Parallel Implementation of Smith Waterman Sequence Alignment Algorithm

Han Su Yin Nyunt; Thinn Thu Naing
University of Computer Studies, Yangon
E-mail: hansuyinnyunt@gmail.com

Abstract

Local sequence alignment is widely used to discover structural and hence, functional similarities between biological sequences. Sequence database alignment is among the most important and challenging tasks in bioinformatics. This paper presents a parallel algorithm that finds all occurrences of a pattern string in a subject string in O(log n) time, where n is the length of the subject string. The number of processors employed is of the order of the product of the two string lengths. It also presents advanced computer architectures that utilize parallelism via multiple processing units. While parallel computing, in the form of internally linked processors, was the main form of parallelism, advances in computer networks has created a new type of parallelism in the form of networked autonomous computers. . The right choice of sequence alignment algorithm is that of Smith-Waterman. To get high quality results in a short time is to use parallel processing.

1. Introduction

A massive volume of biological sequence data is available in over 36 different databases worldwide, including the sequence data generated. These databases, which also contain biological and bibliographical information, are growing at an exponential rate. The computational demands needed to explore and analyze the data contained in these databases is quickly becoming a great concern. To meet these demands, we must use high performance computing systems, such as parallel computers and distributed networks of workstations.

Searching for similarities in protein and DNA databases has become a routine procedure in Molecular Biology. The Smith-Waterman algorithm is based on a dynamic programming approach that explores all the possible alignments between two sequences; as a result it returns the optimal local alignment. The Smith-Waterman algorithm guarantees the maximal sensitivity for local sequence alignments. It should be further considered that biological databases are growing at a very fast exponential rate, which is greater than the rate of improvement of microprocessors. This trend results in longer time. For the above reasons, many widespread solutions running on common microprocessors now use some heuristic approaches to reduce the computational cost of sequence alignment. Most widely used is running the alignment processes in parallel.

This paper presents and analyzes the parallel processing of DNA sequence alignment with single processing. The alignment of two DNA or protein sequences is used to detect functional similarities. High sequence similarity implies structural and functional similarity. Smith-Waterman alignment algorithm will be used to find the alignments. It is an exact alignment algorithm that finds the highest scoring alignment possible between two DNA sequences. The organization of this paper is as follows. Section 2 presents the related work of sequence alignment process. In section 3, DNA structure and how DNA is made of are illustrated.

2. Related Work

A number of efforts have also been made to obtain faster implementations of the Smith-Waterman algorithm on commodity hardware. Farrar [1] exploits Intel SSE2, which is the multimedia extension of the CPU. Its implementation is much faster than SSEARCH [3] (a quasi-standard implementation of Smith-Waterman).

An attempt to implement Smith-Waterman on a GPU was done by W. Liu et al. (2006) [2]. Their solution relies on OpenGL that has some intrinsic limits as it is based on the graphics pipeline. Thus, a conversion of the problem to the graphical domain is needed, as well as a reverse procedure to convert

back the results. Although that approach is faster than SSEARCH but slower than Farrar[1].

3. Smith Waterman Algorithm

The Smith-Waterman algorithm is designed to find the optimal local alignment between two sequences. It was proposed by Smith and Waterman and enhanced by Gotoh [8]. The alignment of two sequences is based on the computation of an alignment matrix. The number of its columns and rows is given by the number of the residues in the query and database sequences respectively. The computation is based on a substitution matrix and on a gap-penalty function.

Consider two strings S1 and S2 of length l1 and l2. To identify common subsequences, the SW algorithm computes the similarity H(i, j) of two sequences ending at position i and j of the two sequences S1 and S2. The computation of J(i, j) for $1 \le i \le l1$, $1 \le j \le i2$ is given by following sequence is given by the following recurrences:

$$H(i,j) = \max\{0, E(i,j), F(i,j), H(i-1,j-1) + Sbt(S1i, S2i)\}$$
$$E(i,j) = \max\{H(i,j-1) - \alpha, E(i,j-1) - \beta\}$$
$$F(i,j) = \max\{H(i-1,j) - \alpha, E(i-1,j) - \beta\}$$

Figure 1 is the example of the SW algorithm to compute the local alignment between two DNA sequences ATCTCGTATGATG and GTCTATCAC. The matrix H{i, j} is shown for the computation with gap costs $\alpha=1$ and $\beta=1$, and a substitution cost of + 2 if the characters are identical and -1 otherwise. From the highest score (+ 10 in the example), a trace back procedure delivers the corresponding alignment (shaded cells), the two subsequences TCGTATGA and TCTATCA.

