Volume 5, Issue 6, June 2015 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

An Improved Genetic Algorithm on MapReduce Framework Using Hadoop Cluster for DNA Sequencing

Jaideep Singh*, Arun Solanki
School of Information and Communication Technology
Gautam Buddha University, Uttar Pradesh, India

Abstract— Genetic Algorithms are one of the most popular among the Evolutionary Algorithms based on the theory of evolution. In this paper the Hybrid MapReduce Algorithm is proposed to solve the DNA Sequencing problem. Given the basic fragment of DNA a sequence is generated and matched against the original sequence. A simple and efficient heuristic method is used to perform distributed computations on a cluster of commodity computers using MapReduce framework. A simple architecture is developed and the results are compared with the existing Genetic Algorithm.

Keywords— evolutionary algorithms; genetic algorithm; hadoop cluster; mapreduce fraamework; DNA sequences

I. INTRODUCTION

There is no upper limit on the size of instances that have to be handled by an exact algorithm for a NP-Hard problem or if the known best exact algorithm is unable to handle the intended instances in a reasonable time, the search for the optimum has to be abandoned. Approximation techniques are recommended in this case.

The evolutionary algorithm (EA) [1] are nature based meta-heuristic optimization algorithm. It uses reproduction, mutation, recombination and selection mechanism which are inspired from biological evolution. The role of individuals in a population, and the fitness function determines the quality of the solutions in optimization problems.

The most prominent among the evolutionary algorithms is genetic algorithm. Genetic algorithm was proposed by [2]. Later genetic algorithm was used to solve various kinds of problems including search and optimization. The main reason for the success of genetic algorithm is its robustness and its ability to explore several possible areas of search space at the same time. Genetic algorithm provides implicit as well as explicit parallelism.

MapReduce [3] is a distributed framework for processing and generating large data sets with a parallel, distributed algorithm on a cluster. A MapReduce Algorithm consists of a Map() functions and Reduce() functions [4] which are defined by the user for specific purpose. The open source implementation of the MapReduce framework[5] is Apache Hadoop that enables the distributed processing of large data sets across clusters of commodity hardware. In this paper the DNA sequencing problem[6] is solved using genetic algorithm along with the MapReduce framework which provide distributed ecosystem.

II. RELATED WORK

An individual solution in genetic algorithm is represented by a set of parameters [7][8]. These parameters are considered as genes that in turn form a chromosome or individual, when structured as a string. The genetic algorithm begins with a population of such chromosomes. Normally all the individuals in the population are randomly generated to have an unbiased representation of the search space. However, if it is know beforehand or otherwise some limited regions of the search space, which contain good potential solutions, then these can be used to start the genetic algorithm so that the available information about the fitness landscape is exploited to get a better start. As a result of reproduction, subsequent population is generated; this is called next generation of the population. The new generation is now evaluated. Termination criterion is tested again and if it is satisfied the algorithm is terminated at once else the genetic algorithm will continue till the termination condition is satisfied.

MapReduce is a programming model [9][10] 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. Many real world tasks are expressible in this model. Programs written in this functional style are automatically parallelized and executed on a large cluster of commodity machines [11][12]. 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.

III. ARCHITECTURE OF THE SYSTEM

The architecture is a planned structure of the system which allows the user to understand the functionality of the system. It represents different modules which work together to produce the desired result. The architecture for the problem of DNA Sequencing consists of the following modules namely:-

- Interface Module
- Hadoop Cluster Module
- Hadoop Distributed File System Module

