TRIBHUVAN UNIVERSITY

INSTITUTE OF ENGINERING

PULCHOWK CAMPUS

THESIS NO: 071/MSCS/655

Parallelization of Star Alignment Algorithm for Multiple Sequence Alignment using MapReduce Model

by

Md Hasan Ansari

A THESIS

SUBMITTED TO THE DEPARTMENT OF ELLECTRONICS AND COMPUTER ENGINEERING IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE DEGREE OF MASTER OF SCIENCE IN COMPUTER SYSTEM AND KNOWLEDGE ENGINEERING

DEPARTMENT OF ELECTRONICS AND COMPUTER ENGINNERING

LALITPUR, NEPAL

APRIL, 2017

Parallelization of Star Alignment Algorithm for Multiple Sequence Alignment using MapReduce Model

by

Md Hasan Ansari

(071/MSCS/655)

Thesis Supervisor:

Prof. Dr. Shashidhar Ram Joshi

A thesis submitted in partial fulfillment of the requirement for

the degree of Master of Science in Computer System and Knowledge Engineering

Department of Electronics and Computer Engineering

Institute of Engineering, Central Campus, Pulchowk

Lalitpur, Nepal

April, 2017

COPYRIGHT

RECOMMENDATION

The undersigned certify that they have read and recommended to the Department of Electronics and Computer Engineering for acceptance, a thesis entitled “Parallelization of Star Alignment Algorithm for Multiple Sequence Alignment using MapReduce Model”, submitted by Mr. Md Hasan Ansari in partial fulfillment of the requirement for the award of the degree of “Master of Science in Computer System and Knowledge Engineering”.

Supervisor: Prof. Dr. Shashidhar Ram Joshi
Department of Electronics and Computer Engineering

External Examiner:

DEPARTMENTAL ACCEPTANCE

The thesis entitled "Parallelization of Star Alignment Algorithm for Multiple Sequence Alignment using MapReduce Model", submitted by Md Hasan Ansari in partial fulfillment of the requirement for the award of the degree of "Master of Science in Computer System and Knowledge Engineering" has been accepted as a bonafide record of work independently carried out by him in the department.

Dr. Dibakar Raj Pant
Head of the Department
Department of Electronics and Computer Engineering,
Pulchowk Campus,
Institute of Engineering,
Tribhuvan University,
Nepal.

ACKNOWLEDGEMENT

It is with great pleasure that I would like to thank my respected supervisor Prof. Dr. Shashidhar Ram Joshi for his expert guidance and mentorship. This piece of work would not have been possible without his supervision, support, motivation and cooperation within this period. I would like to acknowledge our Head of Department Dr. Dibakar Raj Pant, Prof. Dr. Subarna Shakya and Dr. Sanjeeb Prasad Panday for their encouragement and valuable suggestions.

Also, I would like to thank our Program Coordinator (MSCSKE) Dr. Aman Shakya for his encouragement and precious guidance. I would like to thanks, Mr. Janeshower Bohara for his incredible help and suggestions in selection of thesis title.

Sincerely,I am grateful to all those who have directly or indirectly contributed during the study. I would like to thank my friends for their encouragement, support and help during the research work, especially Mr. Om Prakash Mahato for his valuable suggestions and co-operation in my research work. Finally, I don't have adequate word to express my indebtedness to my family for their love, support and encouragement.

Md Hasan Ansari

March, 2017

ABSTRACT

Multiple sequence alignment (MSA) is an important problem in molecular biology. Biological sequences are aligned with each other vertically to show possible similarities or differences among these sequences. To solve an MSA problem is to find an alignment of multiple sequences with the highest score based on a given scoring criterion among sequences. Dynamic programming algorithms like Needleman-Wunch and Smith-Waterman produce accurate alignments but these algorithms are computation intensive, computational complexity of $O(n^2)$ and are limited to a small number of short sequences. Similarly multiple sequence alignment that processes the sequences one by one, called star alignment, takes time until $O(k^2n^2)$. However the computation result still has high accuracy. Consequently, it is very important to get a better way to improve the performance. To achieve this, a MapReduce model of star alignment is designed and implemented that executes in parallel on a hadoop clusters. Since hadoop already handles work/job dispatching and work balance among distributed worker nodes, we need note handle node failure and load balancing required with the traditional distributed computing. The experimental result shows that the MapReduce model of star alignment improve the execution time by 3 times with 8 physical nodes than single node with datasets size greater than 1 GB.

Keywords: Bioinformatics, Multiple Sequence Alignment, Needleman-wunch, Star Alignment, Parallelization, Hadoop, MapReduce.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

DNA	Deoxyribonucleic Acid
RNA	Ribonucleic Acid
MSA	Multiple Sequence Alignment
PAM	Point Accepted Matrix
BLOSUM	Blocks Substitution Matrix
DP	Dynamic Programming
HDFS	Hadoop Distributed File System
VM	Virtual Machine
BLAST	Basic Local Alignment Search Tool

CHAPTER 1: INTRODUCTION

1.1 Background

Life on earth originated and then evolved from a universal common ancestor approximately 3.8 billion years ago. Repeated specification and the divergence of life have occurred throughout this time due to shared sets of biological and morphological traits, or by the shared DNA sequences [1]. In bioinformatics, sequence alignment deals with the comparison of two or more Deoxyribonucleic Acid (DNA), Ribonucleic Acid (RNA) and protein sequences with each other. The comparison aims to identify regions of similarity that may be a consequence of functional, structural, or evolutionary relationships between the sequences.

All life on earth contains DNA and many believe that all life originate from the same DNA (or at least RNA, Walter Gilbert). That is, all DNA have a common ancestor, meaning that at some point back in time, it is believed that there was a single (very basic) life form, from which all life known today has evolved. DNA is made of ribose molecules with one of the four nucleic acids; Guanine (G), Cytosine (C), Adenine (A) or Thymine (T) attached.

Multiple sequence alignment (MSA), the simultaneous alignment among three or more nucleotide or amino acid sequences, is one of the most essential tools in molecular biology. Sequence alignments are used to help demonstrate homology between new and existing sequences, to suggest primers for polymerase chain reaction, and to predict the secondary or tertiary structure of RNA and proteins [2]. Therefore, the development of efficient and accurate automatic methods for multiple sequence alignments is a very important research topic. Sequence alignment is the arrangement of two or more sequences of “residues” that maximizes the similarities between them. In order for a multiple alignment to be meaningful in this context, all sequences in the multiple alignment must have a common origin. The goal of multiple sequence alignment is to align sequences according to their evolutionary relationships.

MSA is important because it reconstructs phylogenetic trees, which in turn predict the function of an unknown protein by aligning its sequences with some other known functions. The various match, mismatch, and gap ("-") events then represent possible reconstructions of the evolution of those related sequences. If a sequence alignment occurs between two sequences, then it is called a pairwise alignment [3], [4] and the main goal is to find the similar or closely related parts between two sequences. If the alignment involves more than two sequences, then it is called a multiple sequence alignment and the main goal is to find the consensus parts among the sequences. For small lengths and small numbers of sequences, it is possible to create the alignment manually. However, efficient algorithms to align such sequences are essential for alignments with more than eight sequences.

MSA problems are solved using several different methods, such as classical, progressive, and iterative algorithms. These algorithms follow either global or local alignment strategies. In global alignments, sequences are aligned over their whole length. By contrast, local alignments identify regions of similarity within a sub sequence [5]. Local alignments are often preferable, but can be more difficult because of the additional challenge of identifying the regions of similarity. A general global alignment technique is the Needleman–Wunsch algorithm [3], which is based on dynamic programming. The Smith–Waterman algorithm [4] is a general local alignment method which is also based on dynamic programming. The dynamic programming (DP) approach is good at finding the optimal alignment for two sequences. However, the complexity of this method grows significantly for three or more sequences.

Star Alignment method to perform multiple sequence alignment. In this method, sequences S_1-S_k, S_2-S_k, S_3-S_k, …, S_{k-1}-S_k are compared one by one in which they will be pair-wise alignment [6]. In simple terms, the complexity of Star alignment algorithm is quite high which is $O(k^2\ n^2)$. To reduce the execution time significantly, it needs to modify the Star Alignment algorithm by implementing parallel programming using Map Reduce model of Hadoop. Hadoop [7] is one of the most popular distributed processing framework/systems in recent years. It provides the Map Reduce programming model and

an associated implementation for processing large data sets in parallel, using a cluster of nodes.

1.2 Problem Statement

Pair-wise sequence alignment is a technique of comparing the similarity of two organisms. It is the basic technique in DNA sequence alignment. There is an extraordinary number of data sequences when they are compared. Problems when comparing the huge data sequences are accuracy and efficiency. Dynamic programming models like Needleman-Wunsch and Smith-Waterman produce accurate alignments, but these algorithms are high computational complexity. Similarly, multiple sequence alignment that processes the sequences one by one, called Star Alignment, takes time until $O(k^2n^2)$ [6]. Therefore, they have a timing issue problem while processing the data. However, the computation result still has high accuracy. Consequently, it is very important to get a better way to improve the performance. This can be done by using parallelization methodology of Map and Reduce framework.

1.3 Objective

The objective of the thesis is to develop a MapReduce model of star alignment algorithm for multiple sequence alignment.

1.4 Scope of work

The scope of this thesis work is to develop a MapReduce model of star alignment algorithm for multiple sequence alignment. The main applications of sequence alignments have included phylogenetic tree reconstruction, Protein family prediction, and pattern identification.

1.5 Organization of the Thesis

The list below presents the organization of the chapters which make up this thesis. Also given is a brief description of the topics each chapter's deals with.

- Chapter 2 covers the necessary background relating to previous work done and general introduction of Needleman-Winch and star alignment algorithms.
- Chapter 3 includes a theoretical basis on DNA, MapReduce, Hadoop and sequence alignment, building blocks, concepts, uses and current alignment methods and substitution matrixes.
- Chapter 4 covers the methodologies, system model, algorithms used, and datasets used in evaluation of the model.
- Chapter 5 provides the test environment, experimental results, and execution time analysis and sample outputs.
- Chapter 6 provides summary of the works and future work.

.

CHAPTER 2: LITERATURE REVIEW

Efficient sequence alignment is one of the most important and challenging activities in bioinformatics. Many algorithms have been proposed previously to perform sequence alignment activities. Dynamic Programming (DP) algorithms such as Needleman-Wunsch [3] and Smith-Waterman [4] produce accurate scores. However, these algorithms are demand high computational power. The progressive approximation method implemented in ClustalW [8]. Progressive MSA aligns the closest sequences first and successively adds in more distant ones. This method is very fast and straightforward but it can easily get caught in local minima. This is because; once a sequence has been aligned it cannot be modified again, even if it is suboptimal when other sequences are subsequently aligned. A time efficient approach to sequence alignment that coupled with data and computational parallelism of hadoop data grids improves the accuracy and speed of sequence alignment [9]. Genomic analysis usually includes a pipeline of three stages: sequence alignment, data conversion, and advanced analysis. Parallelizing genomic analysis is not a simple task. A distributed analysis pipeline is designed and implemented that executes the pipeline in parallel on a hadoop cluster (physical machines or VM nodes) [10]. The healthcare applications can scale well on commercial big data platforms that implement MapReduce framework [11]. Cloud computing and MapReduce framework play an important role in bioinformatics intensive application in achieving parallelization since it provides a consistent performance over time and it provides good fault tolerant mechanism [12].

