Comparison of Parallelized Sequence Alignment Algorithms

CSE4001 – Parallel and Distributed Computing

PROJECT BASED COMPONENT REPORT

By

SHASHWAT NEGI

(17BCE0322)

YASH BHOJWANI

(17BCE0614)

RACHANA REDDY

(17BCE0278)

School of Computer Science and Engineering

VIT

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

March 2019

DECLARATION

I hereby declare that the report entitled "**Comparison of Parallelized Sequence Alignment Algorithms**" submitted by me, for the CSE4001 Parallel and Distributed Computing (EPJ) to VIT is a record of bonafide work carried out by me under the supervision of Dr.Narayanan Prasanth.

I further declare that the work reported in this report has not been submitted and will not be submitted, either in part or in full, for any other courses in this institute or any other institute or university.

Place : Vellore

Date : 5/11/19

Signature of the Candidate
Shashwat Negi

Signature of the Candidate
Yash Bhojwani

Signature of the Candidate
Rachana Reddy

CONTENTS

List of Figures

List of Tables

List of Abbreviations

S-W	Smith-Watermann
N-W	Needleman-Wunsch
OMP	Open Multi-Processing
FPGA	Field Programmable Gate Array
MPI	Message Passing Interface
DNA	Deoxyribo Nucleic Acid

ABSTRACT

Parallel and Distributed computing is the future of technology. All products and their fundamental concepts are being shifted to a parallel computing model. Everybody would agree that serial computing is easy to implement and use, but simply not efficient enough for industry level purposes. Due to this reason, day by day higher number of industries are providing and using cloud solutions which work on the basis on parallel and distributed computing. For instance, Amazon's AWS or Google's
Google Cloud platform are becoming the center for development, may it be in the field of web development, or in the field of data analytics. To cope up with the fast-paced improvement in technology, one must also become familiarized with this domain, and hence this project. Gene sequencing problem is one of the major issues for researchers regarding optimized system models that could help optimum processing and efficiency without introduction overheads in terms of memory and time. Bioinformatics and computational biology is a latest multidisciplinary field which explains many aspects of the fields of computer science, while computational biology harnesses computational approach and technologies to respond biological questions conveniently.
The libraries used for the same are: <stdio.h>,<stdlib.h>,<math.h>,<omp.h>,<time.h>.
We would be learning mainly how to detect sequences in proteins and nucleic acids and how to detect them using parallel computing using multiple threads.
We use this inspiration of our project to create something on a smaller scale, but with a large scope.

1. INTRODUCTION

1.1 OBJECTIVE

To learn and use OpenMP and its related packages:
To learn and use Parallel processing and compare its efficiency with serial execution by implementing the above mentioned algorithm.
This would be done in parallel and in serial and a comparison between them would be done.
To work in a team and understand the advantages of teamwork by assigning roles for each member and deadlines for the project, which would help us get suited to a work-based environment and benefit us in the future.
To improve our report writing and presentation skills through the frequent Reviews conducted to check on our progress.

1.2 MOTIVATION

As time passes, the world is becoming more and more oriented towards Parallel Computing. Many of the tasks that were once carried out sequentially are now being carried out in parallel so as to use resources more efficiently and get faster results. Genome is an emerging field, constantly presenting many new challenges to researchers in both biological and computational aspect of application. Research is being done in the Biology with the application Smith–Waterman algorithm it is possible to process and understand nucleic acid/ protein sequences. Sequence comparison is a very essential and important operation. They detect similar or identical parts between two sequences called the query sequence and the reference sequence. The global and local alignments are the most prevalent kinds of sequence alignment. In global alignment, we find the superior counterpart between parts of the sequences. On the other hand, local alignment algorithms try to match parts of sequences and not the entirety of them. Local alignment is faster than global alignment, due to the lack of need to align the entire sequences. In our project, we would be implementing the Smith-Waterman (Local sequence alignment) Algorithm for randomly generated nucleotide sequences in a serial and parallel manner for comparison and analysis. As common sense suggests, the parallel implementation should execute and provide the same result as the serial implementation but in a lesser amount of time.

