2015 IEEE International Parallel and Distributed Processing Symposium Workshops

Phylogenetic Analysis using MapReduce Programming Model

*Siddesh G M, K G Srinivasa**, Ishank Mishra, Abhinav Anurag, Eklavya Uppal,

Department of Information Science & Engineering
**Department of ComputerScience & Engineering*
M S Ramaiah Institute of Technology, Bangalore

Abstract- **Phylogenetic analysis has become essential part of research on the evolutionary tree of life. Distance-matrix methods of phylogenetic analysis explicitly rely on a measure of "genetic distance" between the sequences being classified, and therefore they require multiple sequence alignments as an input. Distance methods attempt to construct an all-to-all matrix from the sequence query set describing the distance between each sequence pair. Dynamic algorithms like Needleman-Wunsch algorithm (NWA) and Smith-Waterman algorithm (SWA) produce accurate alignments, but are computation intensive and are limited to the number and size of the sequences. The paper focuses towards optimizing phylogenetic analysis of large quantities of data using the hadoop Map/Reduce programming model. The proposed approach depends on NWA to produce sequence alignments and neighbor-joining methods, specifically UPGMA (Unweighted Pair Group Method with Arithmetic mean) to produce rooted trees. The experimental results demonstrate that proposed solution achieve significant improvements with respect to performance and throughput. The dynamic nature of the NWA coupled with data and computational parallelism of hadoop MapReduce programming model improves the throughput and accuracy of sequence alignment. Hence the proposed approach intends to carve out a new methodology towards optimizing phylogenetic analysis by achieving significant performance gain.**

Keywords: ***Sequence Alignment, Phylogenetic analysis, Needleman-Wunsch, UPGMA, Hadoop, MapReduce.***

I. INTRODUCTION

Phylogenetics is the study of evolutionary relationships among the genetically related group of species. Phylogenetic analysis is the means of understanding the relationships among different species during evolution. This study provides the hypothesis regarding the evolutionary history of taxonomic groups known as their *phylogeny*. This phylogeny is generally depicted as tree diagrams, branching diagrams to represent an evolutionary relationships among the species. Distance-matrix based methods of phylogenetic analysis depend on the measure of degree and distance among the sequential pairs of a species. This *genetic distance* between sequential pairs classified requires multiple sequence alignments as an input. Such distances are required to build the distance matrix which in turn depends on distance methods to construct phylogenetic tree. Distance methods aims towards the construction of a matrix from the sequence query set explaining the distance between each sequence pair. Distance matrix is an (m*m) matrix where m is the number of sequences. Rows in the matrix represent a single sequence and columns depict the distance between two sequences. Provided a set of m sequences, with distance n among the pair of sequences the distance matrix can be represented as (here is the n_{ij} is the distance between i^{th} and j^{th} sequences) [1] [2];

$$\begin{vmatrix} n_{11} \ldots n_{1m} \\ n_{12} \ldots\ldots n_{2m} \\ n_{m1} \ldots\ldots n_{mm} \end{vmatrix}$$

Dynamic programming algorithms like Needleman-Wunsch and Smith-Waterman produce accurate alignments. But these algorithms are computation intensive and are limited to a small number of short sequences [3]. One such idea is proposed in this paper for processing these enormous quantities of data is the usage of hadoop Map/Reduce programming model. The computation intensive algorithms required for phylogenetic analysis can be fitted in the Map/Reduce model and a time efficient approach can be carved out. A MapReduce program is composed of a Map() procedure that performs filtering and sorting (such as sorting students by first name into queues, one queue for each name) and a Reduce() procedure that performs a summary operation (such as counting the number of students in each queue, yielding name frequencies) [4] [5].

The *MapReduce System* (also called *infrastructure* or *framework*) orchestrates by marshalling the distributed servers, running the various tasks in parallel, managing all communications and data transfers between the various parts of the system, and providing for redundancy and fault tolerance. MapReduce is a framework for processing parallelizable problems across huge datasets using a large number of computers. Since its inception at Google, MapReduce has found many adopters. Among them, the prominent one is the apache software foundation, which has developed an open source version of the MapReduce framework called Hadoop. Hadoop boasts of a number of large web-based corporate like Yahoo, Facebook, Amazon, etc., that use it for various kinds of data-warehousing purposes [6][7][8][9][10]. This paper proposes a optimized solution for analysing the phylogenetics to produce a phylogram using MapReduce programming model.

