DNA local sequence alignment using Alchemi grid has been developed. Score of individual cell of score matrix is distributed and calculated using data dependency property of local sequence alignment using available the CPUs. Experimental result shows that the time complexity of traditional Smith Waterman algorithm is O (m*n) where AGAligner is O (m + n) where 'm' and 'n' are the length of the input sequences.

REFERENCES

[1] Pearson WR, Lipman DJ: Improved Tools for Biological Sequence Comparison. Proc Natl Acad Sci U S A, 85(8):2444-2448, 1988.

[2] Altschul SF, Gish W, Miller W, Myers EW, Lipman DJ: Basic Local Alignment Search Tool. Journal of Molecular Biology, 215(3):403-410, 1990.

[3] Altschul SF, Madden TL, Schäffer AA, Zhang J, Zhang Z, Miller W, Lipman DJ: Gapped BLAST and PSI-BLAST: A New Generation of Protein Database Search Programs. Nucleic Acids Research , 25(17):3389-3402, 1997.

[4] Smith TF, Waterman MS: Identification of Common Molecular Subsequences. Journal of Molecular Biology, 147:195-197, 1981.

[5] Sellers PH: Pattern Recognition in Genetic Sequences by Mismatch Density. Bulletin of Mathematical Biology, 46(4501-514, 1984.

[http://www.springerlink.com/content/2v4477481102w030].

[6] Pearson WR: Effective Protein Sequence Comparison. Methods in Enzymology , 266:227-259, 1996,.

[7] Pearson WR: Flexible Sequence Similarity Searching with the FASTA3 ProProgram Package. Methods in Molecular Biology, 132:185-219, 2000.

[8] Needleman, S.B. and Wunsch, C.D. A general method applicable to the search for similarities in the amino acid sequences of two proteins. J. Mol. Biol., 48, 443-453, 1970.

[9] Altschul SF. Gap costs for multiple sequence alignment. J Theor Biol, 138:297-309, 1989.

[10] Kent, W., and Zahler, A. Conservation, regulation, synteny, and introns in a large-scale C. briggsae–C. elegans genomic alignment. Genome Res, 10(8), 1115–1125, 2000.

[11] Krishna N. and Akshay L. and Dr. Rajkumar B., 2002, Alchemi Documentation [online], University of Melbourne. Available: http://www.cloudbus.org/~alchemi/doc/0_6_1/index.html

[12] Tahir Naveed, Imitaz Saeed Siddiqui, Shaftab Ahmed. Parallel Needleman-Wunsch Algorithm for Grid, Proceedings of the PAK-US International Symposium on High Capacity Optical Networks and Enabling Technologies (HONET 2005), Islamabad, Pakistan, Dec 19 - 21, 2005.

[13] http://www.cloudbus.org/~alchemi/0_8_0.html

[14] www.ncbi.nlm.nih.gov - Oriza Sativa DNA sequence

[15] http://jaligner.sourceforge.net/ - Jaliger