2.1 Needleman Wunsch Algorithm

The Needleman–Wunsch algorithm performs a global alignment of two sequences. It is commonly used in bioinformatics to align protein or nucleotide sequences. The algorithm was published in 1970 by Saul B. Needleman and Christian D. Wunsch [3]. The Needleman–Wunsch algorithm is an example of dynamic programming and was the first application of dynamic programming to biological sequence comparison. It is sometimes referred to as the optimal matching algorithm. This global sequence alignment method explores all possible alignments and chooses the best one (the optimal global alignment).

It does this by reading in a scoring matrix and a gap penalty (penalties) that contains values for every possible residue or nucleotide match and summing the matches taken from the scoring matrix.

2.2 Smith Waterman Algorithm

The Smith–Waterman [4] algorithm performs local sequence alignment; that is, for determining similar regions between two strings or nucleotide or protein sequences. Instead of looking at the total sequence, the Smith–Waterman algorithm compares segments of all possible lengths and optimizes the similarity measure.

The algorithm was first proposed by Temple F. Smith and Michael S. Waterman in 1981. Like the Needleman–Wunsch algorithm, of which it is a variation, Smith–Waterman is a dynamic programming algorithm. As such, it has the desirable property that it is guaranteed to find the optimal local alignment with respect to the scoring system being used (which includes the substitution matrix and the gap-scoring scheme). The main difference to the Needleman–Wunsch algorithm is that negative scoring matrix cells are set to zero, which renders the (thus positively scoring) local alignments visible. Backtracking starts at the highest scoring matrix cell and proceeds until a cell with score zero is encountered, yielding the highest scoring local alignment.

2.3 Star Alignment Algorithm

Multiple sequence alignment that processes the sequences one by one, called star alignment. Each sequence is compared one by one in pairs by performing pair-wise alignment using Needleman-Wunch algorithm. The star alignment algorithm runs faster, and it is therefore suitable for the MSA of similar DNA sequences [6]. The main approach underlying the star alignment is to transform MSA into pairwise alignment based on center sequence. This center sequence is selected and other sequences are pairwise aligned to the center sequence. Then, all of the inserted spaces are summed to obtain the final MSA result.

CHAPTER 3: THEORETICAL BACKGROUND

3.1 DNA

Deoxyribonucleic acid (DNA) is a molecule that encodes the genetic instructions used in the development and functioning of all known living organisms and many viruses. Along with RNA and proteins, DNA is one of the four major macromolecules essential for all known forms of life. Most DNA molecules are double-stranded helices, consisting of two long biopolymers of simpler units called nucleotides. Each nucleotide is composed of a nucleobase (guanine, adenine, thymine, and cytosine), recorded using the letters G, A, T, and C, as well as a backbone made of alternating sugars (deoxyri-bose) and phosphate groups (related to phosphoric acid), with the nucleobases (G, A, T,C) attached to the sugars. DNA is well-suited for biological information storage, since the DNA backbone is resistant to cleavage and the double-stranded structure provides the molecule with a built-in duplicate of the encoded information.

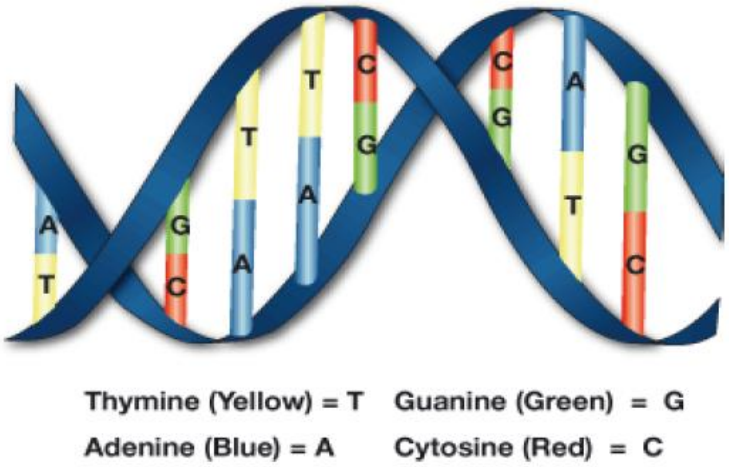


Figure 3.1: Structure of DNA Sequence [13]

3.2 MapReduce

MapReduce [14] is a programming model and an associated implementation for processing and generating large data sets. Users specify a map function that processes a key/value pair to generate a set of intermediate key/value pairs and a reduce function that merges all intermediate values associated with the same intermediate key. Programs

written in this functional style are automatically parallelized and executed on a large cluster of commodity machines. The run-time system takes care of the details of partitioning the input data, scheduling the program's execution across a set of machines, handling machine failures and managing the required inter-machine communication. This allows programmers without any experience with parallel and distributed systems to easily utilize the resources of a large distributed system. A typical MapReduce computation processes many terabytes of data on thousands of machines. Programmers find the system easy to use: hundreds of MapReduce programs have been implemented and upwards of one thousand MapReduce jobs are executed on Googles clusters every day. MapReduce provides an abstraction that involves the programmer defining a "mapper" and a "reducer," with the following signatures:

- Map: (key1, value1) → list (key2, value2)
- Reduce: (key2, list (value2)) → list (key3, value3).

3.2.1 Execution Overview

The Map invocations are distributed across multiple machines by automatically partitioning the input data into a set of M splits. The input splits can be processed in parallel by different machines. Reduce invocations are distributed by partitioning the intermediate key space into R pieces using a partitioning function (e.g., hash (key) mod R). The number of partitions (R) and the partitioning function is specified by the user.

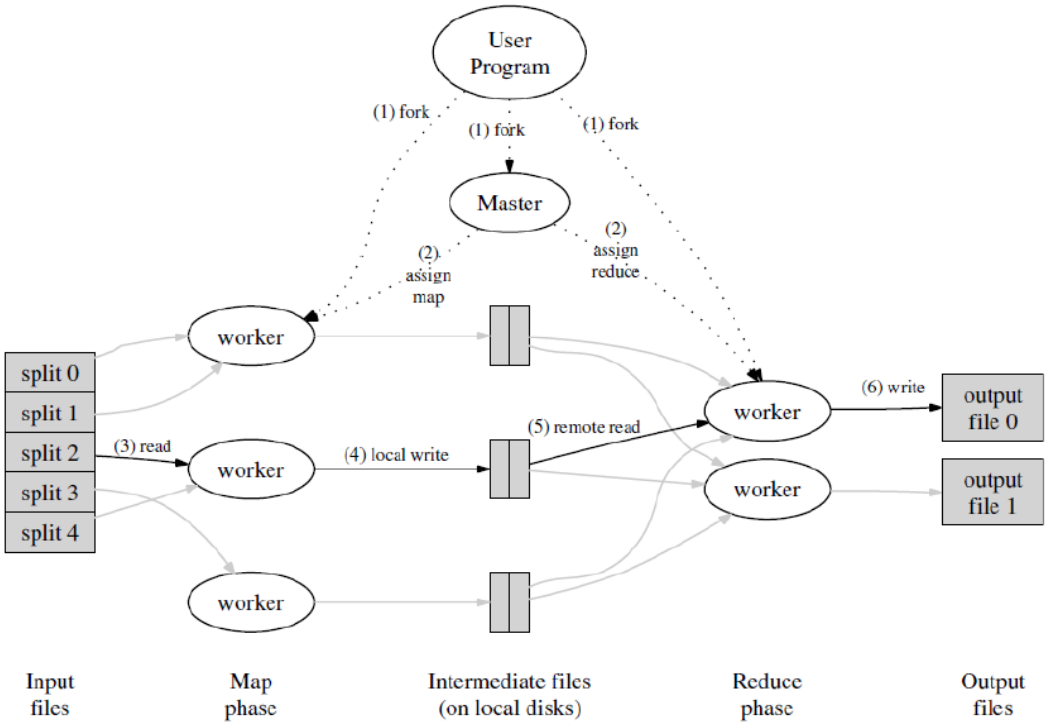


Figure 3.2: MapReduce Execution overview [7]

Figure 3.2 shows the overall flow of a MapReduce operation in the implementation. When the user program calls the MapReduce function, the following sequence of actions occurs (the numbered labels in Figure 3.2 correspond to the numbers in the list below):

1. The MapReduce library in the user program first splits the input files into M pieces of typically 16 megabytes to 128 megabytes (MB) per piece (controllable by the user via an optional parameter). It then starts up many copies of the program on a cluster of machines.
2. One of the copies of the program is special the master. The rest are workers that are assigned work by the master. There are M map tasks and R reduce tasks to assign. The master picks idle workers and assigns each one a map task or a reduce task.
3. A worker who is assigned a map task reads the contents of the corresponding input split. It parses key/value pairs out of the input data and passes each pair to the user-defined Map function. The intermediate key/value pairs produced by the Map function are buffered in memory.

4. Periodically, the buffered pairs are written to local disk, partitioned into R regions by the partitioning function. The locations of these buffered pairs on the local disk are passed back to the master, who is responsible for forwarding these locations to the reduce workers.
5. When a reduce worker is notified by the master about these locations, it uses remote procedure calls to read the buffered data from the local disks of the map workers. When a reduce worker has read all intermediate data, it sorts it by the intermediate keys so that all occurrences of the same key are grouped together. The sorting is needed because typically many different keys map to the same reduce task. If the amount of intermediate data is too large to fit in memory, an external sort is used.
6. The reduce worker iterates over the sorted intermediate data and for each unique intermediate key encountered, it passes the key and the corresponding set of intermediate values to the users Reduce function. The output of the Reduce function is appended to a final output file for this reduce partition.
7. When all map tasks and reduce tasks have been completed, the master wakes up the user program. At this point, the MapReduce call in the user program returns back to the user code.

After successful completion, the output of the mapreduce execution is available in the R output files (one per reduce task, with file names as specified by the user). Typically, users do not need to combine these R output files into one file they often pass these files as input to another MapReduce call, or use them from another distributed application that is able to deal with input that is partitioned into multiple files.

3.3 Hadoop

Hadoop [7] is a popular open source implementation of MapReduce, which is a powerful tool designed for deep analysis and transformation of very large datasets which is inspired by Google's MapReduce and Google File System. It enables applications to work with thousands of nodes and petabytes of data.

