World Academy of Science, Engineering and Technology
International Journal of Bioengineering and Life Sciences
Vol:1, No:1, 2007

Parallezation Protein Sequence Similarity Algorithms using Remote Method Interface

Mubarak Saif Mohsen, Zurinahni Zainol, Rosalina Abdul Salam, Wahidah Husain
School of Computer Sciences
University Sains Malaysia
11800 Penang, Malaysia
mubarak_saif@yahoo.com,{zuri,rosalina, wahidah}@cs.usm.my

Abstract**—** One of the major problems in genomic field is to perform sequence comparison on DNA and protein sequences. Executing sequence comparison on the DNA and protein data is a computationally intensive task. Sequence comparison is the basic step for all algorithms in protein sequences similarity. Parallel computing is an attractive solution to provide the computational power needed to speedup the lengthy process of the sequence comparison. Our main research is to enhance the protein sequence algorithm using dynamic programming method. In our approach, we parallelize the dynamic programming algorithm using multithreaded program to perform the sequence comparison and also developed a distributed protein database among many PCs using Remote Method Interface (RMI). As a result, we showed how different sizes of protein sequences data and computation of scoring matrix of these protein sequence on different number of processors affected the processing time and speed, as oppose to sequential processing.

Keywords**—** Protein sequence algorithm, dynamic programming algorithm, multithread

I. INTRODUCTION

Today, many research works are being carried out by biologist to understand the biological function of the genome. The genome is the complete set of DNA molecules inside any cell of living organism that is passed from one generation to its offspring. DNA is abstracted as a long text over a four-letter alphabet, each representing a different nucleotide: A, C, G and T. It is recognized as what makes two livings thing being biologically similar or distinct. Protein is a linear sequence of simpler molecules called amino acids. Twenty different amino acids found in protein [1], and they are identified by A, C, D, E, F, G, H, I, K, L, M, N, P, O, R, S, T, U, W and Y. Like the DNA, proteins are conveniently represented as a string of three letters expressing its sequence of amino acids. The amino acid sequence of a number of proteins can be compared to determine whether the relationship exist between them could have occurred by chance [2].

Computational method can be used to identify genes and their function including statistic, sequence similarity, motif, profiles, protein folds and probabilistic models [4]. Using this method, it is possible to develop characteristic genome signatures, assign functions to genes, identify metabolic pathways and discover potential drug binding sites [5], [6].

One of the powerful methods to infer the biological function of gene is by doing sequence similarity searching on protein and DNA sequence in database. Protein or genes that have the similar sequence are likely to perform the same function or structure [4]. Two sequences are compared because we want to identify similarities and differences between them. A typical approach to solve this problem is to find a good and plausible alignment between the two sequences. Then, given an appropriate scoring scheme using BLOSUM [6] or PAM250 [6], their similarity can be computed. The following are some definition given to align the sequence [1].

Let say, S and T are strings. An alignment A maps string S' and T' that may contain space character, where

- |S'| =|T'|, where |S'| and |T'| denotes the length of S' and T' respectively
- The removal of all spaces from S' and T' leaves S and T respectively.

The *value* of alignment A is

$$\sum_{i=1}^{l} \sigma(S'[i], T'[i]), \text{ where } l = |S'| = |T'|, \quad (1)$$

S'_i denotes the *i*th character of S'

Let say sequence S = ACAAGACAGT and sequence T = AGAACAAGGCGT, then S' = ACAAGACAG-CGT and T' = AGAACA-AGGCGT. 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 that gives a +1 value to matches and –1 to mismatches and gaps, the alignment scores is 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.

II. RELATED WORK

Sequence similarity requires sequence comparison to be performed. Two general classes of sequence comparison used to calculate similarity scores to infer sequence similarity are heuristic algorithm and exhaustive algorithm. The first method is currently widely used in practice, such as BLASTP and FASTA [7]. The sequence comparison method base on heuristic is faster but do not produce optimal results and do

not guarantee to calculate an optimal score for every sequence in a database. The second method based on dynamic programming such as Needleman–Wunsch algorithm [2] and Smith-Waterman algorithm [1]. The Needleman- Wunsch Algorithm is based on global sequence alignment, which was a method for maximizing the amount of similarity between two sequences. Sellers [9] described another global alignment method, which minimized the differences between the sequences and computed a distance measure. The Smith – Waterman alignment algorithm is base on the local alignment approach that allow user to determine the two protein sequences are distinctly related. This algorithm can be used to compute the optimal alignment score for creating actual alignment. It was memory space proportional to product of the length of two sequences. Smith *et al.* [1] proved that these two methods were equivalent with appropriate substitution scores and gap penalties. Goad and Kanehisa [10] also made some refinements to the Needleman-Wunsch algorithm. When alignments are computed in the context of database searching, the dynamic programming algorithms described by Smith and Waterman [11] and Gotoh [12] are in general too time-consuming to be practical. The time complexity of these algorithms is *O(mn)* where *m,n* are the length of two sequences respectively. The most time spent in these algorithms is calculating the matrix; so research work is focus on parallelize the dynamic programming algorithm (DPA) to speed up the process of calculation the matrix.