II. RELATED WORK

Next generation sequencing has led to the generation of billions of sequence data, making it increasingly infeasible for sequence alignment to be performed on standalone machines. Hence algorithms capable of running in multi-node clusters and deployable on cloud have been the subject of investigation. The performance improvement offered by these algorithms is not usually predictable since the cloud implementation has many overheads which may sometimes negatively affect the performance. A detailed study on the performance of a canonical MapReduce implementation of sequence alignment algorithm known as CloudBurst is proposed in [11]. CloudBurst is highly sensitive short read mapping with MapReduce. They have found that the performance of sequence alignment algorithms depend upon the configuration of the cluster in which it is executed. Running the algorithm with a large input in a cloud computing environment was more efficient than running in a single node. [12] presented a report on application of MapReduce, using its open source implementation Hadoop, to two relevant algorithms: BLAST (Basic Local Alignment and Search Tool) and GSEA (Gene Set Enrichment Analysis). They found their results to be promising and indicated that the framework could have a wide range of bioinformatics applications while maintaining good computational efficiency, scalability and ease of maintenance. Analysis tools for next generation DNA sequencing data using a structured programming framework called the Genome Analysis Toolkit (GATK) is presented in [13]. The framework was used to build genome analysis tools easily and efficiently by using the functional programming paradigm called MapReduce. The report also describes hadoop, which is a software scheme for cloud computing that provides MapReduce functionality for computing clusters and Hadoop Distributed File System for storing large data. The report illustrates the use of MapReduce in Hadoop by running the simple WordCount class and an application called Hadoop-bam. As a part of the results of the report, she describe the execution of a simple custom built example with GATK. A novel approach to Multiple Sequence Alignment (MSA) using Hadoop framework was presented in [14]. This methodology is uses dynamic programming. They achieved parallelism in conducting MSA by using multiple levels of data processing. Their proposed method of MSA improved on the computation time and also maintained the accuracy. They were also able to maintain the accuracy of sequence alignments.

The observations from the various solutions related to analyzing phylogenetics using MapReduce model is that their focus is to achieve either data or computational parallelism. Further the solution does not consider dynamicity of the algorithm to optimize the phylogenetic analysis. The propose work is a time efficient approach to phylogenetic analysis that produces a phylogram (phylogenetic tree or evolutionary tree). It will be using NWA to produce sequence alignments. Further it will be using the neighbor-joining methods, specifically the UPGMA method to produce rooted trees. The proposed methodology will be exploiting the MapReduce programming model using the hadoop framework. The proposed approach is combines data and computational parallelism of hadoop data grids by optimizing the throughput, accuracy and response time of analyzing the phylogenetics.

III. PROPOSED SYSTEM

With the enormous growth in bio-information, there is a corresponding need for tools that enable fast and efficient alignment job of sequences. Dynamic programming algorithms like NWA and SWA produce accurate alignments. But these algorithms are computation intensive and are limited to a small number of short sequences. Hence the concurrent execution of these algorithms will greatly simplify the complexity of the alignment [9][10][15]. The core objective is the design and implementation of parallel approach to Phylogenetic analysis using Hadoop Data Clusters to overcome the limitations of original NWA by dividing the set of sequences into blocks and processing the blocks in parallel. Also there is a need to carry out the performance analysis for different sets of input as well as for different number of nodes in the cluster.

```
Input: Set of Sequences
Output: Dendrogram representing clustering of sequences

MapReduce Stage 1:
Step 1: Read the input set of Records
Step 2: Identify the set of Sequences using logical delimiter
Step 3: Identify the description tags of the Sequences
Step 4: Create a custom record for each of the input  sequence
Step 5: Write the record in an intermediate output

MapReduce Stage 2:
Step 1: Read the records from the output of earlier Mapreduce Stage
Step 2: Form two identical sets of Records using the same   set of
          records
Step 3: Make a Cartesian product of the two set of records
Step 4: Create a custom record for each pair in the Cartesian product
Step 5: Write the record in an intermediate output

MapReduce Stage 3:
Step 1: Read the records from the output of earlier Mapreduce Stage
Step 2: Identify the two set of sequences and their respective tags
Step 3: Align the two sequence using Needleman-Wunsch Algorithm
Step 4: Create a custom record using the pair of tags and alignment
          scores
Step 5: Write the record in an intermediate output

Hierarchical clustering using UPGMA:
Step 1: Read the records from the output of earlier Mapreduce Stage
Step 2: Perform Hierarchical clustering using UPGMA
Step 3: Create a dendrogram using the clustered results
```