2. LITERATURE SURVEY

High Performance Streaming Smith-Waterman Implementation with Implicit Synchronization on Intel FPGA using OpenCL

In this paper, Ernst and Zaid have presented OpenCL-based FPGA Smith-Waterman implementation that employs two key techniques to greatly improve the utilization of its underlying systolic array architecture. By eliminating centralized control and through the use of Query Buffers, an arbitrary number of alignments can be in flight at the same time, resulting in utilization close to theoretical maximum performance. In parallel computer architectures, a systolic array is a homogeneous network of tightly coupled data processing units called cells or nodes. Field-Programmable Gate Arrays (or FPGAs), with their flexible and reprogrammable substrate, are a natural fit for a computationally intensive algorithm such as the Smith-Waterman algorithm. The Query Buffer contains for each Processing Element a separate queue with query symbols for the alignments it needs to process. Whenever the Processing Element encounters the new read token, it checks against the query length to verify if it is active during this alignment; if so, it reads the next query symbol from its queue. Only the Input Parser and Output Parser communicate with the on-board DDR memory.

Development of DNA Sequencing Accelerator Based on Smith Waterman Algorithm with Heuristic Divide and Conquer Technique for FPGA Implementation

In this paper, a new approach has been introduced to reduce the complexity of the Smith Waterman algorithm for FPGA implementation. The technique for the fastest comparison of the two DNA sequencing using Verilog on the Xilinx ISE 7.1. In this paper, it was proved that Smith Waterman algorithm based on divide and conquers technique gives better performance than existing technique.The Smith Waterman algorithm is the complex and heuristic algorithm for the DNA sequencing. Divide and Conquer technique helps to reduce the complexity of the main structure. In previous works, the main structure is divided or break-up into few modules called sub-functions and at cluster based. However, the result is less sensitive. In this technique, the functions involve in Smith Waterman algorithm were break up into few sub- functions. These parts are prediction of the sequence score, previous result score and optimization of the sequencing.

A Parallel Implementation of the Smith-Waterman Algorithm for Massive Sequences Searching

In this paper, Hsien-Yu Liao, Meng-Lai Yin, Yi Cheng has presented a parallel implementation methodology of the Smith-Waterman algorithm. The power of parallelization lays on the massive comparisons. When an unknown sequence is compared with different existing sequences, each comparison is independent and can be performed independently. This observation highlights the potential of massive parallelism existing in this particular application. Using this method, it was proven that high efficiency can be achieved.

Parallel Processing Cell Score Design of Linear Gap Penalty Smith-Waterman Algorithm

In this paper, the optimize computational processing element for linear smith-waterman algorithm cell score based on the parallel computational approach is introduced. Two bits optimized comparator block were used to compare the DNA sequence characters, while the computation block was used to complete computation towards reducing the numbers of components involved. The complete architecture is designed and developed in Altera Quartus II version 13.0 and targeted to Altera Cyclone IV EP4CE115 Field Programmable Gate Array (FPGA). In realizing the important of the logic functionality of the design, the modular design approach has been adapted to this design. The simulation over the module and complete architecture were used for validating the design against the expected result. Finally, the result obtained from the study indicates that the optimized computational processing element design for linear smith-waterman algorithm cell score based on the parallel computational approach is feasible to be implemented in the DNA sequence alignment as the processing element.

Performance Improvement of the Parallel Smith Waterman Algorithm Implementation Using Hybrid MPI – Openmp Model

In this paper, a hybrid parallel model was introduced that combines both shared and distributed memory architectures to improve the performance of the Smith waterman algorithm (SW). The hybrid model uses both MPI and OpenMp as programming techniques for different memory architectures. Our improved implementation executes a parallel version of SW algorithm with a row wise computation of the alignment matrix, which mainly optimizes the memory usage. Using different parallel programming models and their corresponding implementations.

A Comparative Analysis of Smith-Waterman Based Partial Alignment

In this paper, Aruk, Ustek, and Kursun have given a comparative simulation of three Software based methods in terms of their runtimes and errors in estimated position of the start of the deletion in the query sequences. As a result, they have found that Binary Partial Align has the lowest error and very high speed. Partial alignment aims at finding the best splitting of a query sequence into the former and latter parts such that their matching scores with (different parts of) the reference sequence are mutually maximized. In this paper, we give a more detailed comparison of the classical SW, IncrementalPartialAlign, and BinaryPartialAlign. BinaryPartialAlign has better EDPE than the other two and is faster than IncrementalPartialAlign, making it the best option among the three.