Hadoop uses a distributed file system called Hadoop Distributed File System (HDFS), which creates multiple replicas of data blocks and distributes them on computer nodes throughout a cluster to enable reliability and has extremely rapid computations to store data as well as the intermediate results. The Hadoop runtime system coupled with HDFS manages the details of parallelism and concurrency to provide ease of parallel programming with reinforced reliability. In a Hadoop cluster, a master node controls a group of slave nodes on which the Map and Reduce functions run in parallel.

3.4 Sequence Fundamentals

3.4.1 Alignment

The arrangement of two or more biological sequences in such a way that tells us at what point the sequences are similar and at what point they differ is known as alignment. An alignment is said to be the optimal one, if it has more similar sequences as compared to dissimilar sequences.

3.4.2 Sequence Alignment

Sequence alignment is a way of arranging the biological sequences so as to identify the region of similarity that may be a result of structural, functional, or evolutionary relationships between the sequences. In bioinformatics, the aligned sequences of DNA, RNA, or Protein are represented inside the matrix, in the form of rows. Gaps are inserted at some point in the sequences to achieve maximum similar character in a column. It aims to infer clues about the unknown sequence by inferring biological characteristics of the matched sequence. One of the most challenging tasks in sequence alignment is its repetitive and time-consuming alignment matrix computations.

3.4.3 Multiple Sequence Alignment

By referring to Figure 1, we can define multiple sequence alignment (MSA) as the optimal alignment technique of three or more sequences with or without inserting gaps. It plays an important role in sequence analysis and can also be used to judge and identify

the similarity between DNA, RNA or protein sequences. With these features, MSA is proved as an important tool for prediction of function and/or structure of an unknown protein sequences.

```
ATACGAT---------CTACG----GATGAAAGCGGGGGACCTTCGGGCCT---CGCGCT
ATACGCC---------CTACG----GGGGAAAGCAGGGGACCTTCGGGCCT---TGCGCG
ATAGAATTTAGTACCTCTACGAGGTGAGGTAGGCTGAGGAGCAAAAGGGAGGAATCCGCC
ATATTATGCTG----CCTGGATAACCAGGCTGCATCAAAGGCGGCTTTTTGCC-TCCGCT
ATATGTGACGGA---CCTGCATGGGTA--CCGTCTGGAAAG----TTTTT------CGGT
```

Figure 3.3: Example of a multiple sequence alignment

An MSA can be obtained by inserting gaps "-" at proper places such that no column in the sequences contains only gap character. Insertion of gaps will result in equal length sequences in the resulting alignment.

3.4.4 Gaps

In order to have the best resulting alignment, gaps are permitted within the sequences along with a user defined mechanism for penalizing these gaps. Gaps are inserted between the residues so that identical or similar characters are aligned in successive columns.

The values of gap penalties depend on the choice of matrix such as the PAM250, PAM350 or the Substitution matrices such as BLOSUM which are used for sequence alignment of proteins. A Substitution matrix assigns a score for aligning any possible pair of residues and must balance their values. Adopting a high gap plenty scheme will restrict the appearance of gaps within the alignment. On the other hand, a too low gap plenty scheme will allow the gaps to appear everywhere in the alignment.

3.4.5 Substitution Matrix

In case of matching DNA sequences, the substitution matrix is simple. It is either 1 or 0 depending on whether a match occurred or not. If there is a perfect match, which is 'A' aligns with 'A', 'G' aligns with 'G', and so on. In this case, the substitution matrix for DNA matching can be written as follows. (It is called identity matrix):

Table 3.1: Identity score matrix

	A	G	C	T
A	1	0	0	0
G	0	1	0	0
C	0	0	1	0
T	0	0	0	1

Table 3.1 shows the identity score matrix. This matrix has been used for the alignment of DNA sequences and also used for calculation of sum of pair score of the aligned sequences in this thesis. For protein sequence alignment, because the alphabet size increases from 4 to 20 and also because the scoring scheme of 1 for a match and 0 for a mismatch is not enough, the score matrix becomes more complicated. A substitution matrix describes the likelihood that two residue types would mutate to each other in evolutionary time. This is used to estimate how well two residues of given types match when they are aligned in a sequence alignment. There are many substitution matrices, such as PAM (Percent Accepted Mutations or Point Accepted Mutations), BLOSUM (BLOcks Substitution Matrix), etc.

CHAPTER 4: METHODOLOGY

The main approach underlying the star alignment algorithm is to transform MSA into pairwise alignment based on a "centre sequence". This centre sequence is selected, and other sequences are pairwise aligned to the centre sequence. Then, all of the inserted spaces are summed to obtain the final MSA result.

4.1 Pairwise Sequence Alignment

The sequence alignment DNA using Needleman-Wunsch was introduced in 1970. This algorithm employs an iterative matrix which is represented in a two-dimensional array for finding the best score of pair-wise alignment of two sequences. It has complexity of $O(n^2)$. The formula for calculating score is defined as follow [3]:

$$M[i,j] = Max \begin{cases} M[i-1,j-1] + sub(s[i], b[j]; \\ M[i-1,j] + del(s[i]); \\ M[i,j-1] + ins(b[j]; \end{cases} \quad (4.1)$$

There are four steps in a pair-wise sequence alignment algorithm. They are the initialization step, the matrix filling step, the backtracking matrix constructing step, and the alignment obtaining step.

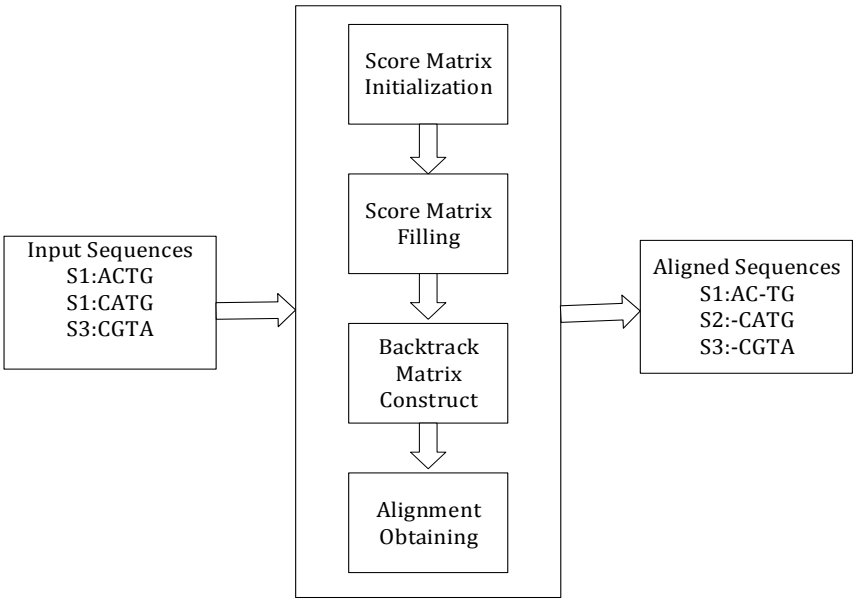


Figure 4.1: Pair-wise sequence alignment using Needleman-Wunsch

In the score initialization step the first row and the first column are filled with decrements of the gap score. Score matrix initialization and construction is done using Algorithm 4.1 [13].

Aligning sequences s_a and s_b of length m and n, respectively, with linear gap penalty. Here F is score matrix of dimensions n, m and d is the gap penalty.

```
begin
  initialization:
        F(0, 0) = 0
        for i=0 to m do
          F(0, i) = −i * d
        end
        for j=0 to n do
            F(j, 0) = −j* d
      end
  matrix fill:
       for i=1 to n do
         for j=1 to m do
              F(I, j) = max { F(i-1,j-1)+ s(x, y), F(i-1,j) – d, F(i,j-1) – d }
          end
       end
  end
```

Algorithm 4.1: Score Matrix Initialization and Construction

Backtrack matrix is constructed using Algorithm 4.2. Aligning sequences sa and sb of length m and n, respectively, with linear gap penalty [13].

```
begin
for i := 1 to n do
   for j := 1 to m do
      UP_Value = F(i – 1, j)
      Left_Value = F(i, j – 1)
      UP_Left_Value = F(i –1,  j – 1)
      if (s^j_a := s^j_b) do
         BM(i, j) = '*'
      else
         if (Left_Value >= U_Value) do
            if (Left_Value + gap_penalty >= UP_Left_Value + Mismatch) do
               fill BM(i, j) with '–'
            else
               fill BM(i, j) with '*'
```

```
            end
        else
            if (UP_Value + gap_penalty >= UP_Left_Value + Mismatch) do
                fill BM(i, j) with '#'
            else
                fill BM(i, j) with '*'
            end
        end
    end
end
end
```

Algorithm 4.2: Backtrack Matrix Construction Algorithm

Final alignment process starts from the lower right corner cell and records sequences from right to left as shown in figure 4.2.

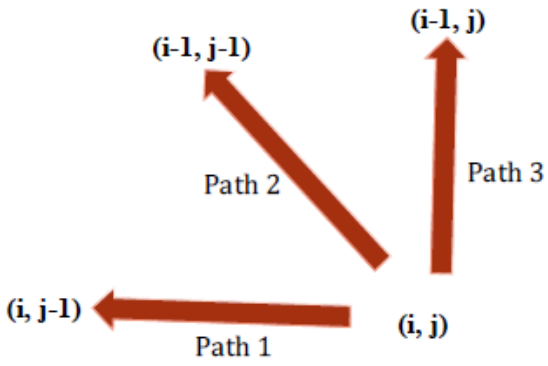


Figure 4.2: Backtrack matrix traverse flow

Above figure shows how to choose the next cell path to obtain an alignment step. If cell (i, j) is "*", choose path 2. If cell (i, j) is "-", choose path 1. If cell (i, j) is "#", choose path 3 [13].

4.2 Star Alignment

In Star Alignment, each sequence from S_1 to S_k was compared one by one in pairs by performing pair-wise alignment using Needleman-Wunsch algorithm. Number of pairs of sequences can be calculated by the following combination of sequence:

$$c(k, 2) = k(k-1)/2 \quad (4.2)$$

After getting the score of each pair, then the center star was chosen. Selection of the star center has its own mechanism considering each star center candidate sequence has a different sequence alignment results. In the flowchart [6] below, if *Sc* has the highest score with longest string then it becomes center star sequence.