The modules are shown in detail as follows:-

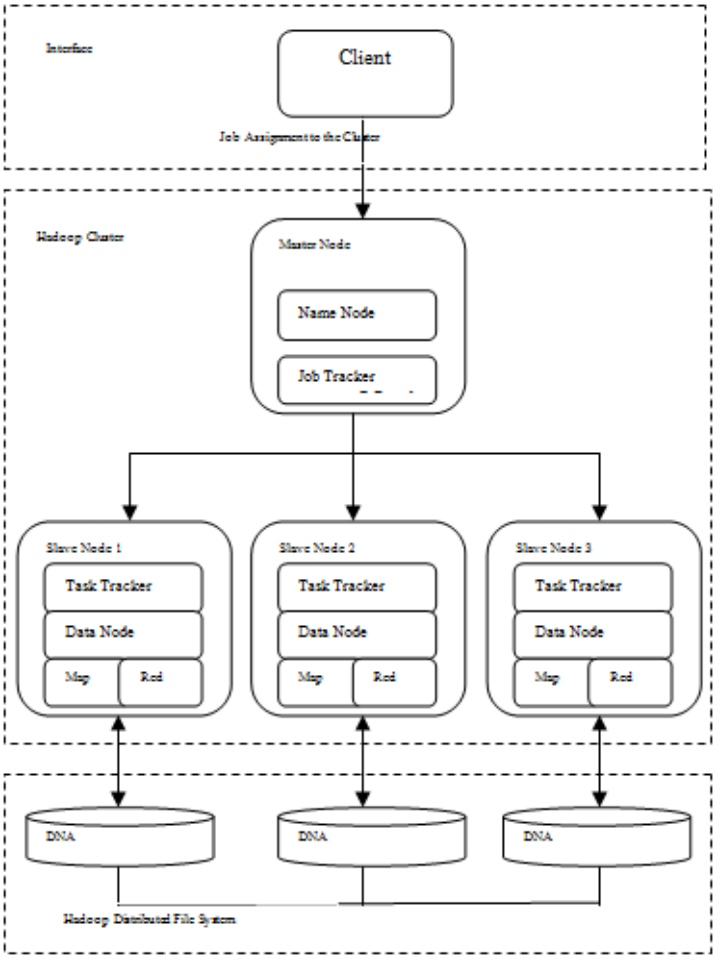

Fig. 1 Architecture of the System

A. Interface Module

It consists of the node from the cluster that provides the input to the system. From this node user can assign the job to the cluster. A job is the set of inputs provided to the system in order to execute the algorithm. The inputs may be the number of nodes in the cluster, assignment of the master node, number of iterations performed by the algorithm, the termination conditions of the algorithm based on these input data the output may vary.

B. Hadoop Cluster Module

Apache Hadoop is the open source implementation of the MapReduce framework which is used for the data intensive applications and the application that involve the parallel processing on the real time data as well. This software enables us to use the commodity hardware to perform large computations which are economic and reliable.
Hadoop provides the computing solutions that are:

- Scalable
- Flexible
- Fault tolerant
- Cost effective

1) Master node:

Mater node in the Hadoop ecosystem provides different storage and processing management functionalities, it also keep track of the redundant data to avoid one points failure. A master node consists of a name-node which manages the Distributed Hadoop File System and also creates the check points for the back-up, a job tracker handles resource management and scheduling.

2) Slave node:

The slave node in the Hadoop ecosystem executes the task assigned by the master node. They are present where the data proceeding is required. It consists of the data node which enables the name node to store block on the slave node, the

Task Tracker that manages the individual Map and Reduce functions executing on the slave node. The Map and Reduce function which are defined by the user according to the problem and are executed on the different slave nodes.

3) Hadoop distributed file system:

The Hadoop Distributed File System (HDFS) is a distributed file system designed to run on commodity hardware. The data present in the HDFS is the DNA structure which is used to match against the evolved DNA structure generated by the algorithm.

IV. PROCESS FLOW OF THE SYSTEM

The process flow of the system describe flow of data within the system and also how the data is processed by different modules present in the system. The job is assigned to the master node by the user .Then the master node start the service and distribute the job to the slave nodes in the system. On the slave node map() and reduce() functions are present which divide the task into fragments and then rejoin them as the task completed. The completed task is send back to the master node and then to the client as shown in Fig. 2.