Algorithm 1: Proposed algorithm for phylogentic analysis using MapReduce

The proposed system uses a parallel approach to Phylogenetic analysis using Hadoop Data Clusters to overcome the limitations of original NWS by dividing the set of sequences into blocks and processing the blocks in parallel. The fig.1 shows the various stages of

MapReduce carried out in the process of making Phylogenetic analysis. Further, it also shows how the dendrogram would look like. The whole process is carried out in Three MapReduce stages followed by hierarchical clustering using UPGMA which produces a dendrogram. The process of making phylogenetic analysis requires a certain amount of data pre-processing. Hence even before the MapReduce job of sequence alignment is carried out using NWA; two stages of MapReduce are carried out in order to pre-process the data. The system takes the input data in FASTA format. Then a MapReduce stage is run for data pre-processing. What follows is another stage of MapReduce which produces a cartesian product of the records generated by the earlier MapReduce phase. The output of this stage is fed into the NWA which makes the corresponding sequence alignments. The result of the third stage of MapReduce is fed to the UPGMA clustering algorithm which produces a phylogenetic tree in Newick format. This tree in Newick format [16] then used to build a dendrogram using the D3.js library of JavaScript. Algorithm 1 explains the process of phylogentic analysis in an efficient and robust manner. It highlights the three MapReduce stages followed by hierarchical clustering using UPGMA to produce a dendrogram.

IV. IMPLEMENTATION

The fig.2 shows the activity diagram for the system. The input for the system is a set of sequences in FASTA format. The input sequences are split into blocks and submitted for a MapReduce job. The Job tracker is responsible for assigning tasks to the slave nodes which have a task tracker running. The slave node process the block by producing a sequence alignment. The results are then written to the HDFS.