III. Methodology

Our parallel DPA is based on the existing DPA [13]. DPA consists of two parts that are the calculation of scores indicating the similarity between the two given sequences, and the identification of the alignment(s) that lead to the score(s). The data structure used is a two dimensional array is called similarity matrix(SM). This SM is used to represent all possible alignments that can be constructed from the two sequences.

Comparison of two sequences, X = GGATAGG and Y= TGATGGAGGT, using the DPA technique is illustrated in Fig. 1. The sequences are placed along the left margin (X) and along the top (Y). The SM is initialized with zeros along the first row and first column so that alignments between subsequences are not penalized by gaps on its left and right. The other elements of the matrix are calculated by finding the maximum value using the following equation

$$SM(I,J) = \max \begin{cases} SM(I,J-1) + gp \\ SM(I-1,J-1) + ss \\ SM(I-1,J) + gp \end{cases} \quad (2)$$

The *gp* in equation (2) is the gap penalization and *ss* is the substitution score.

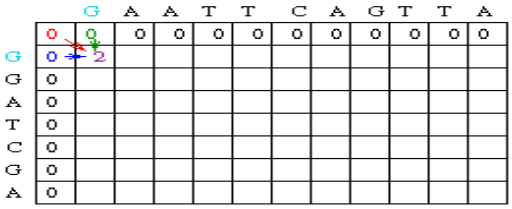

Fig. 1: Matrix filling step

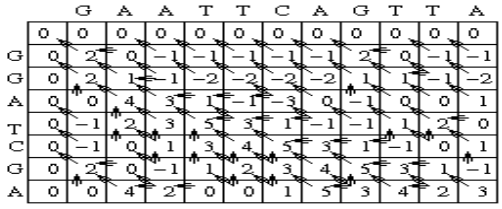

Fig. 2: The completed score matrix

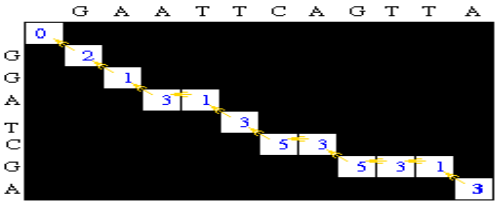

Fig. 3: Completed trace back

Following this recurrence equation, the matrix is filled from top left to bottom right with entry (i; j) requiring the entries (i; j - 1), (I-1; j-1), and (i -1; j). By choosing the maximum value at the SM (n,m) the best score is found and stored. Once the SM is computed as shown in Fig. 2, the second part of the algorithm will identify the sequence alignments. Each matrix element, a trace-back procedure is applied to find out the actual base pairs that constitute the alignment. Starting at the end of the alignment and moving backwards to the beginning, this procedure follows a path like the ones described by arrows in Fig. 3 and the optimal alignment for "GAATTCACTTA" and "GGATCGA" is "GGA-TC-G- -A".

In order to enhance the algorithm for local and global alignment for pair wise protein sequence data, new version of DPA were proposed. There are as follows:

(a) Database preprocessing
(b) Distributed database
(c) Multithreading DPA

Each of this proposed solution would be discussed in detail starting with database preprocessing, distributed databases and followed by the multithread dynamic programming algorithm.

A. Database Preprocessing

Each record in the database contains the sequence and others information, including the identification (ID), source organism, accession numbers, gene or protein name, and more. To reduce the time on disk reading and to perform effective database searches, the database text files should be parsed and stored in a more efficient format. In this format, all protein sequences with its ID are stored in one file, while

other information about each protein sequence, e.g. organism, accession numbers, gene or protein name etc. are stored in the other file.

B. Distributed Database

Using RMI, more than one sequences will be read at the same time and calculate the protein sequences similarity parallely. The reading of the protein sequence is shown as Fig. 4. The multithreaded system is design to solve the reading sequence and compute the scoring matrix. The design mainly consists of three main items that are; master node serve as databank that hold the sequence data protein, i.e. PC that is needed to submit and collect the result from the slaves nodes, and set of slave nodes which are needed to multithread the sequence and compute the scoring matrix parallels. Once the user input the protein sequence to be searched, the system will read more than existed protein sequences from the databank and distribute the pair of sequences to different PCs. This will help to keep all processors busy through most of the computation and the speed of the algorithm will be increased.