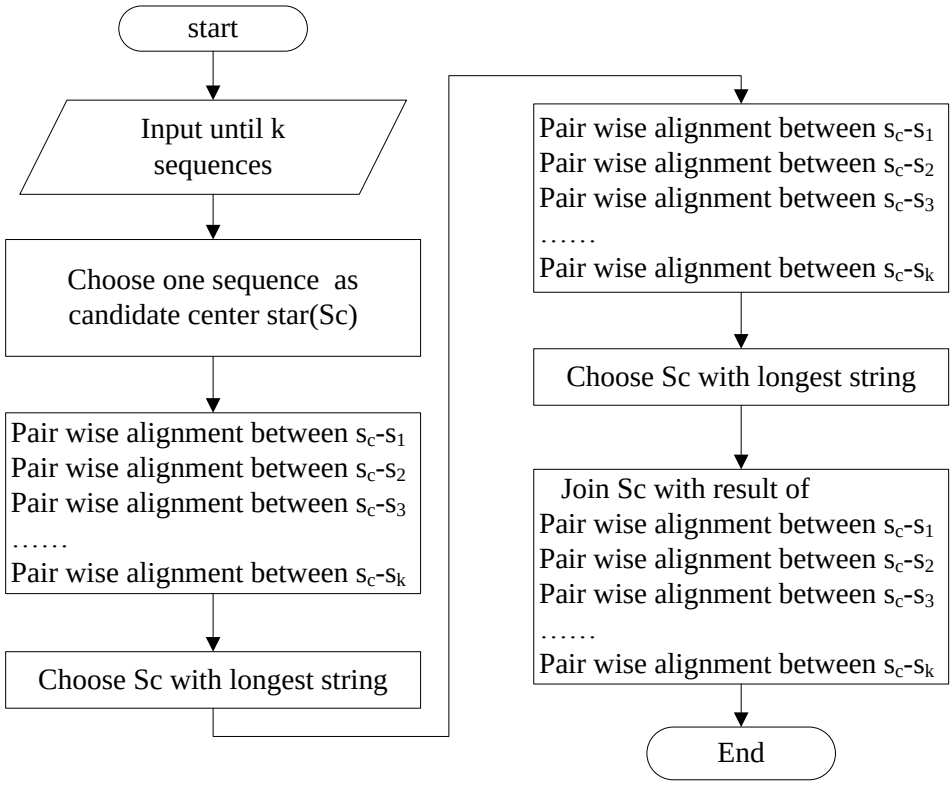


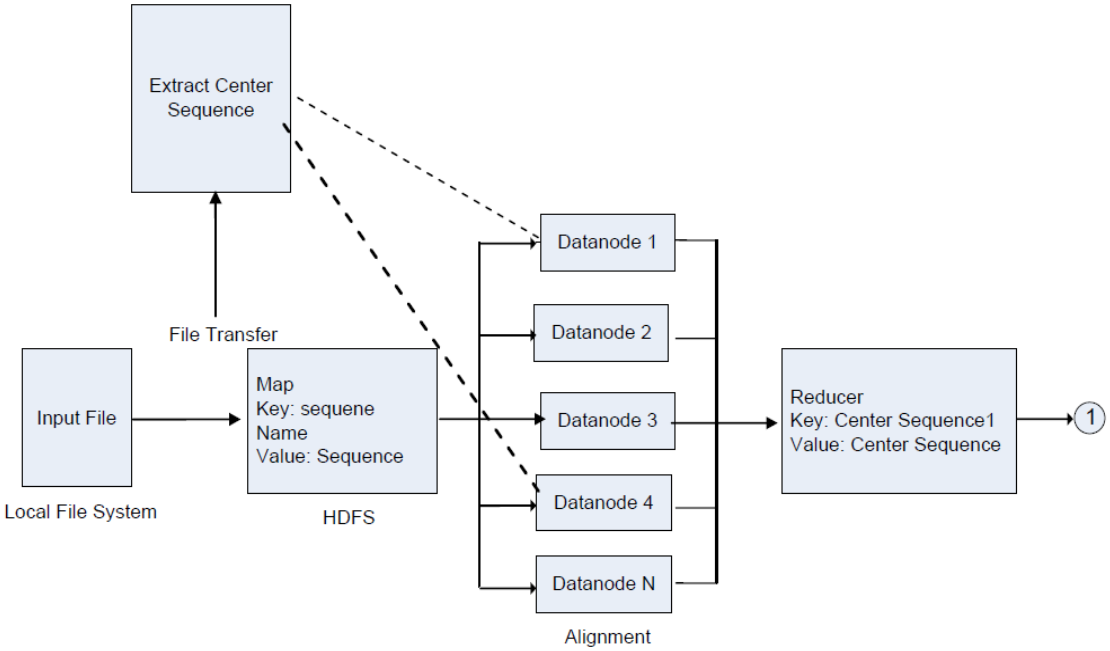
Figure 4.3: Flowchart of Star Alignment Algorithm

Next, every input sequence was aligned with the center star (*Sc* sequence). This process generated a new *Sc*. This *Sc* was then aligned with the next input sequence until the last sequence. The purpose of that process is to get the highest score representing the similarity between *Sc* and the other sequences. The complexity of this process is $O(2k\,n^2)$. Overall complexity using star alignment can be calculated by [6]:

$$Complexity = \frac{k(k-1)}{2}n^2 + (k-1) + 2kn^2 \qquad = O\left(k^2n^2\right) \tag{4.3}$$

4.3 Parallel Implementation

The map reduce model of star alignment algorithm is outline in figure 4.4. The map reduce parallel framework is employed in two stages. In the first stage candidate center star sequence is chosen, and pairwise alignment between candidate center start and other sequences are done. In the second stage final center start sequence is chosen and pairwise alignment between center star and other sequences are done.



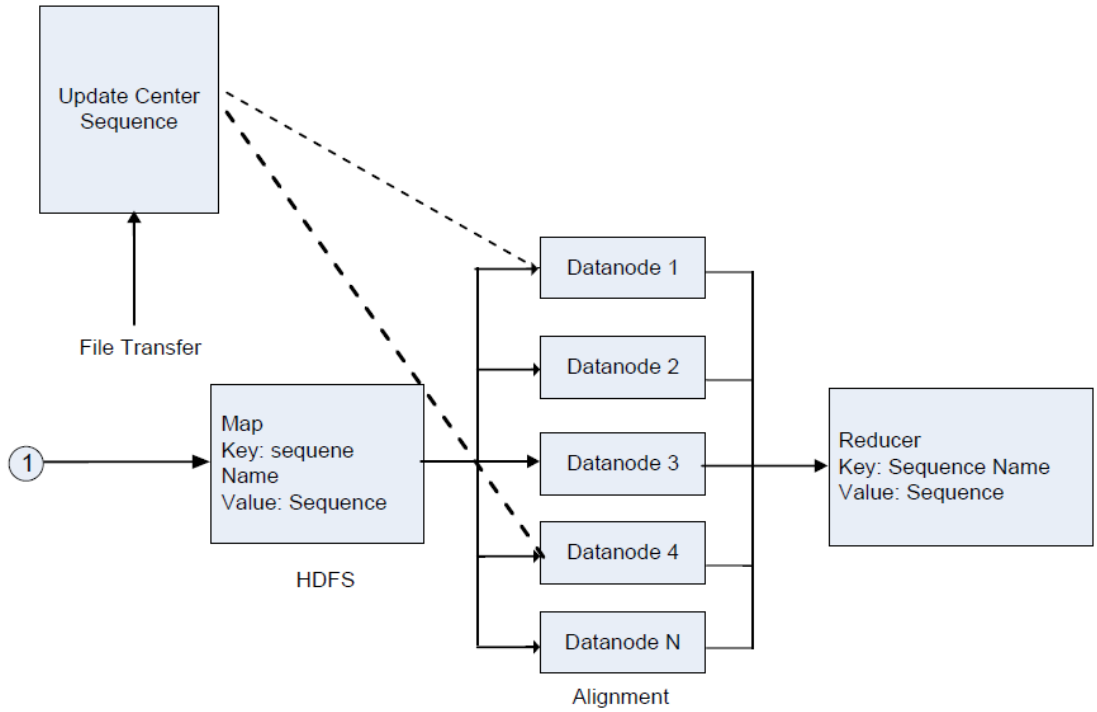


Figure 4.4: Map Reduce model of star alignment algorithm

Entries in Map Reduce are recorded with a (key, value) format. Key is denoted as the sequence name and value as the DNA sequence. All of the input sequences are formatted as (key, value) pairs for hadoop. In the first stage of the Map function, the data file is automatically divided into several split files, which size is default block size of HDFS (128 MB for Hadoop 2.7.3 version). These split files are sent to different data nodes and aligned to the centre sequence in parallel. After alignment, the centre sequence and the sequence in the split file are updated with inserted spaces. They are still recorded with a (key, value) format, where the key is the sequence name and the value is the two updated aligned sequences. The flow of the map function is shown in Fig. 4.5 and Algorithm 4.3. Then, the output (key, value) pairs reach the Reduce stage.

In the first stage of the Reduce function, the data are note processed and are output to the HDFS file system directly. Then, the data are collected from the HDFS file system on a local computer, and the aligned centre sequences are extracted and collected. For the *k* aligned sequences, maximum spaces between every two neighboring characters are counted. The maximum spaces are retained for the final centre sequence.

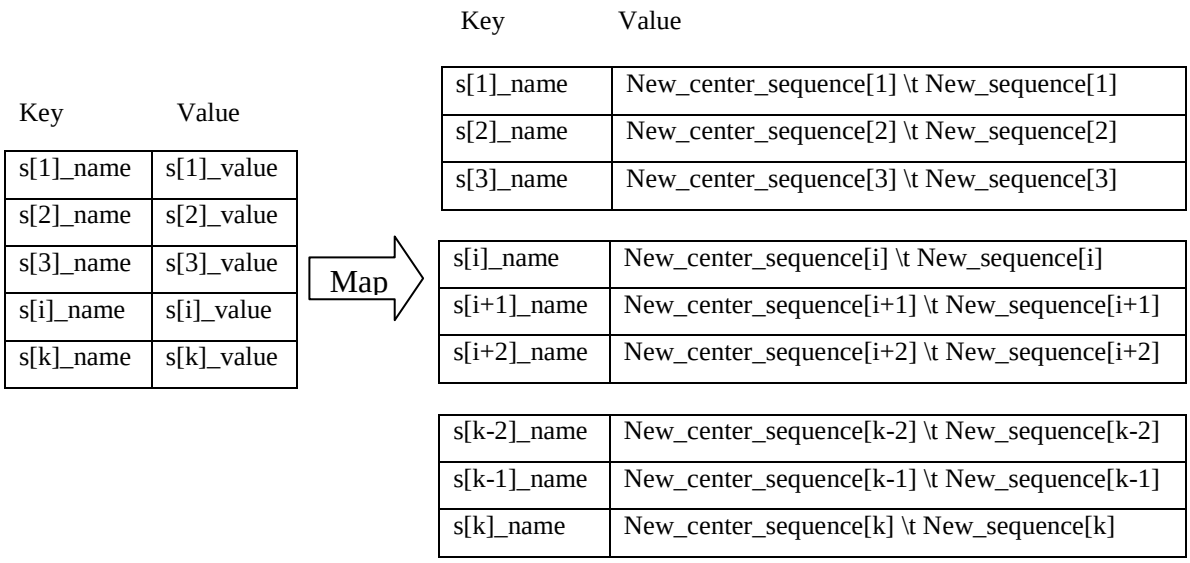


Figure 4.5: The input and output of map function in the first stage

Algorithm 4.3.Function Map_1

For each Map_1(key=sequence_name[i], value=sequence[i])