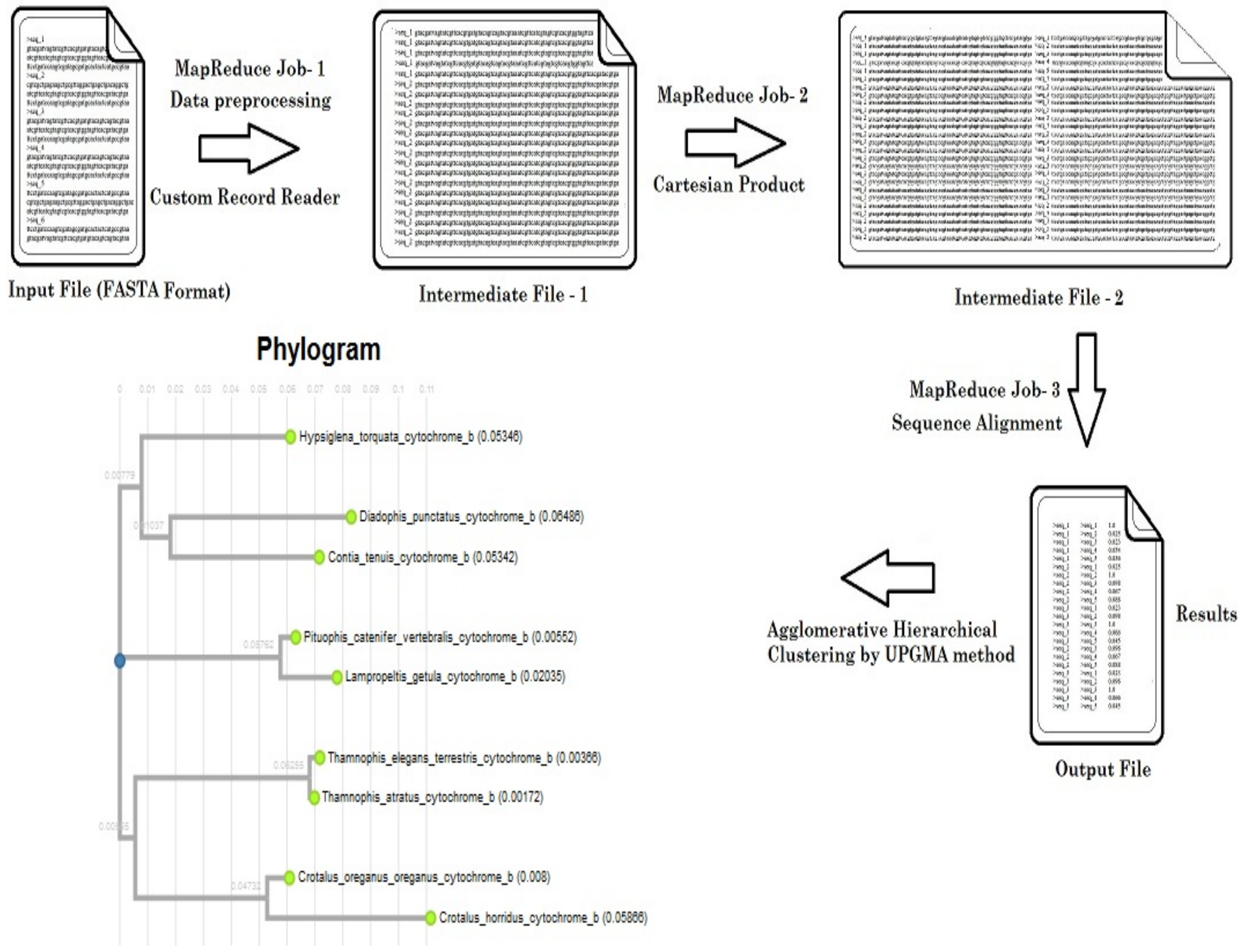

MapReduce Job- 1
Data preprocessing
Custom Record Reader
Input File (FASTA Format)
Intermediate File - 1
MapReduce Job- 2
Cartesian Product
Intermediate File - 2
MapReduce Job- 3
Sequence Alignment
Output File
Results
Agglomerative Hierarchical Clustering by UPGMA method
Phylogram
Hypsiglena_torquata_cytochrome_b (0.05346)
Diadophis_punctatus_cytochrome_b (0.06486)
Contia_tenuis_cytochrome_b (0.05342)
Pituophis_catenifer_vertebralis_cytochrome_b (0.00552)
Lampropeltis_getula_cytochrome_b (0.02035)
Thamnophis_elegans_terrestris_cytochrome_b (0.00366)
Thamnophis_atratus_cytochrome_b (0.00172)
Crotalus_oreganus_oreganus_cytochrome_b (0.008)
Crotalus_horridus_cytochrome_b (0.05866)