A Novel Structure of the Smith-Waterman Algorithm for Efficient Sequence Alignment

In this paper, a new architecture for SW algorithm has been presented for the implementation of the matrix fill-up stage in the SW algorithm. The matrix fill-up stage is one of the most time-consuming operations in the SW algorithm. The conventional implementation of this phase takes four cycles for computation of each element in the matrix. The newly proposed design reduces this latency to three cycles. However, the proposed design takes one overhead cycle at the start of the processing for ensuring the 100% accuracy. With the newly proposed architecture, the SW algorithm achieves up to 25% speedup.

An Efficient and High Performance Linear Recursive Variable Expansion Implementation of the Smith-Waterman Algorithm

In this paper, an efficient and high performance linear recursive variable expansion (RVE) implementation of the Smith-Waterman (S-W) algorithm and compare it with a traditional linear systolic array implementation. An efficient and high performance implementation of the S-W algorithm based on the linear RVE approach and compared it with a traditional linear systolic array implementation. The linear RVE implementation is efficient in terms of hardware utilization (both slices and IOBs) and high performance in terms of time consumption (latency). The results demonstrate that the linear RVE implementation is upto 2.33 times faster than a traditional linear systolic array implementation at the cost of utilizing 2 times more resources. The results demonstrate that the linear RVE implementation performs up to 2.33 times better than the traditional linear systolic array implementation, at the cost of utilizing 2 times more resources.

3. TECHNICAL SPECIFICATION

Package Requirements:

- **<stdio.h>** The C programming language provides many standard library functions for file input and output. These functions make up the bulk of the C standard library
- **<stdlib.h>** stdlib.h is the header of the general purpose standard library of C programming language which includes functions involving memory allocation, process control, conversions and others. It is compatible with C++ and is known as cstdlib in C++. The name "stdlib" stands for "standard library"
- **<math.h>** The math.h header defines various mathematical functions and one macro. All the functions available in this library take double as an argument and return double as the result.
- **<omp.h>** It is a library that allows memory multiprocessing programming in C
- **<time.h>** In C programming language time.h (used as ctime in C++) is a header file defined in the C Standard Library that contains time and date function declarations to provide standardized access to time/date manipulation and formatting.

Major Algorithm Explanation:

Smith-Waterman Algorithm

Smith-Waterman algorithm calculates the local alignment of two sequences. It guarantees to find out the best possible local alignment taking into account the specified scoring system. This includes a substitution matrix and a gap-scoring method. Scores consider match, mismatch and substitution. To measure the comparison between two sequences, a score be calculated as follows:

Given an alignment between sequences S0 and S1, the following values must be assigned, for each column:

Procedure:

- ma = (+5)
- mi = (-3)
- G = (-4)

[Match] [Mismatch] [Gap]

- Initialization of the matrix and consider two sequences A and B.
- Matrix filling with the suitable scores. The two sequences are set in a matrix form by means of A+1 columns and B+1 rows. The value in the first row and first column are set to zero.
- The second and essential step of the algorithm is filling to entire matrix. To fill each and every cell it is important to know the diagonal values.
- Trace back the sequence for an appropriate alignment is trace backing; before that the maximum score obtained in the entire matrix has to be detected for the local alignment of the sequences.
  It is likely to those maximum scores can be present in one or more than one cell, in such case there may be option of two or more alignments, and the best alignment can be obtained by scoring it.
- Tracing back begins from the position which has the highest value, pointing back with the pointers, consequently find out the possible predecessor, then go to next predecessor and continue until it reaches the score 0.

$$M_{i-1,j-1} = Max \begin{cases} + S_{i,j} \\ M_{i,j-1} + W, \\ M_{i-1,j} + W, \\ 0 \end{cases}$$

Example:

		-	C	G	T	G	A	A	T	T	C	A	G
	-	0	0	0	0	0	0	0	0	0	0	0	0
G		0	0	5	1	5	1	0	0	0	0	0	0
A		0	0	1	2	1	10	6	2	0	0	5	1
C		0	5	1	0	0	6	7	3	0	5	1	2
T		0	1	2	6	2	2	3	12	8	4	2	6
T		0	0	0	7	3	0	0	8	17	13	9	7
A		0	0	0	3	4	8	5	4	13	12	18	14
C		0	5	1	0	0	4	5	2	9	18	14	15

Fig.3 Trace back of possible Alignment

Figure 1 Track back of Alignment

Needleman-Wunsch Algorithm

In order to prove that the proposed algorithm is better, we compared this algorithm with Needleman Wunsch algorithm. The Needleman–Wunsch algorithm is an algorithm used in bioinformatics to align protein or nucleotide sequences. It was one of the first applications of dynamic programming to compare biological sequences. The algorithm was developed by Saul B. Needleman and Christian D. Wunsch and published in 1970.[1] The algorithm essentially divides a large problem into a series of smaller problems, and it uses the solutions to the smaller problems to find an optimal solution to the larger problem.[2] It is also sometimes referred to as the optimal matching algorithm and the global alignment technique.