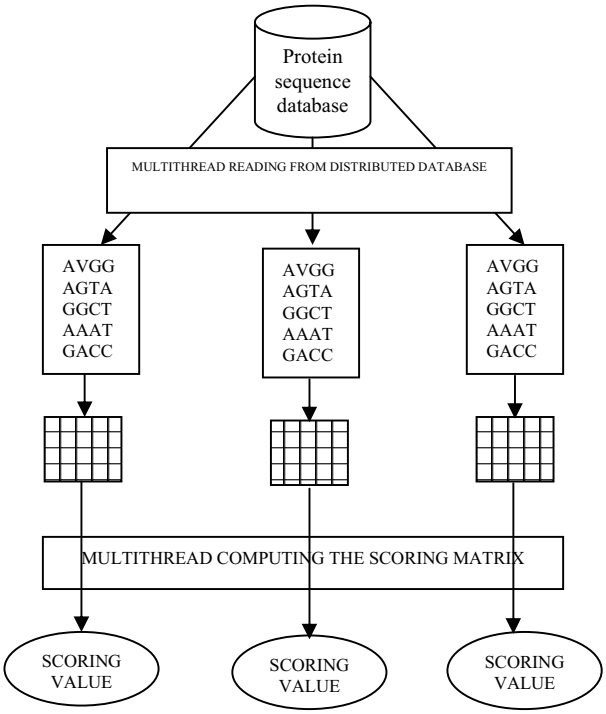

Fig 4: Distributed DB and computing scoring matrix

C. Multithreading DPA

Given the data presented by the DPA in section III, the SM can be filled row-by-row, column-by-column. The problem is that most of the elements in a row or column depend on other elements in the same row (or column) and also on the previous row (or column). This means the row (or column) cannot be computed in parallel. Another challenge is to do with the number of elements to be computed by each processor in each step. This would lead to expensive computation.

As shown in the Fig. 1 of the previous example, the SM consists of R rows and C columns. In this work, firstly we divide the SM into two thread or processor; one thread for computing the columns, and another thread for computing the rows. We used multithread program with two processors, one processor for each thread. We named them as "rows processor" and "columns processor". Secondly, we initialize the matrix by zeros and initialize the value of the variables "next column" and "next row" by one's.

When the computation starts at position (1,1), the "row processor" starts computing for all cells from row 1 and the "column processor" starts its computation for all cell from column 1. At the same time the value for "next row" and "next column" are increase to 2. These two variables are the control variables to determine which column and row to be computed next by the processor. In the third step, "row processor" and column processor" computes all cells from the second row and second column respectively started from position (2,2) and increase the value of "next row" and "next column" to 3. The value for these variables will increase in the same way for all the next steps.

To compute (I,J) cell, we will pass the row I and row I-1 to the "rows processor", and pass the column J and column J-1 to the "columns processor". This will decrease the memory space. Sometimes the protein sequences are too long thus to make it easy to return the value, we will pass the rows/columns as block and return its score value one by one.

IV. Implementation

We implemented this system using multithreading approach to improve the parallelization of the DPA. We used Swiss-Prot, as sequence database. As described in our design as in section III, the SM will be parallelized into two threads, one for rows and the other for columns. We implemented this approach on many PCs by using RMI architecture. RMI provides a mechanism for the server and client to communicate and pass information back and forth. We used Java language, which is a powerful programming language and its support for multithreaded programming. Java provides RMI architecture, which is a standard architecture for distributed object systems. It allows a distributed, heterogeneous collection of objects to interoperate.

A. Sequential and Parallel DPA

In order to evaluate and compare the performance between the parallel DPA and sequential DPA, the sequential algorithm was implemented first by using Java programming language on a 1.4 GHz Pentium 4 PC machine with 256 MB main memory, 30GB hard disk. The PC machine runs windows XP professional operation system.

The parallel DPA was implemented on a shared distributed architecture using three PCs cluster. A 10-100 MB LAN connects the PC clusters. The setting and configuration of the cluster were done using RMI. After the program stated up for parallel DPA, one node will be chosen as master node and the rest will serve as slaves. The master is responsible for determining the length of the two protein sequences intended for comparison. The master node creates thread for each slave node and distributes the database to the all slave nodes by remote method. Slave nodes calculate the scoring value and master node combines the protein sequence ID, alignment, and scoring value and return back to the slave nodes in one file. The slave nodes calculate the similarity matrix for the pair wise sequence comparison for protein sequence similarity by request to the "rows processor" and "columns processor" at the same time.