Fig. 1. Stages in the proposed system in optimizing phylogenetic analysis

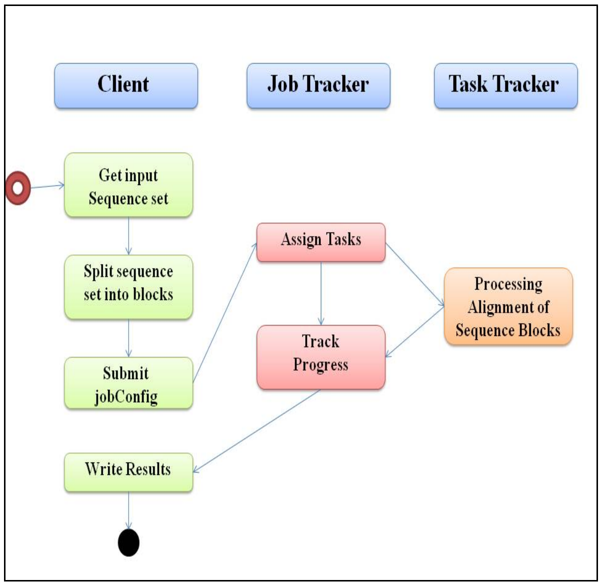


Fig.2. Contextual Activity Diagram for the proposed system

InputFormat

The proposed system uses the input data in FASTA format. It is a text-based sequential representation of peptides or nucleotides, which are represented using single-letter codes. Further the format permits for sequence names and comments to follow the sequences. The FASTA format is has become a defacto standard in the area of bioinformatics. A FASTA sequence format begins with a single-line depiction, followed by stripes of sequence data. The description strip is differentiated from the sequence data strip by a greater-than symbol (">") in the first column. The identifier of the sequence is the word followed by ">" symbol and rest of the line is the description. Between the ">" and the first character of the identifier there should not be any spacing. It is suggested that all strips of text be shorter than 80 characters. The sequence terminates if another strip starting with a ">" appears; which signifies the start of another sequence [2]. A simple example of one sequence in FASTA format is shown in the fig.3.

```
>gi|5524211|gb|AAD44166.1| cytochrome b [Elephas maximus
maximus]

LCLYTHIGRNIYYGSYLYSETWNTGIMLLLITMATAFMGYVLP
WGQMSFWGATVITNLFSAIPYIGTNLVEWIWGGFSVDKATLN
RFFAFHFILPFTMVALAGVHLTFLHETGSNNPLGLTSDSDKIPF
HPYYTIKDFLGLLILILLLLLLALLSPDMLGDPDNHMPADPLNT
PLHIKPEWYFLFAYAILRSVPNKLGGVLALFLSIVILGLMPFLHT
SKHRSMMLRPLSQALFWTLTMDLLTLTWIGSQPVEYPYTIIGQ
MASILYFSIILAFLPIAGXIENY
```

Fig.3. Input sequence data set in FASTA format

Output Format

The output format used is of custom type. It consists of records separated by newline character. Each record has three fields. The first and second fields indicate the names of sequences that are aligned. The third field indicates the final score of the aligned sequences. A simple example of the custom output format is shown in the fig.4.

```
>sequence1        >sequence2        0.67843
>sequence1        >sequence3        0.57852
>sequence1        >sequence4        0.44349
>sequence1        >sequence5        0.98640
>sequence1        >sequence6        0.14802
```

Fig.4. Custom output format

To create a dendrogram the output of this format is provided as an input for the UPGMA clustering program. This in turn produces a tree in Newick format. Newick tree format (or Newick notation or New Hampshire tree format) is a way of representing graph-theoretical trees with edge lengths using parentheses and commas as seen in fig.5.

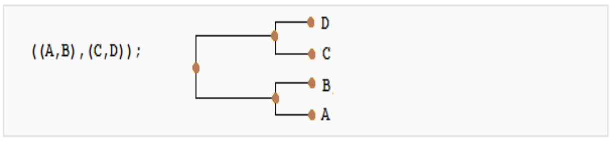


Fig.5. *Tree in Newick format and its corresponding dendrogram*

MapReduce Job -1: Data Pre-processing

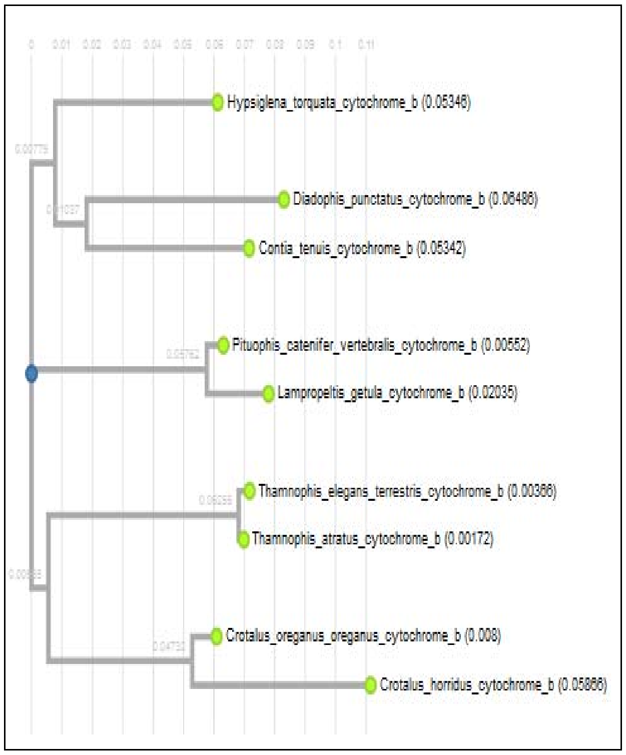
This is first of the three stages of MapReduces that are to be implemented. This stage takes the input in FASTA format which a text based format for storing sequences. A sequence in FASTA format begins with a single-line description, followed by lines of sequence data. The output of this stage is a customized format suitable for the next MapReduce stage. To start with the input set of Records are read. Then the set of Sequences are identified using logical delimiter (which in this case is '>' as FASTA format is being used). Then the description tags of the Sequences are identified. A custom record for each of the input sequence is created. It starts with the description tag followed by a tab and the input sequences stitched together. Finally the records are written in an intermediate output.