This method aligns the pair of sequences from end to end. The entire length of the sequence is taken into account. An optimal score is calculated from the matrix formed using the maximum similarity of each character using match, mismatch and gap penalty values of the sequences. The optimal alignment is achieved by trace back of the matrix.

If x and y are strings, where length(x) = n and length(y) = m, the Needleman-Wunsch algorithm finds an optimal alignment in O(nm) time, using O(nm) space. Smith Watermanez algorithm is a clever modification of the Needleman-Wunsch Algorithm which still takes O(nm) time, but needs only O(min{n,m}) space and is much faster in practice.[2] One

application of the algorithm is finding sequence alignments of DNA or protein sequences. It is also a space-efficient way to calculate the longest common subsequence between two sets of data such as with the common diff tool.

ExistingTool:

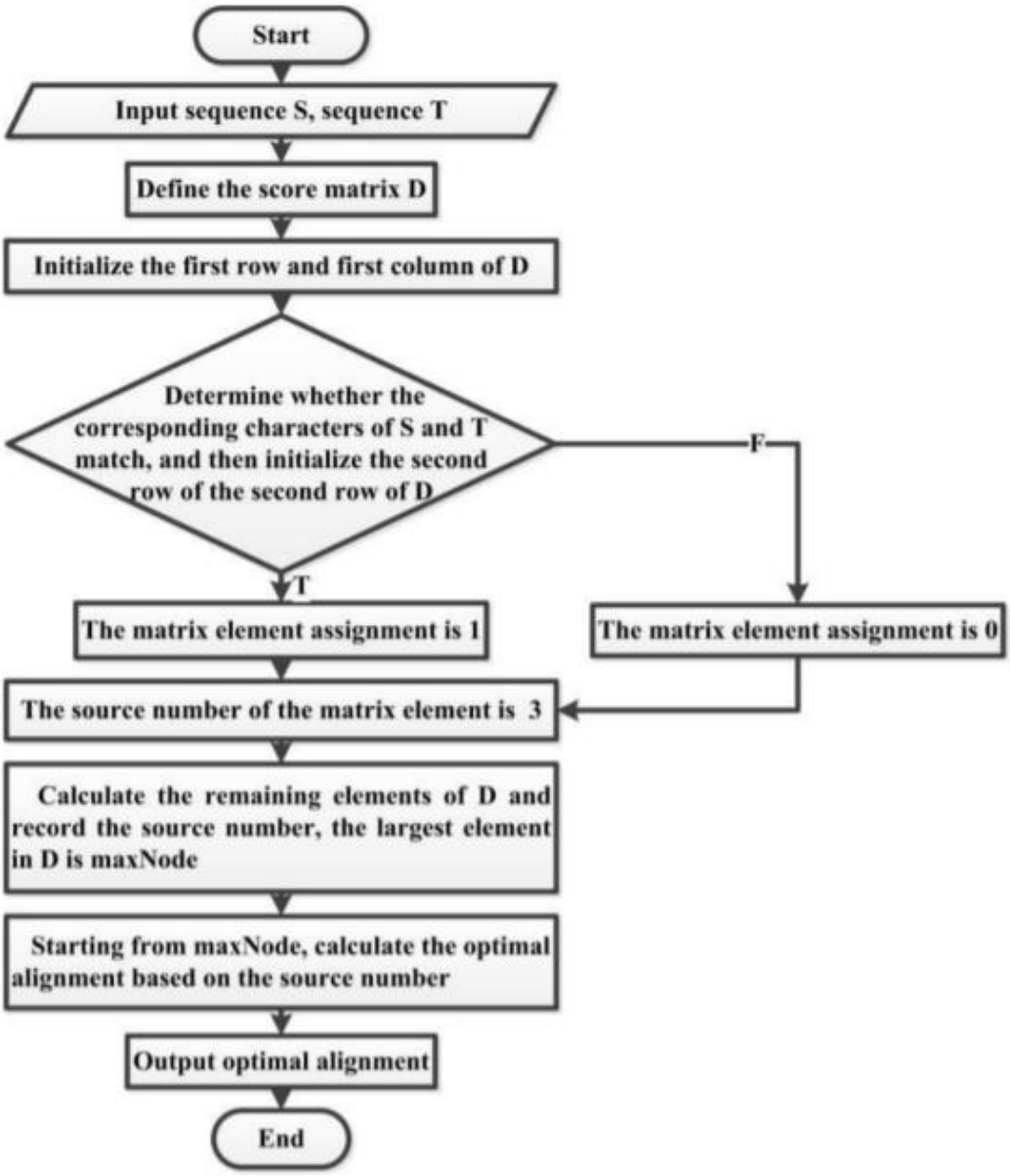
There are few existing tools which have a parallel implementation of the Smith-Waterman algorithm, but the most prominent one is Crustal W [EMBOSS WATER]. **Link:** https://www.ebi.ac.uk/Tools/psa/emboss_water/