	-	A	T	C	T	C	G	T	A	T	G	A	T	G
-	0	0	0	0	0	0	0	0	0	0	0	0	0	0
G	0	0	0	0	0	0	2	1	0	0	2	1	0	2
T	0	0	2	1	2	1	1	4	3	2	1	1	3	2
C	0	0	1	4	3	4	3	3	3	2	1	0	2	2
T	0	0	2	3	6	5	4	5	4	5	4	3	2	1
A	0	2	2	2	5	5	4	4	7	6	5	6	5	4
T	0	1	4	3	4	4	4	6	5	9	8	7	8	7
C	0	0	3	6	5	6	5	5	5	8	8	7	7	7
A	0	2	2	5	5	5	5	4	7	7	7	10	9	8
C	0	1	1	4	4	7	6	5	6	6	6	9	9	8

Figure 1: Example of Two Sequence Alignment

Process of Smith Waterman Algorithm

1. Assigns a score to each pair of bases – Uses similarity scores only
   - Uses positive scores for related residues
   - Uses negative scores for substitutions and gaps
2. Initializes edges of the matrix with zeros
3. As the scores are summed in the matrix, any score below 0 is recorded as 0.
4. Begins the trace back at the maximum value found anywhere in the matrix
5. Continues until the score falls to 0.

4. Parallel Processing

Parallel computer is a computer with many processing units or processors. Given a problem to be solved, it is broken into a number of sub problems. Having multiple processors working simultaneously on a problem, the processors work collectively to solve the single problem usually for the maximum performance. All of these sub problems are now solved simultaneously, each on a different processor. The results are then combined to produce an answer to the original problem.

There are four classes of computers for processing

1. Single Instruction stream, Single Data Stream (SISD) – a computer in this class consists of a single processing unit receiving a single stream of instructions that operate on a single stream of data.
2. Multiple Instruction stream, Single Data Stream (MISD) – N processors each with its own control unit share a common memory unit where data reside. There are N streams of instructions and one stream of data. At each step, one datum received from memory is operated upon by all the processors simultaneously. Parallelism is achieved by letting the processors do different things at the same time on the same datum.
3. Single Instruction stream, Multiple Data Stream (SIMD) – a parallel computer consists of N identical processors. Each of N processors possesses its own local memory where it can store both programs and data. All processors operate under the control of a single instruction stream issued by a central control unit.

Multiple Instruction stream, Multiple Data Stream (MIMD) – In this class of computer, the

computation classifies parallel computers according to whether the instruction and / or the data streams are duplicated. There are N processors, N streams of instructions, and N streams of data. Figure 1 presents the parallel processing architecture of SIMD model, where each processor has its own data stream but performs task for single instruction. SIMD architecture is mainly focused in this paper.

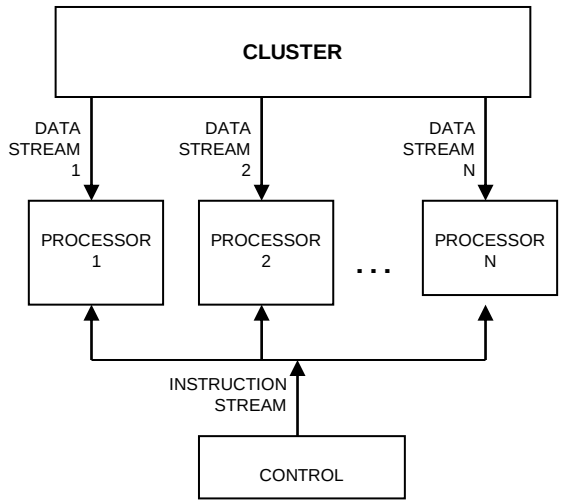


Figure 2: SIMD Parallel Processing

5. DNA

Deoxyribonucleic acid (DNA) is a nucleic acid that contains the genetic instructions used in the development and functioning of all known living organisms and some viruses.