MapReduce Job -2: Cartesian product

This MapReduce stage takes the input from the earlier MapReduce phase which was responsible for data pre-processing. The stage is responsible for generating the Cartesian product of the input sequence which will help in making sequence alignments using NWA in the next MapReduce stage. First the records from the output of earlier MapReduce Stage i.e, data pre-processing stage is read. Then two identical sets of Records using the same set of records are formed. Cartesian product of the two set of records is carried out. Then a custom record for each pair in the cartesian product s created. Finally the records are written in an intermediate output. The record has the description tag of a record from the first set followed by its sequence which is followed by the description tag of a record from second set and its corresponding sequence. Each field in the record are tab separated.

MapReduce Job -3: Sequence Alignment

The stage takes the input from the second MapReduce stage which produces the Cartesian product of input sequences. This stage unlike the other stages does not require a custom Input or Output format. It uses the TextInputFormat class for both Map and Reduce phases. The Mapper class does the very basic job of tokenization. It read each record from the input file by using newline character as a delimiter. It then just tunnels the record to the reduce phase. The Reducer class takes each record and breaks it into a string array. The records contain fields which are tab separated. Since the position of the corresponding sequences is known, the two sequences are used for making the sequence alignment. To start with the records from the output of earlier Mapreduce Stage are read. Then the two set of sequences and their respective tags are identified. The two identified sequences are aligned using NWA. Then a custom record is created using the pair of tags and alignment scores. Finally the records are written in an intermediate output.



0
0.01
0.02
0.03
0.04
0.05
0.06
0.07
0.08
0.09
0.1
0.11
Hypsiglena_torquata_cytochrome_b (0.05346)
0.00779
Diadophis_punctatus_cytochrome_b (0.06486)
Contia_tenuis_cytochrome_b (0.05342)
0.05762
Pituophis_catenifer_vertebralis_cytochrome_b (0.00552)
Lampropeltis_getula_cytochrome_b (0.02035)
0.06255
Thamnophis_elegans_terrestris_cytochrome_b (0.00366)
Thamnophis_atratus_cytochrome_b (0.00172)
0.04733
Crotalus_oreganus_oreganus_cytochrome_b (0.008)
Crotalus_horridus_cytochrome_b (0.05866)

Fig 6. Dendrograms representing the clustered results

Hierarchical clustering using UPGMA

The output from the third MapReduce stage produces an output which could be interpreted as a distance matrix. The distance matrix is necessary for the clustering algorithm. To start with the records from the output of earlier Mapreduce Stage i.e. Sequence alignment stage is read. Then the corresponding distance matrix is constructed. Hierarchical clustering using UPGMA is performed. Finally a dendrogram is created using the clustered results.The external interface is created using D3.js (D3 for Data-Driven Documents). The D3.js is used to build a dendrogram which is a form of phylogenetic tree. The UPGMA clustering algorithm produces a tree in the Newick format. Fig. 6 represents the dendrograms representing the clustered results for the given input.

Implementation of the algorithm an object of UPGMA class is created. The class has three methods. First, clusterTwoNode() which combines those two nodes that have the smallest distance. Second, clustertwoMatrix() which regenerates the distance matrix after two closest nodes are clustered. Third, checkMatrixToTerminateIterate() which determines when to terminate the clustering process. Algorithm 2 depicts the process of Hierarchical clustering using UPGMA.

```
Input: Set of Sequences
Output: Dendrogram representing clustering of sequences

UPGMA upgma = new UPGMA();
while(!upgma.checkMatrixToTerminateIterate(matrix)){
point point = MatrixAction.findPointMinValueInMatrix(matrix);
nodeList=
upgma.clusterTwoNode(nodeList,matrix[point.getRow()][point.getCol(
                                        point.getRow(), point.getCol());
matrix = upgma.clusterTwoMatrix(matrix, point.getRow(),
point.getCol());
for(int i=0;i<nodeList.length;i++)
            System.out.println(nodeList[i].getNameNode());
```

Algorithm 2: *Hierarchical clustering using UPGMA*

It is made sure that as the nodes are clustered they are in Newick format. Algorithm 3 shows how to build the Newick format which is used to build dendrograms using D3.js library.

```
Node node =
new
Node("("+sequences[firstNode].getNameNode()+":"+distance/2+ ","
+  sequences[secondNode].getNameNode()+":"+distance/2+")");
```

Algorithm 3: *Node in Newick format*

V. Performance Analysis

The programs were run on single node cluster, two node cluster as well as four node cluster. Various metrics such as run time, number of sequences, input size and output sizes were used to test the performance of the system.