The process flow of the proposed system is as follows:-

Step 1:-A job is assigned to the master node by the user.
Step 2:-Master distribute the tasks among the system in the node(slave nodes).
Step 3:-Slave call the map() function which divide the task into small fragments.
Step 4:-The result from the map() function are combined by the reduce() function.
Step 5:-The output is returned to the salve node after completion of the task.
Step 6:-The slave node provide the status of the job to the master node.
Step 7:-The final result obtained from the different nodes is provided to the user.

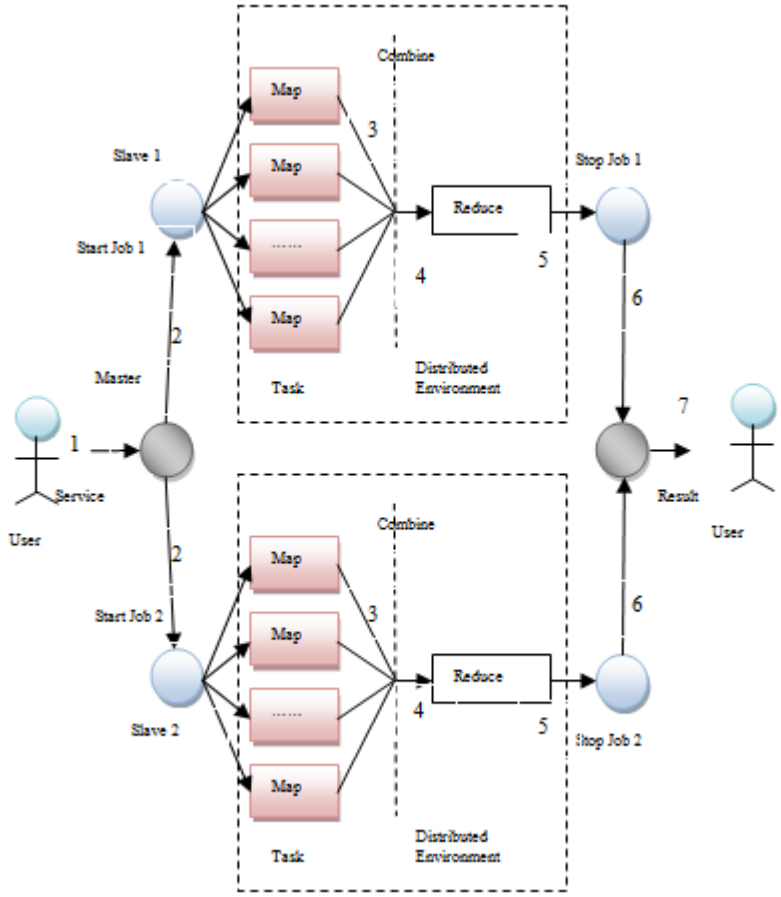

Fig. 2 Process Flow of the System

V. DNA SEQUENCING PROBLEM

DNA can be defined as a string of symbols drawn from the set S = {A,C,T,G}, where A represents Adenine, C represents Cytosine, G represents Guanine, and T represents Thymine. Each symbol is known as a nucleotide.A sequence or strand is a string of larger number of nucleotides. Given the random fragments of any length from the set S, a sequence has to be generated which is the closest match to the original sequence.

The main parameter which affect the performance of the algorithm are:-

A. Population Model

Generational population method is used is used in the algorithm. In this method the whole population or the individuals with the best fitness value replaces the existing pool of population.

B. Representation of the individual

The individuals are represented as the string from the set S={A,T,C,G}. The individuals are generated randomly to form a fixed length string and then these strings are represented in the secondary form that is the integer representation.

C. Fitness Function

The fitness is calculated by matching the generated Sequence with the original sequence. The fitness is calculated by one to one matching of the chromosomes.