EMBOSS Water uses the Smith-Waterman algorithm (modified for speed enhancements) to calculate the local alignment of two sequences.

We can perform the alignment for protein, DNA or RNA sequences.

4. DESIGN

The program flow is given by the following program.



5. PROPOSED SYSTEM

The problem at hand was tackled with a modular approach. Eight functions were constructed, each of which would be explained as follows:

- **nElement** – This function is used to calculate the number of elements that have been found by the Smith Waterman Algorithm. Three conditions are given: One of which is to find out if the number of elements in the diagonal are increasing, decreasing or stable.
- **calcFirstDiagElement** – This function is used to calculate the position of the maximum scored value in the matrix. This value needs to be found because the algorithm suggests that the backtracking to find the path should be started from this particular point.

- **similarityScore** – This function is used to find out the optimal order of execution based on three conditions, which are used to calculate the new values of left, upper and the diagonal elements.

  If the diagonal element > maximum element, Move diagonally upwards

  If upper element > maximum element, Move upwards

  If left element > maximum element, Move leftwards

  Every iteration, the values of maximum element is updated and inserting into the similarity and predecessor matrices.

- **matchMismatchScore** – This function is used to calculate a similarity function or the alphabet for a match or mismatch.
  If the value of the two elements are equal, it is a match, otherwise it's a mismatch.
- **Backtrack** – The purpose of this function is to modify the matrix that needs to be printed and helps us identify the path that needs to be taken to get the most optimum solution.
- **printMatrix** – It's a looped iteration implementation to display the matrix.
- **printPredecessorMatrix** - It is in this function in which we print the arrows

  depicting the path of local alignment.

  Generate – This function generates the two sequences A and B which would be locally aligned with each other. A random seed is used to ensure the reproducible nature of the output.

If x and y are strings, where length(x) = n and length(y) = m, the Needleman-Wunsch algorithm finds an optimal alignment in O(nm) time, using O(nm) space. Smith-waterman's algorithm is a clever modification of the Needleman-Wunsch Algorithm which still takes O(nm) time, but needs only O(min{n,m}) space and is much faster in practice.[2] One application of the algorithm is finding sequence alignments of DNA or protein sequences. It is also a space-efficient way to calculate the longest common subsequence between two sets of data such as with the common diff tool.

6. Working

Both the programs have been parallelized using section and for directives only. The number of threads have been set as same.

Executing SWomp.c (Smith-waterrman Algorithm)