V. Experimental Results

The parallel DPA have tested 500 protein sequences using 2 and 3 processors. The results were compared with the results of the sequential DPA. A ratio of 47.02%, 64.28% of execution time for parallel DPA is reduced when 2 and 3 processors were used respectively.

Fig. 8 shows the reduction of execution time when the number of the processors increased. The parallel DPA achieved the reduction of the execution time because of the sequences were distributed over all processors. Each processor worked on its sequences to construct the similarity matrix and calculate the scoring value. The ratio of the execution time reduction is 32.60% when we used 3 processors compared to the 2 processors. The ratio of the execution time decreased to 14.43% if we used 3 processors compared to 4 processors. The reason for this small ratio of execution time reduction between 3 and 4 processors was the communication overhead, which rose when the number of the processors increased. The communication overhead was due to the increasing of the number of the messages, data movement between processors and the request/callback between master/slaves nodes.

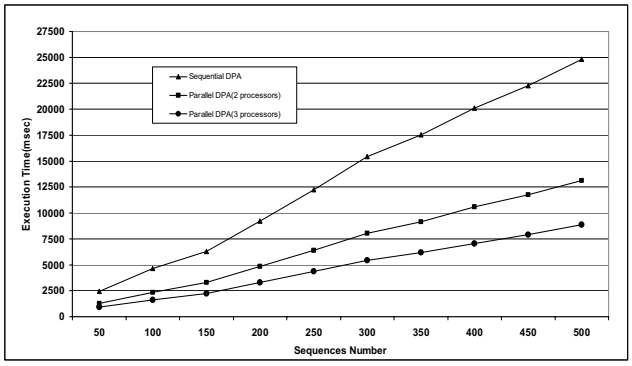

Fig. 8: Parallel and Sequential DPA execution time

VI. Conclusion and Future Research

In this paper we present a new approach parallel DPA for sequence alignments using RMI technique in a cluster system. In our approach, using RMI, more than one sequences will be read at the same time and distributed to every processor and calculate the protein sequences similarity parallely. Our parallel DPA significantly reduced the processing time that the existing DPA required. Further work , we try to implement this parallel DPA in other platform such as MPI to reduce the communication overhead among the PCs.

References

[1] T. F. Smith and M. S. Waterman. Identification of common molecular subsequences. J. Mol. Biol., 147:195–197, 1981.
[2] S. Needleman and C. Wunsch. A general method applicable to the search for similarities in the amino acid sequences of two proteins. J. Mol. Biol., 48:444–453, 1970.
[3] Pearson, W.R., Wood, T., Zhang, Z. and Miller, W. Comparison of DNA sequences with protein sequences. Genomics, 1997
[4] William R. Pearson, Protein sequence comparison and protein evolution, University of Virginia, Charlotesville, VA 22908, USA, 2000
[5] Pearson, W. R. & Lipman, D. J. Improved tools for biological sequence comparison. Proc. Natl. Acad. Sci. USA. 1988.
[6] Hobohm, U. and Sander, C. A sequence property approach to searching protein databases. 1995.
[7] Gibbs, A.J. and McIntyre, G.A. The diagram, a method for comparing sequences. Its use with amino acid and nucleotide sequences. Eur. J. Biochem, 16, 1-11.1970.
[8] Pearson, W.R. Rapid and sensitive sequence comparison with FASTP and FASTA. Methods Enzymol, 183:63-98, 63-98.1990.
[9] Sellers P.H. On the theory and computation of evolutionary distances. SIAM J.Appl. Math, 26, 787-793. 1974.
[10] Goad, W.B. and Kanehisa, M.I. Pattern recognition in nucleic acid sequences. I. A general method for finding local homologies and symmetries. Nucleic Acids. Res., 10, 247-263.1982.Waterman, M.S., Smith, T.F. and Beyer, W.A. Some biological sequence metrics. Adv. Appl. Math, 20, 367-387. 1976.
[11] Waterman, M.S., Smith, T.F. and Beyer, W.A. Some biological sequence metrics. Adv. Appl. Math, 20, 367-387. 1976.
[12] Gotoh, O. An improved algorithm for matching biological sequences. J. Mol. Biol., 162, 705–708. 1982.
[13] http://www.sbc.su.se/~per/molbioinfo2001/dynprog/dynamic.html.