D. Mutation

The mutation is performed by randomly replacing the value of one place of the chromosome from the set S

E. Crossover and Selection

It is used to generate off-springs from the parents and here one point crossover is used. The chromosomes with the high fitness values are selected.

VI. PROPOSED MAPREDUCE ALGORITHM

The MapReduce Programming model provides the distributed ecosystem for the development of the proposed system. The MapReduce program consists of following main classes:-s

- Mapper Class
- Reducer Class

The Algorithm for the Mapper Class is described as follows:-

Mapper Class
BEGIN
Step1:-*Run Job1: Creates initial population and assign key and value*
Step2:-*for max generation number*
Run Job i: evolve population
End for
Step3:-*Run job N: write resulting populations in a readable format to HDFS*
END

The mapper class initialize the population and assign the key and value to each individual in the population. It takes the input file of the original sequence. The population is now evolved based on the operator described in the Reducer class. Then the Reducer class is called and the population is evolved. The results generated are write back into the HDFS.

The Algorithm for the Reducer Class:-

Reducer Class
BEGIN
Step1:-*Receive individual with same key*
Step:-*for evolution number*
for population size
calculate fitness
apply selection operation
apply one point crossover
add new individual to population
end for
apply mutation to new population
end for
Step4:-*write to HDFS*
END

The reducer class perform the recombination, mutation, selection operations along with the fitness values are also calculated in this class. The individuals with the same key are put together and for a population size the fitness is calculated. As per the fitness values the individuals are selected for the generation of offspring. The new generated offspring are mutated and again the fitness is calculated and the process continue until the termination condition is reached.

VII. RESULTS AND ANALYSIS

Time is the most effective parameter which decide the performance of the Hybrid MapReduce Algorithm(HMA). In this study there are several parameter which are fixed and the time taken by the Hybrid MapReduce Algorithm is comparatively less than the existing algorithm. The input data is a text file stored in HDFS consisting of the DNA of an individual freely available online. All the results are obtained for,

The population size = 100,
Chromosome length = 10,
No. of generation = 1200.

Table 1. Comparison between GA and HMA

No. of Nodes	Parameters			Time Taken	
	Population	Chromo-some Length	No. of Generations	Hybrid MapReduce Algorithm	GA
1	100	10	1200	7.5	8.1
2	100	10	1200	6.2	8.35

*****The Genetic Algorithm is executed on single machine only.

The Table 1 represents the time taken by the Hybrid MapReduce Algorithm and the Genetic Algorithm when executed on different nodes. For fixed values of population size, chromosome length and the no. of generations the time taken by the GA is 8.1 hours and Hybrid Mapreduce Algorithm takes 7.5 hours when executed on single node. Similarly when executed on 2 nodes time taken by GA and Hybrid MapReduce Algorithm is 8.35 and 6.2 respectively.

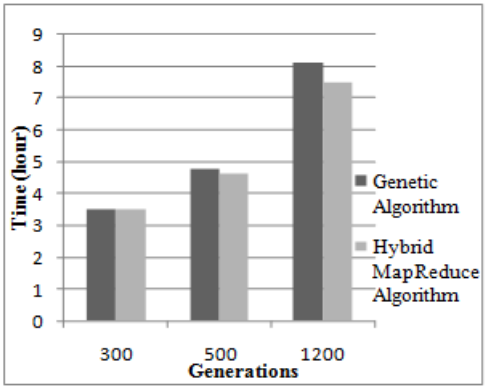

Fig. 3 Execution Time of the Algorithms (1 Nodes in cluster)

Fig. 3 represents the time taken by the HMA executed on single node of the Hadoop cluster and the Genetic Algorithm executed on single machine. In the initial generations the execution time is almost equal as the generation increases the Genetic Algorithm takes longer time to complete. For 300 generation the time by both the algorithm is equal, for 500 generation the time taken by GA is 4.76 and 4.6 by Hybrid MapReduce Algorithm. After 1200 generations time taken by GA is 8.1 and HMA is 7.5.