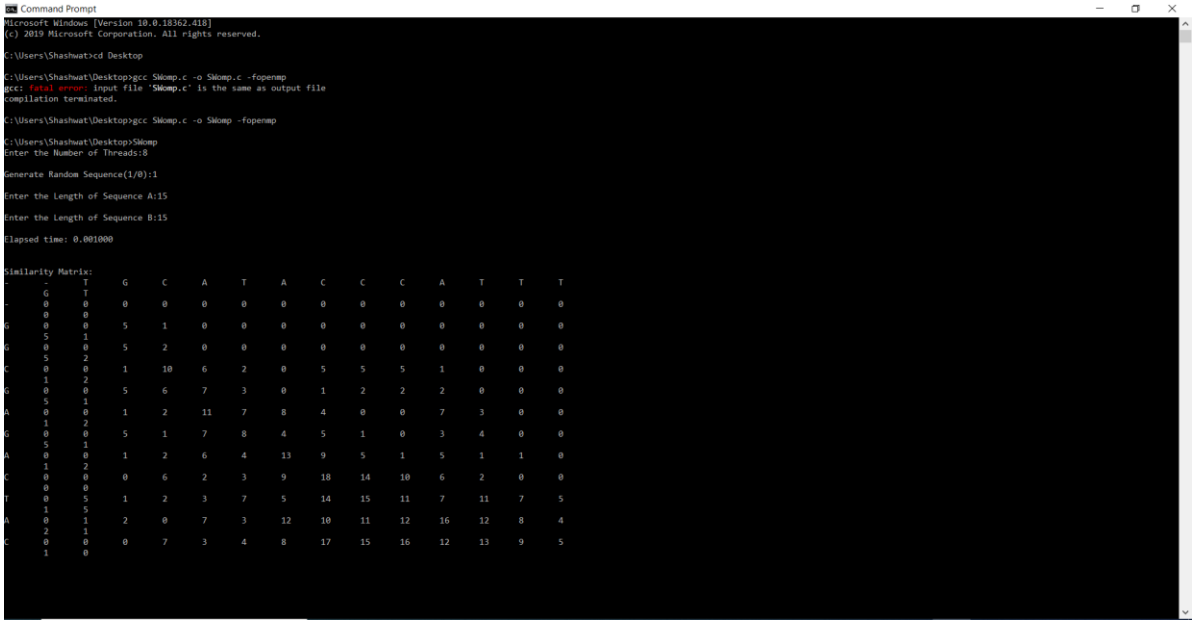


Figure 4a). Executing SW

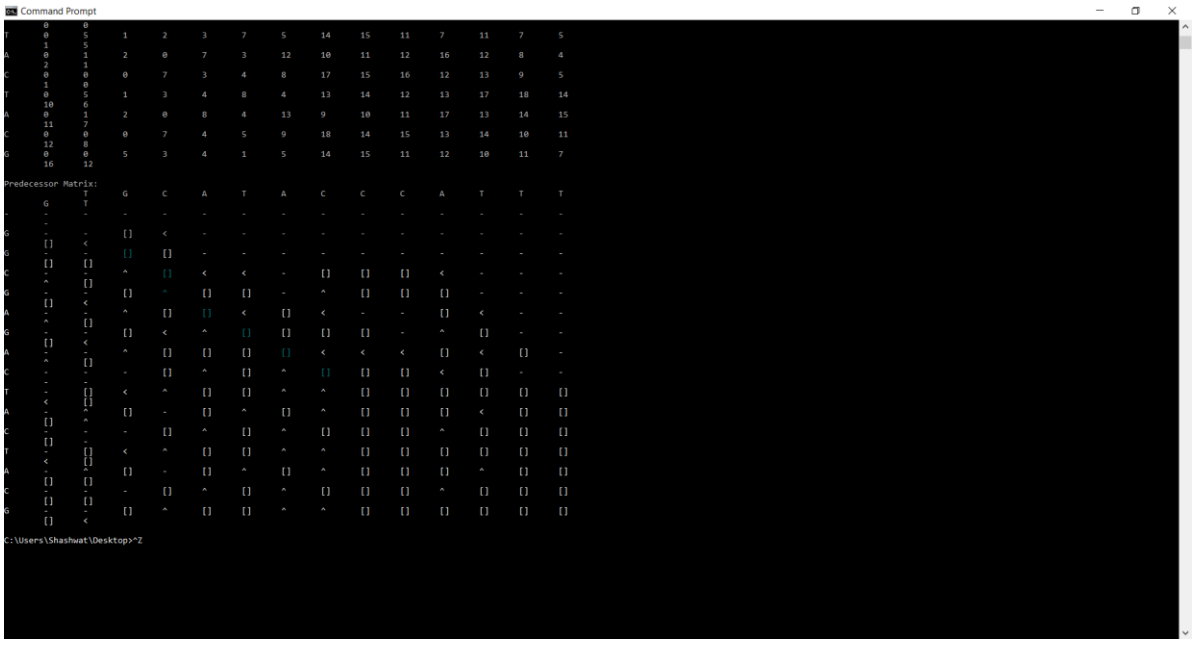


Figure 4b)Executing SW

The Predecessor Matrix helps in identifying the common sequence
Sequence 1: GTTGCATACCCATTT
Sequence 2: GGCGAGACTACTACA
The common sequence visible is " GC-A-AC"