A. *Execution Time with varying load*

A total of five sequences of varying lengths were taken for sequence alignment. The first one comprised of 40 sequences and the ones that followed had sequences in increasing order by 20 sequences. The sequences were first run on a single node cluster, followed by a two node cluster and four node cluster. Then a multi-node cluster was setup and the same set of sequences were aligned and the corresponding results were noted. The results are depicted in fig.7, it can be noted that initially when the number of sequences were less (in this case 40) the difference in time taken to align the sequences were insignificant. The expected reason was the network overhead. But as the number of sequences were increased, a significant improvement was noted. As we see in the graph increasing the number of nodes does not significantly improve the performance significantly because of network overheads, communication overhead incurred during execution.

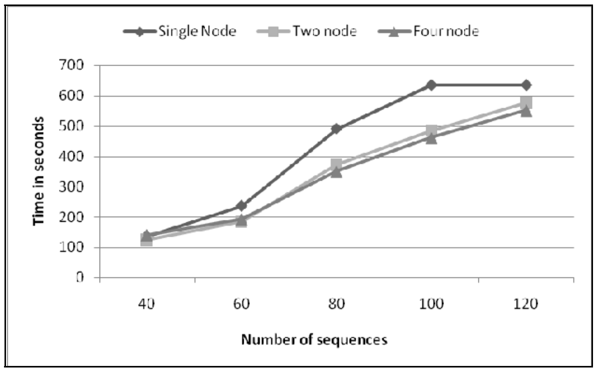


Fig.7. Execution time of sequence alignments

B. *Execution time in different MapReduce phases of proposed system*

The set of sequences in proposed system are to be aligned go through three MapReduce stages. The first two stages custom record reader and cartesian product have almost similar time as their input sizes are small. But the third stage sequence alignment requires a much larger input and hence a large amount of time is required for the third stage. Fig.8 shows the comparison of execution times of different MapReduce phases in the proposed system shown in fig.1. The reason why the third stage requires a large amount of time is because the input size grows by a quadratic time. This is due to the second MapReduce stage which makes a cartesian product of the input sequence set. The time taken for this Cartesian product of input sequences is due to time needed by NW algorithm.

C. *Throughput of sequences aligned time per sequence*

The throughput here is number of sequence alignments made per second. As usual a total of five sequence sets were taken that have increasing number of sequences. Throughput was calculated in case of both single node and two node cluster. From the fig.9 it can be observed that the throughput has gradually increased as size of sequence set increases. Also the difference between the throughputs of a single node cluster and two-node cluster also increase gradually. The difference was initially low due the parallelizing overheads and network overheads.