Deoxyribonucleic acid, or DNA, carries the hereditary information. DNA and proteins make up the chromosomes of cells. DNA is made up of molecules of the sugar deoxyribose, phosphate groups, and nitrogen bases. The basic unit of DNA, the nucleotide, is made up of one of each. A molecule of DNA may contain as many as 200,000 nucleotides. The nucleotides make up two chains that are linked and twisted around one another in the form of a double helix. The rungs of the DNA ladder consist of pairs of nitrogen bases. There are two kinds of nitrogen bases: purines and pyrimidines. The purines have a two-ringed structure; they are adenine (A) and guanine (G). The pyrimidines have a one-ring structure; they are cytosine (C) and thymine (T).

The main role of DNA molecules is the long-term storage of information. DNA is often compared to a set of blueprints or a recipe, or a code, since it contains the instructions needed to construct other components of cells, such as proteins and RNA molecules. The DNA segments that carry this genetic information are called genes, but other DNA sequences have structural purposes, or are involved in regulating the use of this genetic information.

There are different DNA sequence formats, such as FASTA, BLAST, etc. In the FASTA format, sequence is starts with “>” and sequence number. General sequence structure for all formats is as shown below.

ACAAGATGCCATTGTCCCCCGGCCTCCTGCT
GCTGCTGCTCTCCGGGGCCACGGCCACCGCT
GCCCTGCC
CCTGGAGGGTGGCCCCACCGGCCGAGACAG
CGAGCATATGCAGGAAGCGGCAGGAATAAG
GAAAAGCAGC
TCCTGACTTTCCTCGCTTGGTGGTTTGAGTGG
ACCTCCCAGGCCAGTGCCGGGCCCCTCATAG
GAGAGG
AAGCTCGGGAGGTGGCCAGGCGGCAGGAAG
GCGCACCCCCCCAGCAATCCGCGCGCCGGGA
CAGAATGCCTGCAGGAACTTCTTCTGGAAGA
CCTTCTCCTCCTGCAAATAAAACCTCACCCA
TGAATGCTCACGCAAG
TTTAATTACAGACCTGAA

5.1 Sequence Alignment for DNA Classification

DNA that have a significant biological relationship to one another often share only isolated regions of sequence similarity. For identifying relationships of this nature, the ability to find local regions of optimal similarity is advantageous over global alignments that optimize the overall alignment of two entire sequences. Having sequenced a particular protein, it is of interest to compare it with previously characterized sequences. Sequence alignment of DNA sequences is used to analyze the data and interpret the results in a biologically meaningful manner.

6. Proposed System

This paper presents the parallel processing approach for the DNA sequence alignment. There are n processors processing for the input DNA sequence in this system. Smith-Waterman algorithm is used to find the matches scores of the sequences. It is implemented as the distributed system. When user request arrives at a processor, it splits jobs into sub jobs and sends requested to other processors. Then request is processed in parallel and results are replied to sender processor. Results are compiled at the sender processor and best result is sent back to user. Figure 3 is the proposed system architecture for the parallel processing of DNA sequence alignment.

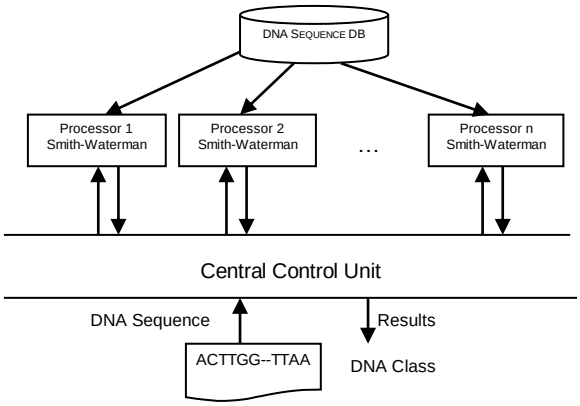


Figure 3: Proposed System Design

7. System Implementation

This system is implemented using Java programming language. Processors are implemented in the distributed computers. Remote Method Invocation is used to send and receive results from distributed machines.

The main process of the system is finding the matched DNA sequence in a set of DNA databases. Smith-Waterman algorithm is used to find the similarity of input DNA sequence and DNA databases. Since DNA sequences are large, and there are a lot of DNA sequence database, processing time of the DNA sequence alignment is the major concern in the DNA sequence alignment. Parallel processing architecture is used to optimize the processing time.