1：for sequence i ← 1 to Data_Size/64 M do

2：key ← sequence_name[i]

3：value ←star_alignment_algorithm (center star sequence, sequence[i]);

4：end for

The second Map-Reduce phase is similar to the first stage. All of the aligned sequences from the first stage are aligned again to the Final Centre Sequence. Because the Final Centre Sequence has the maximum number of spaces between every character, there will be no space inserted into the Final Centre Sequence. Therefore, all of the other sequences will be aligned to the same length as the Final Centre Sequence, which will be the final alignment result. The input and output of the map function in the second stage are shown as figure 4.6:

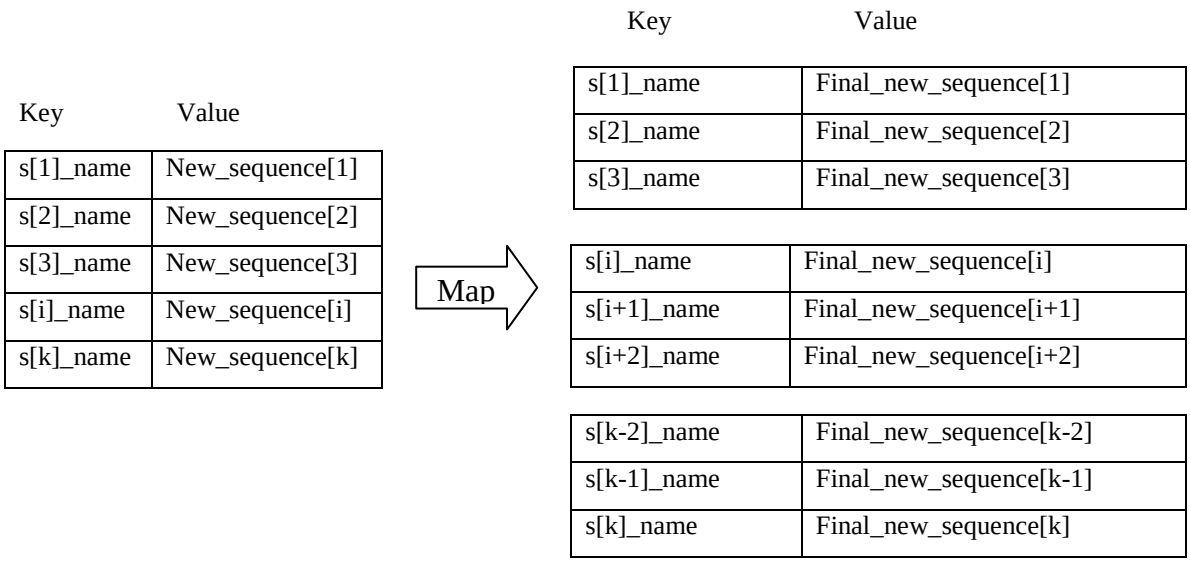


Figure 4.6: The input and output of map function in the second stage

4.4 Data Collection

The sample data is collected from National Center for Biotechnology Information (ftp://ftp.ncbi.nlm.nih.gov/genbank) in the FASTA format.

For example:

```
>gi|292493920|ref|NC_013962.1| Candidatus Riesia pediculicola USDA plasmid pPAN, complete sequence
ATAAAATTCCCGTCTTCTAAGGAAGAAGTCCCGAAAGAAAGGGAGAGTAAATGAAAAGGAATTGATTTTTTTTTTTCA
AAAAAAAATGGTCTAACAGGTAAGGGAAATTTGAGGTCATGAAGAAGAATCCTAAGTTAAATACAACTATTGAAGATA
TAAGAATTTGGAAGAAGAACTCTTCAAAGTTTGCTGCTTTAA…..
```

From the collected data different data sets are prepared, that contains the different number of sequences and different size of the sequence file.

CHAPTER 5: RESULT AND DISCUSSION

In this section, the test environment is introduced. The result verification for star alignment algorithm (sequential and parallel) with the result of clustalW is done. Finally the execution time of the parallel algorithm for different datasets and for different number of nodes is shown. The scoring schemes for match column is 1, mismatch column is 0 and for gap is -1 have been used. These scores are needed in pairwise sequence alignment steps.

5.1 Test Environment

The system is deployed on Intel core i3 PC with 4 GB RAM and total 4 cores. The test environment consists of a total 8 physical nodes hadoop cluster and the hardware configuration of each node is shown in Table 5.1. The software and Hadoop configuration is shown in Table 5.2.

Table 5.1: Hardware of each node

CPU	Intel core i3 / 4 cores @ 3.30 GHz
Memory	4 GB
Disk Size	100 GB

Table 5.2: Software and Hadoop configuration of each node

OS	Ubuntu 16.04.2 / 32 bit
Java	1.8.121 version
Hadoop	2.7.3 version

5.2 Verification of Result

The correctness of the result is verified by comparing it with the sequential result and it seems result of both is same and also it is compared with the result of clustalW by using the same input data. The aligned multiples sequences from parallel star algorithm and clustalW are shown in Figures 5.1 and 5.2 respectively. An asterisk is used to indicate a match column.

```
CLUSTAL 2.1 multiple sequence alignment

Juniperus          --------GGTGTTTCCAGTGGCGAACGGGTGAGTAATG--CGTAAGAACCTGCCCTTGG
Burkholderia       -----CACCTGGTGGCGAGTGGCGAACGGGTGAGTAATA--CATCGGAACATGTCCTGTA
Aeromonas          TGCTACTTTTGCCGGCGAGCGGCGGACGGGTGAGTAATG--CCTGGGAAATTGCCCAGTC
Acetohalobium      -------CAGTTAGTAAAGCGGCGGACGGGTGA-GTAACGCGTGAGTAATCTACCTTTAA
Janibacter         -------CGAGTGGATCAGTGGCGAACGGGTGAGTACACACGTGAGCAACCTGCCCCAGA
                                   ** **** ********             **  *  *

Juniperus          GAGGGGAACAACAGCTGGAAACGGTTGCTAATACCCCATAGAATTTAGTACCTCT
Burkholderia       GTGGGGGATAGCCCGGCGAAAGCCGGATTAATACCGCATACGATCTACGG-----
Aeromonas          GAGGGGGATAACAGTTGGAAACGACTGCTAATACCGCATACGCCCTACGGGGG--
Acetohalobium      GTCTGATATAACTTCTCGAAAGGGAAGCTAATTTCGGATATTATGCTGCCTGGAT
Janibacter         CTCTGGAATAAGCGCTGGAAACGGCGTCTAATACTGGATATGTGACGGACCTGCA
                       *  * *       ****       ****     ***
```

Figure 5.1: Multiple sequence alignment result of data set RefSeqtest.fasta by using ClustalW

The clustalW simulation platform is obtained from the web site http://www.clustal.org/download/current and it can also be accessed online in http://www.ch.org/software/ClustalW.html.

```
Burkholderia       GGTGTTTC-CAGTGGCGAACGGGTGA---GT-A-ATGCGTAAG-AACCTGCCCTTGGGAG
Aeromonas          CAGTTAGTAAAGCGGCGGACGGGTGA----GTA-ACGCGTGAGTAATCTACCTTTAAGTC
Juniperus          CGAGTGGATCAGTGGCGAACGGGTGA---GT-ACACACGTGAGCAACCTGCCCCAGACTC
Acetohalobium      CACCTGGTGGCGAGTGGCGAACGGGT-GAGT-A-ATACATCGG-AACATGTCCTGTAGTG
Janibacter         TGCTACTTTTGCCGGCGAGCGGCGGACGGGTGAGTAATGCCTGGGAAATTGCCCAGTCGA
                               *  *       *       *         *  *  *  *

Burkholderia       GGGA-ACAACAGCTGGAAACGGTTGCTAATACCCCATAGAATTTAGTACCTCT
Aeromonas          T-GATATAACTTCTCGAAAGGGAAGCTAATTTCGGATATTATGCTGCCTGGAT
Juniperus          TGGA-ATAAGCGCTGGAAACGGCGTCTAATACTGGATATGTGACGGACCTGCA
Acetohalobium      GGGG-ATAGCCCGGCGAAAGCCGGATTAATACCGCATACGAT-CT-ACG-G--
Janibacter         GGGGGATAACAGTTGGAAACGACTGCTAATACCGCATACGCC-CT-ACGGGGG
                     *  * *       ****       ****     ***
```

Figure 5.2: Multiple sequence alignment result of data set RefSeqtest.fasta by using MapReduce model

Above figures shows the results from the MapReduce model and ClustalW. An asterisk has been used to represent the match columns. There is little difference in finding the number of match columns in these results. This difference is due to the use of different scoring schemes. In this thesis identity score matrix has been used where as in ClustalW BLOSUM and PAM matrixes are used.

5.3 Execution Time Analysis

In order to evaluate the map reduce model of star alignment approach, two DNA data files of different sequence length are used. One file is of human mitochondrial genomes, which contains total 672 genome sequences of average sequence length 8200 bp. Another data file is 16s rRNA, which contains total 708,129 sequences of average sequence length 1442 bp. The size of the sequence files are 10 MB and 1457 MB respectively.

5.3.1 Varying number of sequences of same size

By using the above two sequence files different reference sequence files are generated of different sequence numbers, which are tabulated in table 5.3 and also its execution time with sequential and parallel algorithm are shown.

Table 5.3: Execution time of sequential and parallel algorithm varying the number of sequences with each sequence average length 1442 bp.

Sequence File Name	No of sequences	Size of sequence file	Sequential (seconds)	Parallel (seconds)
RefSeq1.fasta	2	4 KB	1	19
RefSeq2.fasta	3	5 KB	2	22
RefSeq3.fasta	4	7 KB	2	23
RefSeq4.fasta	6	10 KB	5	23
RefSeq5.fasta	8	13 KB	8	24
RefSeq6.fasta	10	16 KB	11	24
RefSeq7.fasta	20	30 KB	40	26
RefSeq8.fasta	30	45 KB	90	26
RefSeq9.fasta	40	60 KB	153	28
RefSeq10.fasta	50	74 KB	235	29
RefSeq11.fasta	100	148 KB	909	32
RefSeq12.fasta	150	222 KB	2094	36
RefSeq13.fasta	200	296 KB	3740	43
RefSeq14.fasta	250	369 KB	5698	49

As shown in the above table, Different reference sequences have been generated by using the 10 MB data set file. These reference sequences contain different number of sequences. From the above table it is shown that sequential version of the star alignment algorithm is good for alignment of 10 numbers of sequences than the MapReduce model.