Executing NWomp.c (Needleman-Wunsch Algorithm)

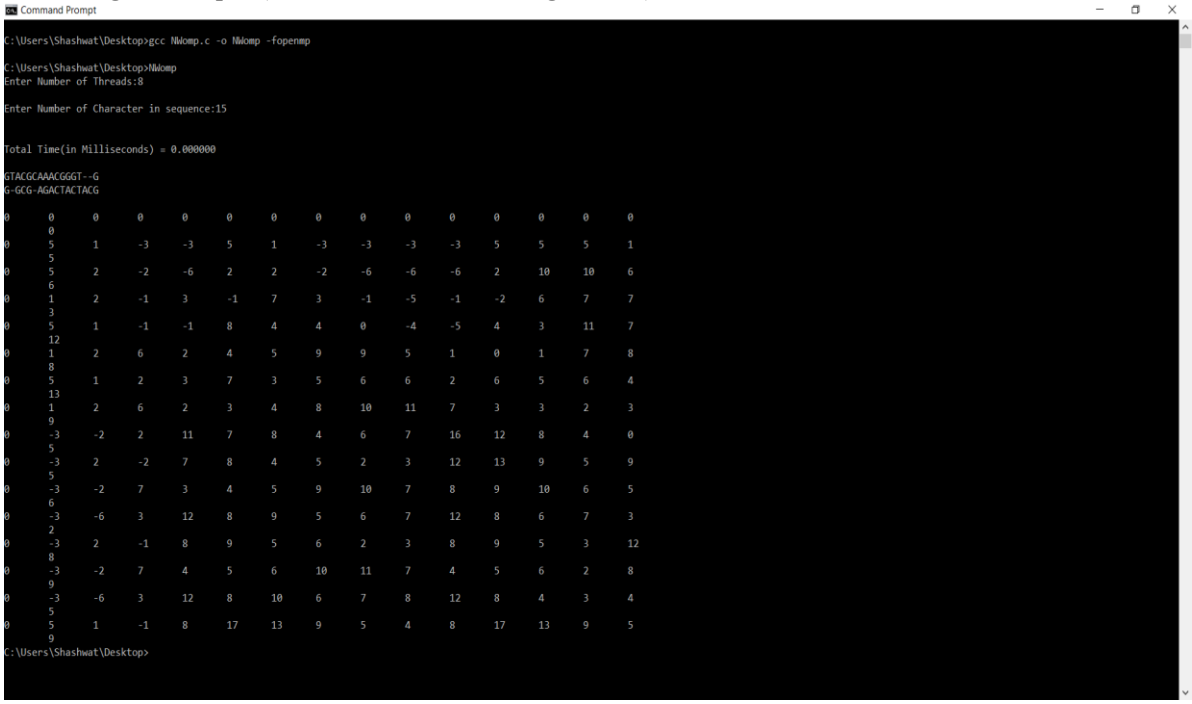


Figure 5. Executing NWomp.c

We can observe the Similar Sequence with the highest score

GTACGCAAACGGGT--G
G-GCG-AGACTACTACG

7. RESULTS AND DISCUSSION

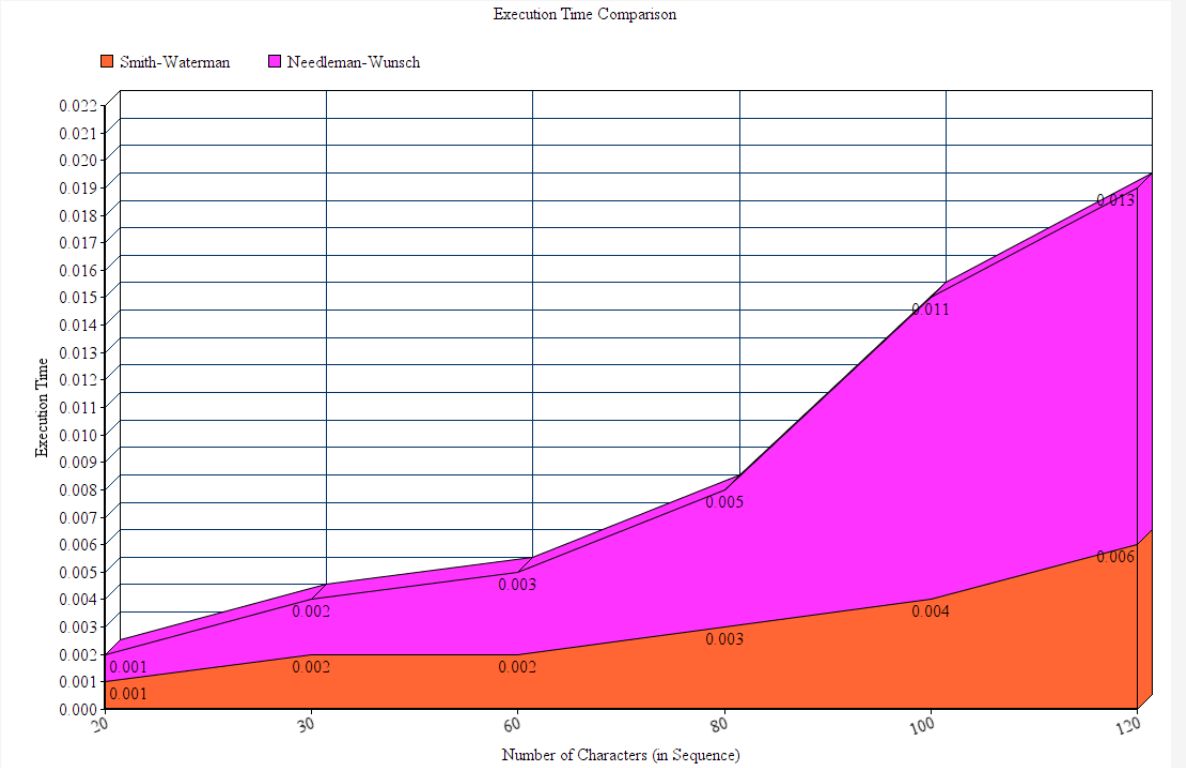


Figure 6 Graph

By considering sequences of lengths 20,30,60,80,100,120 we found the Execution time of the Parallel Implementation of Smith-Waterman and Needleman-Wunsch Algorithm

Table1 Execution Time Comparison

Sequence Length	Smith-Waterman Time	Needleman-Wunsch Time
20	0.0010	0.0010
30	0.0020	0.0020
60	0.0020	0.0030
80	0.0030	0.0050
100	0.0040	0.0110
120	0.0060	0.0130

By applying similar parallel techniques ie using “For”-“Task”-“Section” on both algorithm we get the above execution time.
The execution time so calculated is the total time to find the optimal alignment between 2 sequences without printing it.
We can conclude from the graph and the table that it is much more beneficial to implement Smith-Waterman Algorithm in parallel and Needleman-Wunsch due to reduced execution time.

8. CONCLUSION

As we can see in the above graph, for small lengths of the sequence, Smith-Waterman and Needleman- Wunsch algorithms tend to give the output in almost the same amount of time. However, as the length or the number of characters increases, the execution time for Needleman Wunsch implementation also increases exponentially, hence forming a steep graph.