Each processor stores the required information (computer name, remote object) of other processors. When a job request arrives, it locates the remote processors for distributed parallel processing. After processing parallel jobs, the main processor combines all results and displays outputs. The main process flow and implementation of the system is shown in Figure 4, where when a job arrives, tasks are separated according to number of processors assigned. Then each task is redirected to corresponding remote processors. Tasks are processed at the remote processors and remote processors return the results back to the main system. It collects, finalizes results and returns the final result back to the user. In this system, it is assumed that Smith Waterman algorithm is implemented in each processor and they have their own sequence database. Whenever there is update in the sequence database, it is distributed to other remote processor. Therefore, all remote processors maintain the consistency state of sequence database, i.e., having the same sequence database all time.

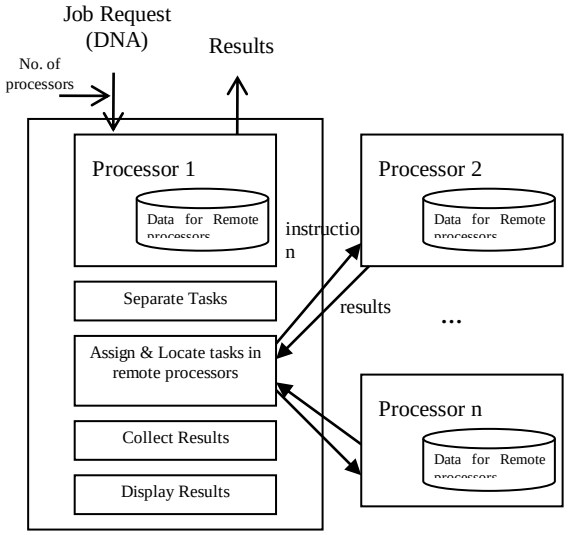


Figure 4: Implementation of the System

8. Experimental Result

We have run this program with four computers, implementing four distributed processors. Computers are desktop computers with Processor Intel(R) Pentium(R) Dual CPU 2.20GHz, Memory 2 GB of RAM. The type of network used in this system is P2P with no domain server.

For the same request, we have tested with one processor, two processors, three processors and four processors. Running example is shown in Figure 5. Figure 5 (a) is the input DNA, Figure 5 (b) presents the types of DNA available in this system, Figure 5 (c) shows the processing status of current request, and Figure 5 (d) is the results for different classes. This example scenario is run with number of processors = 4 and Similarity Threshold = 0.75.

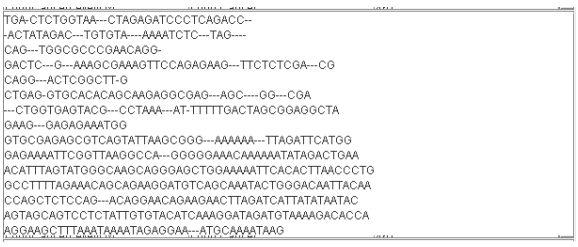
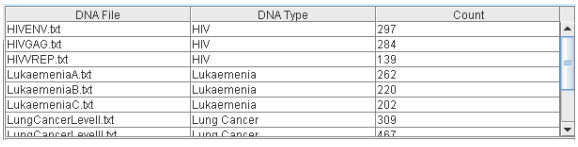


Figure 5 (a): Input DNA



DNA File	DNA Type	Count
HIVENV.txt	HIV	297
HIVGAG.txt	HIV	284
HIVVREP.txt	HIV	139
LukaemeniaA.txt	Lukaemenia	262
LukaemeniaB.txt	Lukaemenia	220
LukaemeniaC.txt	Lukaemenia	202
LungCancerLevelI.txt	Lung Cancer	309
LungCancerLevelII.txt	Lung Cancer	467