For more than this little increase in time but as the numbers of sequences increases from 50 sequential version of star alignment execution time grows exponentially and it becomes infeasible where as the MapReduce model execution time is quite low as compared to the sequential. A line graph plot of the above result is shown in figure 5.3.

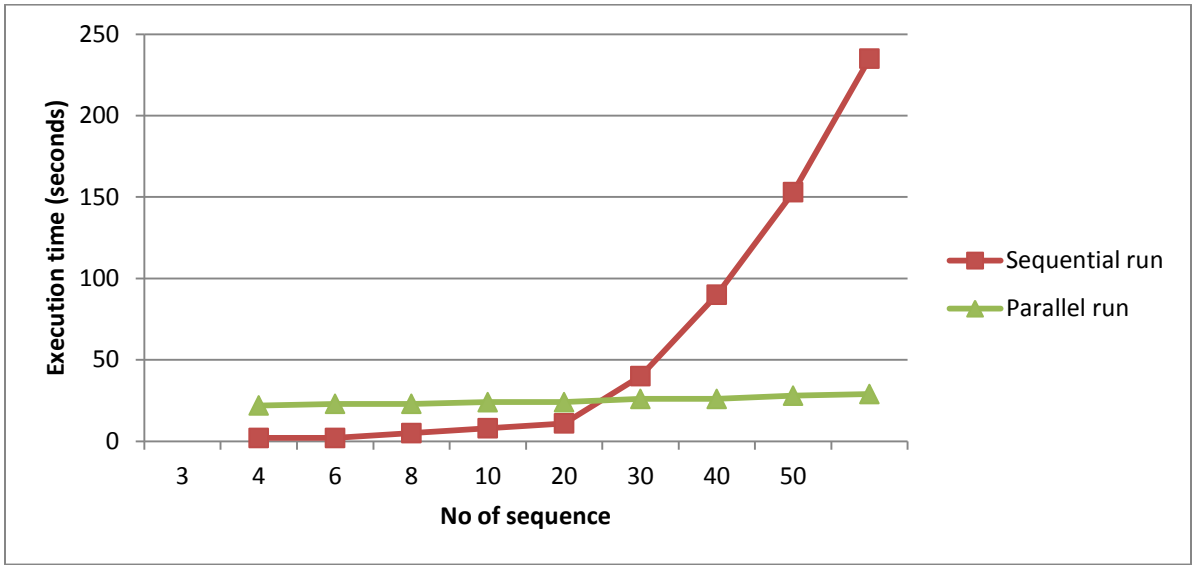


Figure 5.3: Execution time of sequential and parallel algorithm varying the number of sequences with each sequence average length 1442 bp.

Another experiment has been done in human mitochondrial genomes datasets in which average length of DNA sequences are 8200 base pair. The result of this experiment is shown in table 5.4.

Table 5.4: Execution time (in seconds) of sequential and parallel algorithms with varying number of sequences of average sequence length 8200 bp.

Sequence File Name	Number of Sequences	Size of Sequence File	Sequential (seconds)	Parallel (seconds)
REFSEQ2.fasta	2	16 KB	41	38
REFSEQ3.fasta	3	24 KB	95	41
REFSEQ4.fasta	4	32 KB	160	44
REFSEQ6.fasta	6	48 KB	333	73
REFSEQ8.fasta	8	63 KB	569	85
REFSEQ10.fasta	10	79 KB	885	96
REFSEQ20.fasta	20	158 KB	3046	171
REFSEQ30.fasta	30	236 KB	11184	282

The result shows that for DNA sequences having higher length (approx 8200bp) the sequential algorithm is not suitable for multiple sequence alignment. For more than 10 number of sequences sequential algorithm is infeasible because it takes time more than 3 hours to align 30 sequences, where as MapReduce model align this in less than 5 minutes. The line graph plot of the above result is shown in figure 5.4.

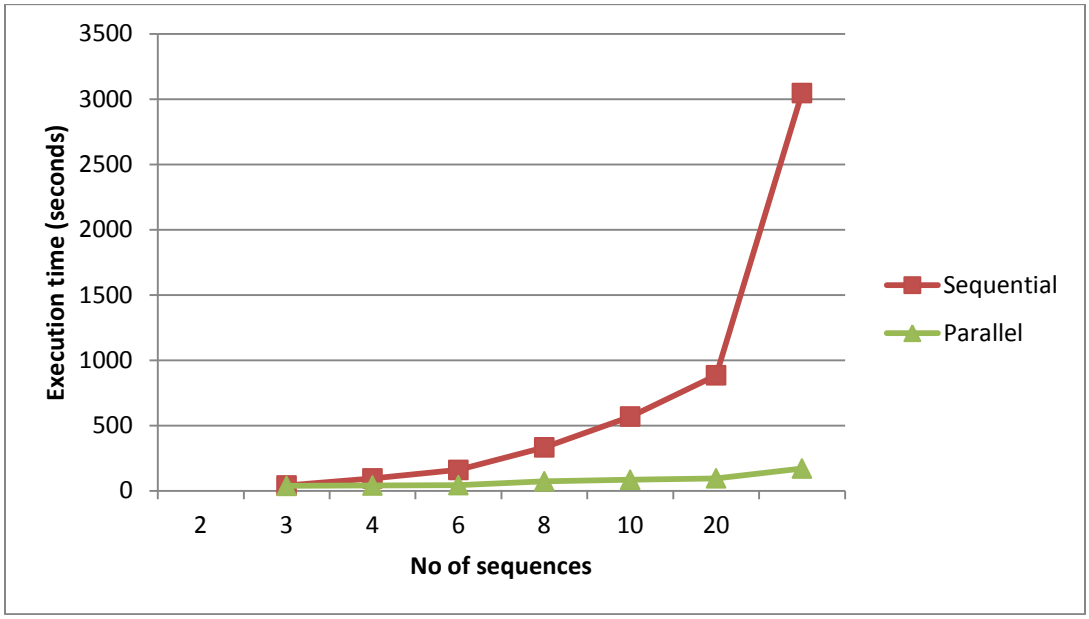


Figure 5.4: Execution time (in seconds) of sequential and parallel algorithms with varying number of sequences of average sequence length 8200 bp.

5.3.2 Varying number of nodes

A set of experiments is done in datasets 16srRNA with datasets file size from 102 MB to 1457 MB and the average length of the sequence is 1442 bp. These experiments execution time in one node cluster, two node cluster, three node cluster, four node cluster, six node cluster and eight node clusters are shown in table 5.5 and 5.6 and it's graphical representation is shown in figure 5.5 and 5.6 respectively. This experiment has been done by using default block size of the HDFS which is 128 MB for the Hadoop version 2.7.3.

Table 5.5: Execution time (in minutes) with varying sequence file size (102 MB to 1457 MB) in one, two, three and four nodes.

Sequence File Name	Size of sequence file	Number of sequences	One node (min)	Two node (min)	Three node (min)	Four node (min)
RefSeqS1.fasta	102 MB	69512	68	68	68	68
RefSeqS2.fasta	159 MB	108416	91	74	72	70
RefSeqS3.fasta	228 MB	163893	130	110	104	92
RefSeqS4.fasta	433 MB	210446	226	192	168	141
RefSeqS5.fasta	1024 MB	497683	468	326	223	193
RefSeqS6.fasta	1457 MB	708129	-	-	255	223

The result shows that for reference sequence file size smaller than HDFS block size (default block size 128 MB for Hadoop 2.7.3) file is not splitted so it is not distributed to all the nodes of the cluster so execution times for all the nodes are same. For reference sequence file size higher than block size. Reference sequence file is split and distributed no all the data nodes of the cluster. Due to this execution time decreases. The line graph plot of the above result is shown in below figure 5.5.

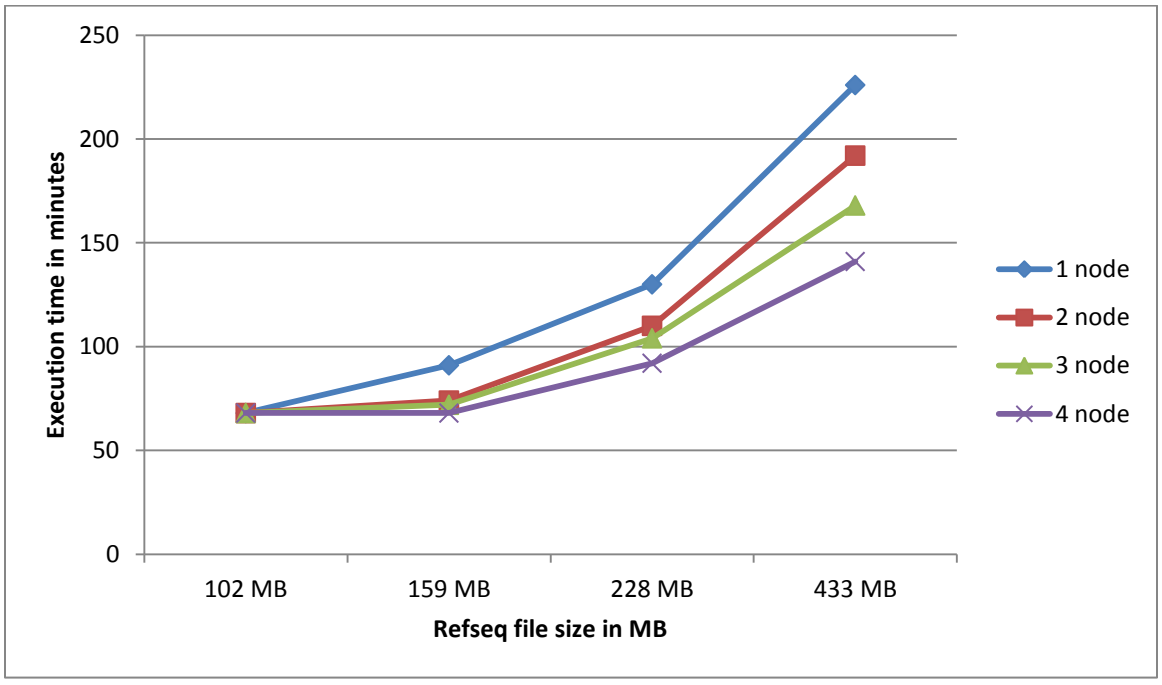


Figure 5.5: Execution time (in minutes) with varying sequence file size (102 MB to 1457 MB) in one, two, three and four nodes.