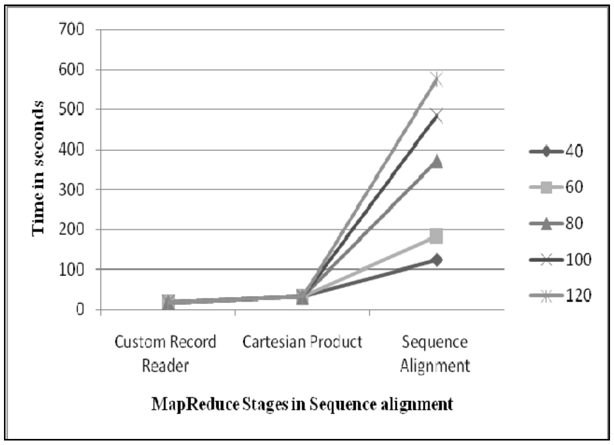


Fig. 8. Comparison of MapReduce phases in proposed system

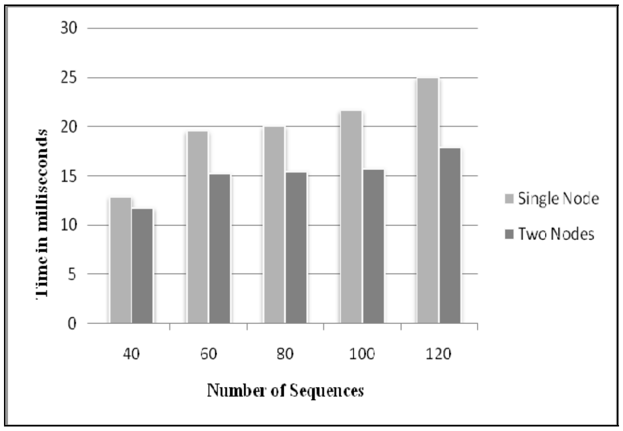


Fig. 9. Throughput of sequences aligned time per sequence

VI. Conclusions

Dynamic programming algorithms like NWA and SWA produce accurate alignments, but these algorithms are computation intensive and are limited to a small number of short sequences. The aim of the proposed solution is the design and implementation of parallel approach to phylogenetic analysis using hadoop data clusters to overcome the limitations of original NWA by dividing the set of sequences into blocks and processing the blocks in parallel. Further, there is a need to carry out the performance analysis for different sets of input as well as for different number of nodes in the cluster. The proposed solution is a time efficient approach to phylogenetic analysis that produces a phylogram. It used the neighbor-joining methods, specifically the UPGMA method to produce rooted trees. It also used the NWA to produce sequence alignments. The proposed methodology also exploited the Map/Reduce model using the hadoop framework. The dynamic nature of the proposed solution couples the data and computational parallelism of hadoop data grids by improving the accuracy and speed of sequence alignment. Further due to the scalability of hadoop framework, the proposed method for phylogenetic analysis is also highly suited for large scale problems.

References

[1]Baxevanis, Andreas D., and BF Francis Ouellette. *Bioinformatics: a practical guide to the analysis of genes and proteins*. Vol. 43. John Wiley & Sons, 2004.

[2]National Center for Biotechnology Information (NCBI), http://www.ncbi.nlm.nih.gov/

[3] Needleman, Saul B., and Christian D. Wunsch. "A general method applicable to the search for similarities in the amino acid sequence of two proteins." *Journal of molecular biology* 48.3 (1970): 443-453.

[4] Dean, Jeffrey, and Sanjay Ghemawat. "MapReduce: simplified data processing on large clusters." *Communications of the ACM* 51.1 (2008): 107-113.

[5] Dean, Jeffrey, and Sanjay Ghemawat. "MapReduce: a flexible data processing tool." *Communications of the ACM* 53.1 (2010): 72-77.

[6]*Apache Hadoop Mapreduce*, http://hadoop.apache.org/Mapreduce

[7] White, Tom. *Hadoop: The definitive guide*. " O'Reilly Media, Inc.", 2012.

[8]*HDFS,* http://Hadoop.apache.org/hdfs

[9] Ekanayake, Jaliya, Shrideep Pallickara, and Geoffrey Fox. "Mapreduce for data intensive scientific analyses." *eScience, 2008. eScience'08. IEEE Fourth International Conference on*. IEEE, 2008.

[10] Taylor, Ronald C. "An overview of the Hadoop/MapReduce/HBase framework and its current applications in bioinformatics." *BMC bioinformatics* 11.Suppl 12 (2010): S1.

[11] Schatz, Michael C. "CloudBurst: highly sensitive read mapping with MapReduce." *Bioinformatics* 25.11 (2009): 1363-1369.

[12] Gaggero, Massimo, et al. "Parallelizing bioinformatics applications with MapReduce." *Cloud Computing and Its Applications* (2008): 22-23.

[13] Maharjan, Merina. "Genome Analysis with MapReduce". *June* 15 (2011): 3-4.

[14] Sadasivam, G. Sudha, and G. Baktavatchalam. "A novel approach to multiple sequence alignment using hadoop data grids." *Proceedings of the 2010 Workshop on Massive Data Analytics on the Cloud*. ACM, 2010.

[15] Batzer, Mark A., and Prescott L. Deininger. "Alu repeats and human genomic diversity." *Nature Reviews Genetics* 3.5 (2002): 370-379.

[16] Trooskens, Geert, et al. "Phylogenetic trees: visualizing, customizing and detecting incongruence." *Bioinformatics* 21.19 (2005): 3801-3802.