Figure 5 (b): DNA Types available in this system

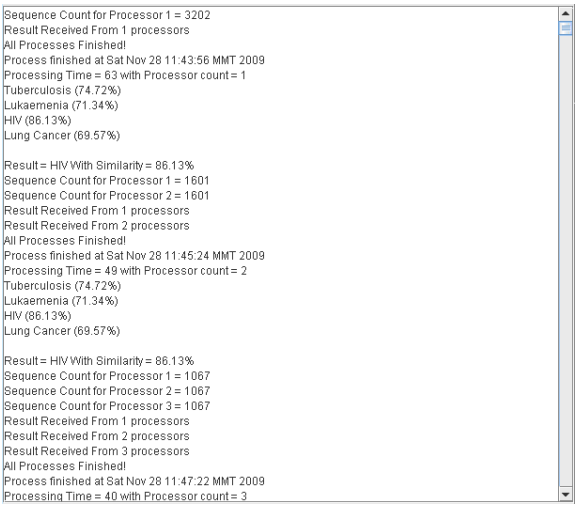


Figure 5 (c): Processing status of requested task at a processor

No.	Type	Max Sim	Length	Sim Score
1	Tuberculosis	464	621	74.72%
2	Lukaemenia	443	621	71.34%
3	HIV	621	721	86.13%
4	Lung Cancer	432	621	69.57%

Figure 5 (d): Similarity Score Results

Processing Time comparison (Performance analysis) for different processors is shown in Figure 6.

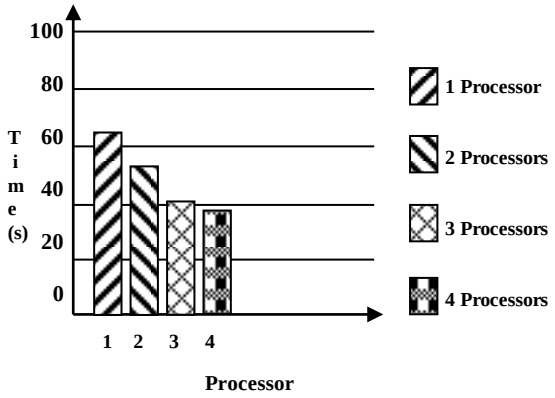


Figure 6: Comparison of Processing Times in different number of processors.

Since this system uses data in small amount, the performance shows no significant changes for different processors for taking costs in initializing network services. But in the reality, DNA sequences are relatively long and network initialization time becomes a small factor of processing time. Therefore this system performs well for long DNA sequences with a large number of DNA sequences in the sequence database.

When compared to other sequence alignment algorithms BLAST and single processor Smith Waterman Algorithm, this parallel processing of Smith Waterman algorithm outperforms over above two algorithms.

9. Conclusion

Parallelism is needed to keep pace with the ever-growing demands in sequence comparison. It reduces the running time from the sequential to the parallel processing. This paper also presents performance analysis of these two different approaches with different processor counts. The analysis results show that the performance of sequence alignment process is significantly higher in longer sequences for different number of processors. There is no significant difference for short DNA sequences in processing with different number of processors since network initialization cost takes the great factor of overall processing time.

10. References

[1] Farrar M: "Striped Smith-Waterman speeds database searches six times over other SIMD implementations", Bioinformatics 2007, 23(2):156-161.

[2] Liu W, Schmidt B, Voss G, Schroeder A, Muller-Wittig W: "Bio-Sequence Database Scanning On GPU. In Proceeding of the 20th IEEE International Parallel & Distributed Processing Symposium", IPDSP 2006 (HICOMB Workshop Rhode Island, Greece; 2006.

[3] Pearson W: "Searching protein sequence libraries: comparison of the sensitivity and selectivity of the Smith-Waterman and FASTA algorithms", Genomics 1991, 11:635-650.

[4] Hesham El-Rewini and Mostafa ABD-El-Barr, "Advanced Computer Architecture and Parallel Processing", Wiley Series on Parallel and Distributed Computing.

[5] Ronges T. and Seeberg E, "Six-fold speed-up of Smith-Waterman sequence database searches using parallel processing on common microprocessors", Oxford University Press 2000.

[6] Yang B. H. W, “A Parallel Implementation of Smith-Waterman Sequence Comparison Algorithm”, December 6, 2006.

[7] Slim G.Akl, “The Design And Analysis of Parallel Algorithms ”, ISBN- 0-13-200056-3, 1989, Pages 1-38

[8] http://en.wikipedia.org/wiki/Smith-Waterman_algorithm

[9] http://searchlauncher.bcm.tmc.edu/help/SmithWaterman.html