Table 5.6: Execution time (in minutes) with varying sequence file size (102 MB to 1457 MB) in four, six and eight nodes

Sequence File Name	Size of sequence file	Number of sequences	Four node (min)	Six node (min)	Eight node (min)
RefSeqS1.fasta	102 MB	69512	68	68	68
RefSeqS2.fasta	159 MB	108416	70	70	70
RefSeqS3.fasta	228 MB	163893	92	88	84
RefSeqS4.fasta	433 MB	210446	141	138	136
RefSeqS5.fasta	1024 MB	497683	193	178	168
RefSeqS6.fasta	1457 MB	708129	223	196	181

This result shows that execution time decreases while increasing the number of data nodes in the cluster. When the reference sequence is sufficiently large and number of splits is greater than the nodes of cluster then it is distributed to all the nodes and at that time actual parallelism is achieved. For reference sequence RefSeqS6.fasta number of splits = 1457 MB/128 MB ≈ 12. In this case all the nodes execute the split files in parallel and execution time decreases. The line graph of above result is plotted in figure 5.6.

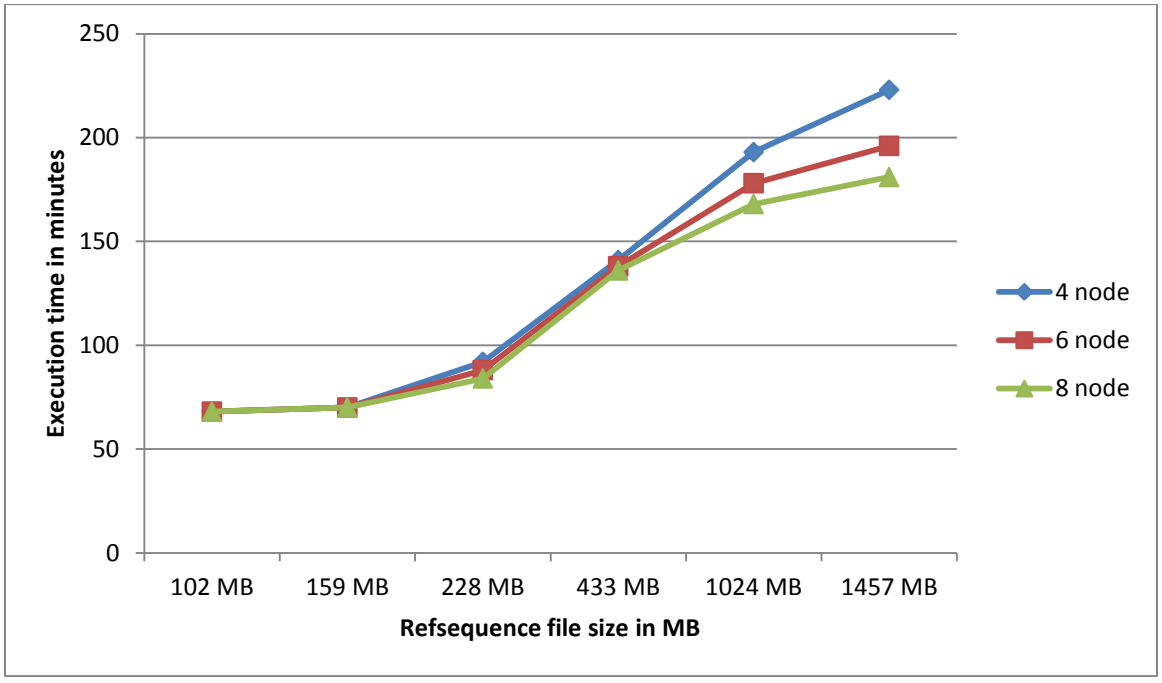


Figure 5.6: Execution time (in minutes) with varying sequence file size (102 MB to 1457 MB) in four, six and eight nodes

5.4 Sample output

Below are the sample outputs at different stages:

5.4.1 Job tracker window

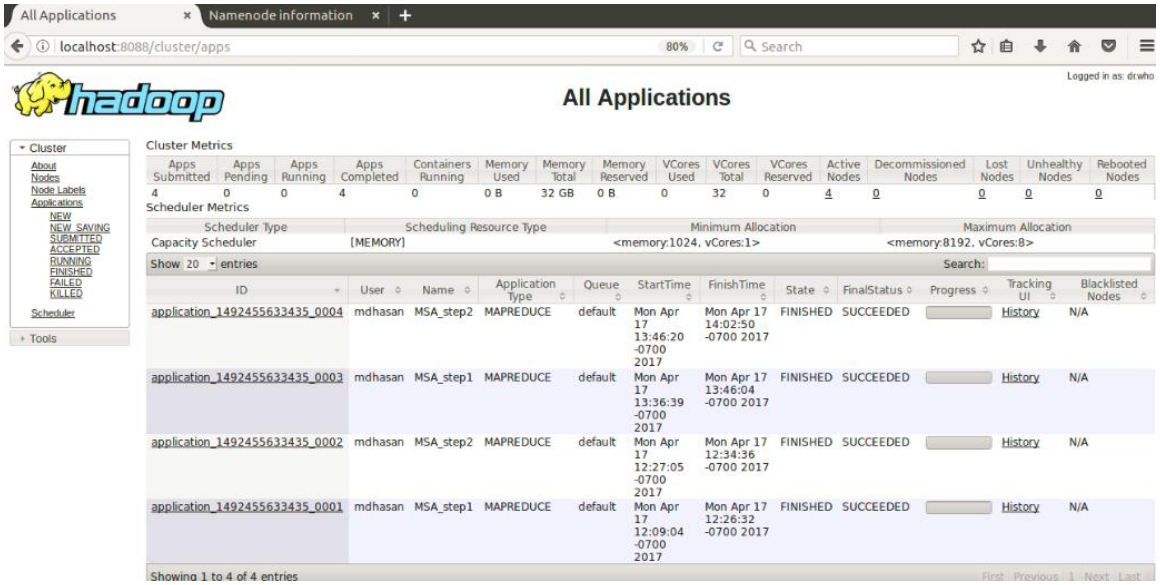


Figure 5.7: All application shown in job tracker window

The above figure shows the job tracker window that can be accessed from the master node. In this window status of the submitted and running jobs can be shown. It displays the status of running jobs and also shows the completed jobs.