The serial implementation of Smith-Waterman Algorithm is consider better in terms of performance and optimal result as it requires lot less space, considers only positive value. But real life DNA & gene sequences are very long character sequences of the order 100^s .This creates a necessary requirement for executing the sequencing algorithm in parallel.

However, the Smith-Waterman implementation remains stable with not a high rise in the execution time because of the parallel execution of the task with two threads, making the process faster than other algorithms. Hence, the proposed system will be useful when dealing with large amount of data and can be used for practical purposes..

8. REFRENECES

1. High Performance Streaming Smith-Waterman Implementation with Implicit Synchronization on Intel FPGA using OpenCL

2. Development of DNA Sequencing Accelerator Based on Smith Waterman Algorithm with Heuristic Divide and Conquer Technique for FPGA Implementation

3 A Parallel Implementation of the Smith-Waterman Algorithm for Massive Sequences Searching

4. Parallel Processing Cell Score Design of Linear Gap Penalty Smith-Waterman Algorithm

5. Performance Improvement of the Parallel Smith Waterman Algorithm Implementation Using Hybrid MPI – Openmp Model

6. An Efficient and High Performance Linear Recursive Variable Expansion Implementation of the Smith-Waterman Algorithm

7. A Novel Structure of the Smith-Waterman Algorithm for Efficient Sequence Alignment

8. Performance Improvement of the Parallel Smith Waterman Algorithm Implementation Using Hybrid MPI – Openmp Model