5.4.2 First stage of Map Reduce

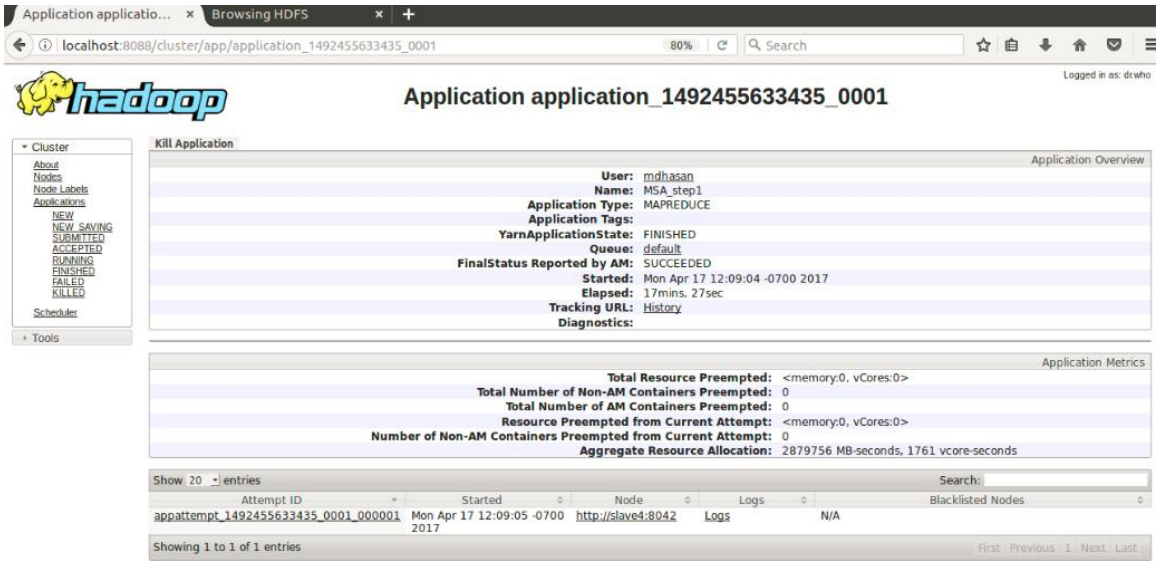


Figure 5.8: Status of first stage of map reduce

5.4.3 Second Stage of Map Reduce

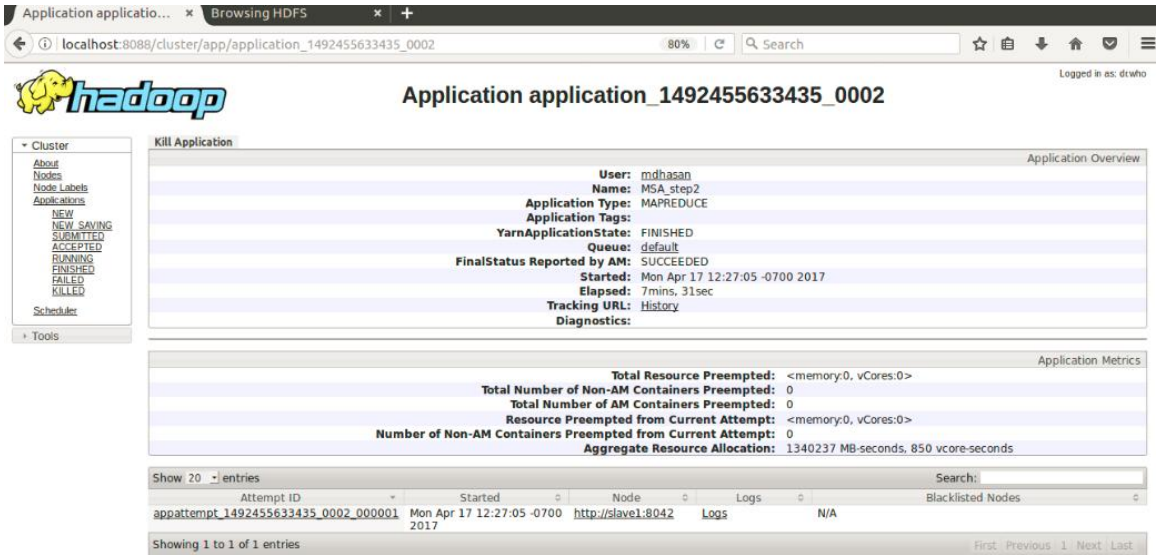


Figure 5.9: Status of second stage of map reduce

5.4.4 Datanode Information

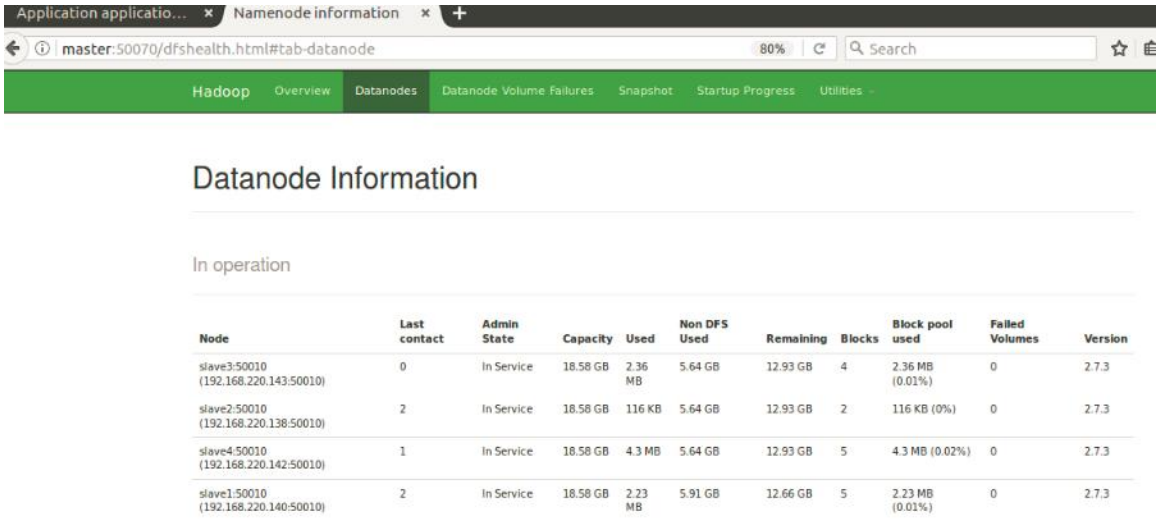


Figure 5.10: Datanode information accessed from master node while running jobs

5.4.5 Progress of Map Reduce

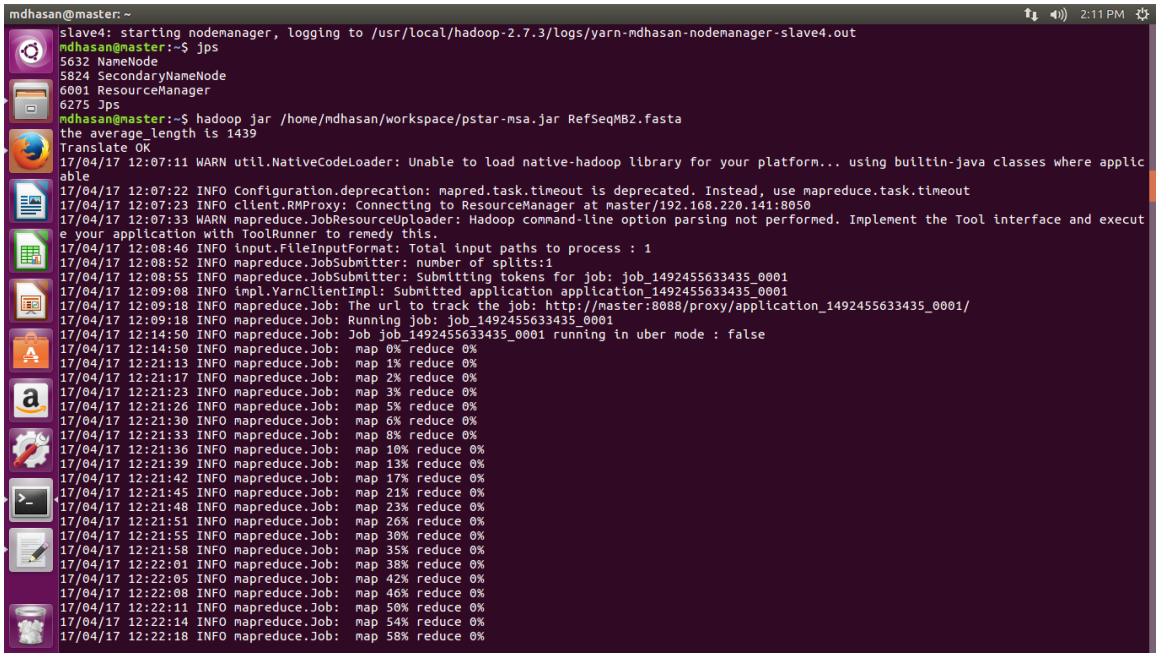


Figure 5.11: Progress of map reduce job shown in console

5.5 Result Summary

In this thesis a MapReduce model of star alignment algorithm for multiples sequence alignment has been developed and implemented using java as programming language and hadoop as MapReduce framework. This model has been tested using 8 physical node cluster and its result is verified using the result of sequential model of star alignment and with external tools ClustalW. The aligned sequences are same as that of the sequential algorithm, while comparing the result with ClustalW it seems little difference in number of match columns which are due to the use of different scoring matrixes. In this thesis identity matrix is used where as in ClustalW BLOSUM and PAM matrixes are used.

The experiments are done by using different reference sequences (number of reference sequences from 2 to 708129 and reference sequence file size from 4 KB to 1457 MB). For small number of sequences up to 250 for average sequence length 1442 bp and up to 30 for average length 8200 bp both the sequential and parallel algorithms are run and results have been compared. From these experiments it is shown that sequential model of start alignment algorithm is only suitable for few numbers of sequences. For 30 human genome sequences, sequential model takes execution time more than 3 hours where as map reduce model only takes 4.7 minutes. Due to high execution time of sequential model more than 369 KB sequence file size is not tested. Even in a single hadoop node execution time in quit low that compared to the sequential algorithm.

Another set of experiments have been done in reference sequence of file size from 102 MB to 1457 MB and nodes of cluster are varied from 2 nodes to 8 nodes. In this thesis default HDFS block size is used (128 MB for Hadoop 2.7.3). The large reference sequence file is splitted into file size equal or less than block size of the HDFS and distributed to all the data nodes that executes in parallel. The number of map function is equal to the number of splits and number of reducer is only one because all the intermediate results are combined for final result. From these experiments scalability of this model has been seen that when the number of nodes is increases then execution time decreases. It is shown from the experiment that speed up of 3 times is achieved from one node cluster to 8 physical nodes cluster for data sets greater than 1 GB.

CHAPTER 6: CONCLUSION AND FUTURE WORK

6.1 Conclusion

MSA is an important and fundamental tool in bioinformatics, especially for phylogenetic tree reconstruction. Biological sequences are aligned with each other vertically to show possible similarities or differences among these sequences. MSA is the process of aligning three or more nucleotides/amino-acids sequences at the same time. Dynamic programming algorithms like Needleman-Wunch and Smith-Waterman produce accurate alignments, but these algorithms are computation intensive and are limited to a small number of short sequences. It is a complete optimization problem where the time complexity of finding an optimal alignment raises exponentially when the number of sequences to align increases.

In this thesis work, a MapReduce model of start alignment algorithm for multiple sequence alignment has been developed. The dynamic nature of the model couples the data and computational parallelism of hadoop data grids by improving the speed of sequence alignment while maintaining the accuracy. To evaluate the performance of the model several datasets (file size from few KB to 1.4 GB) with different number of sequences (upto 708129 sequences) has been used. The experiments have been done from one node cluster to 8 node physical clusters.

From the experimental analysis it is shown that sequential version of star alignment is only suitable for less than 20 number of sequences. On the other hand, the result reveals that speed up of 3 times is achieved from one node cluster to 8 physical nodes cluster for data sets greater than 1 GB.

6.2 Limitation

In this thesis a MapReduce model of star alignment algorithm is developed and tested using different datasets. This model only align DNA sequences, it do not accept the protein sequences. Because DNA sequences have only four characters it’s easy to score

the alignments but protein sequences have twenty characters and scoring the alignments of protein sequences is more time consuming.

6.3 Future Work

This model can be extended for the alignment of protein sequences by using the protein substitution matrix and protein structure information. Substitution matrix is used to fill the score matrix at the time of alignment and after alignment used for finding the sum of pair score of aligned sequences and this is more time consuming and it can be reduced by using more physical nodes in the cluster.

The result of this model is verified by using the result of sequential version of the algorithm and also with the external tools ClustalW, which can't align sequences with file size more than 1 MB. Parallelizing the ClustalW in some way to fit with big data will be a research topic. The output of this model is aligned multiples sequences which can be used for phylogenetic tree reconstruction and further analysis of the genome.

In this thesis, only global alignments of sequences using Needleman-Wunsch is considered, Local alignment can also be done using Smith-Waterman algorithm and its results can be verified by using external tools like BLAST.

REFERENCES:

[1] A. A. K. Saad Khan Zahid, "A Novel Structure of Smith-Waterman Algorithm for Efficient Sequence Alignment," *Bioinformatics*, pp. 6-9, 2015.

[2] R. S. D. E. Farhana Naznin, "Progressive Alignment Method Using Genetic Algorithm for Multiple Sequence Alignment," *IEEE Transactions on Evolutionay Computation*, vol. 16, no. 5, pp. 615-630, 2012.

[3] W. Needleman, "A general method applicable to the search for similarities in the amino acid sequences of two proteins," *Molecular Biology*, vol. 48, no. 3, pp. 443-453, 1970.

[4] T. F. S. a. M. S. Waterman, "Identification of common molecular subsequences," *Molecular Biology*, vol. 147, no. 1, pp. 195-197, 1981.

[5] F. P. J. D. Thompson, "A comprehensive comparision of multple sequence alingnment programs," *Nucleic Acids Res.*, vol. 27, no. 13, pp. 2682-2690, 1999.

[6] w. A. K. H. S. Asril Adi S, "Parallelization of Star Alignment," *International Conference on Instrumentation, Communication, Information Technology and Biomedical Engineering*, pp. 167-171, 2013.

[7] T. White, "Hadoop: The definitive guide.," 2012.

[8] K. Li, "ClustalW-MPI: ClustalW analysis using distributed and parallel computing," *Bioinformatics*, vol. 19, no. 12, pp. 1585-1586.

[9] M. G. B. Dr G Sudha Sadasivam, "A Noval Approach to Multiple Sequence Alignment using Hadoop Data Grids," 2010.

[10] H.-I. H. S. Y. D. Wei Yi Liu, "Genomic Analysis with MapReduce," *2015 IEEE International Conference on Big Data*, pp. 1330-1335, 2015.

[11] H. Y. a. C. H. Crawford, "Big Data: Cloud Computing in Genomics Applications," *2015 IEE International Conference on Big Data*, pp. 2904-2906, 2015.

[12] L. I. S. Dr. Siddu P. Algur, "Parallelized Genomic Sequencing Model: A Big data approach for Bioinformatics application," *2015 International Conference on Applied and Theoretical Computing and Communucation Technology*, pp. 69-74, 2015.

[13] J. D. Watson, DNA: The Secret of Life, 2004.

[14] J. a. S. G. Dean, "MapReduce: simplified data processing on large clusters," *Communications of the ACM*, vol. 51, no. 1, pp. 107-113, 2008.

[15] W.-S. J. a. S.-F. Su, "Multiple Sequence Alignment using modified dynamic programming and particle swarm optimization," *Journal of the Chinese Institute of Engineers*, vol. 31, no. 4, pp. 659-673, 2008.