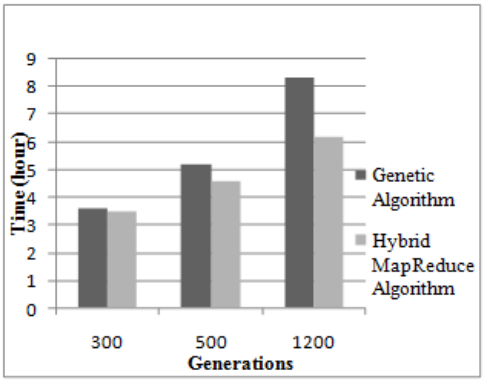

Fig. 4 Execution Time of the Algorithms (2 Nodes in cluster)

The time taken by HMA and Genetic Algorithm when executed on a two node cluster is shown in the Fig. 4. For the different set of generation it is observed that HMA takes lesser time as compared to Genetic Algorithm.

TABLE II. Average Fitness Value of GA and HMA

Sequence Length	Average Fitness	
	GA	HMA
100	33.9	34.9
1000	270	278.4

The Table 2 represents the average fitness values of the individuals in the Hybrid MapReduce Algorithm and the Genetic Algorithm when the original sequence length is 100 and 1000. For fixed values of population size, chromosome length, the no. of generations and sequence length is 100, the average fitness value of individuals for Mapreduce Algorithm is 34.9 and for GA it is 33.9.When sequence length is 1000 the average fitness is 278.4 for Hybrid MapReduce Algorithm and 270 for GA.

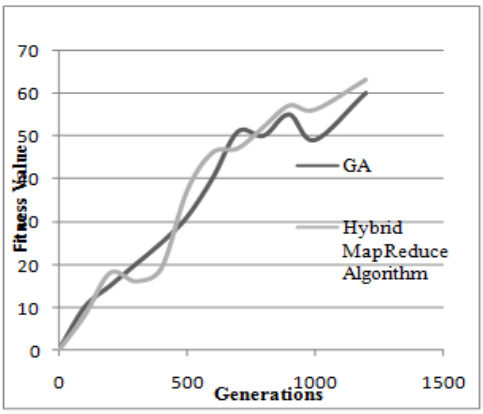

Fig. 5 Fitness Value vs Generation (Sequence length 100)

The Fitness Value of the individual is calculated by comparing it by the original sequence. In this experiment the original size of the sequence is of length 100. The fitness values obtained are shown in the Fig. 5. It is seen that the fitness values calculated by the Hybrid MapReduce algorithm are better at the end of the 1200 generations.

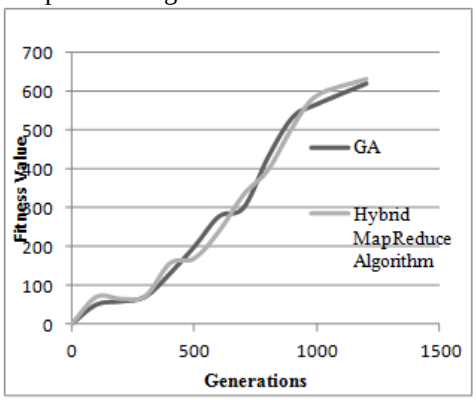

Fig. 6 Fitness Value vs Generation (Sequence length 1000)

The fitness values obtained are shown in the Fig. 6. For the total generation of 1200 and the sequence length of 1000 proposed Hybrid MapReduce Algorithm provides individuals with better fitness values.

VIII. CONCLUSION

In this paper the optimization power of the Genetic Algorithms is enhanced with the use of MapReduce Framework which provides the distributed ecosystem using the Apache Hadoop Cluster. A parallel architecture is also designed for the proposed system. Genetic algorithms are the simple and most efficient among the Evolutionary Algorithms which is based on the theory of evolution. The Genetic Algorithm is applied to solve the problem of DNA Sequencing in such a way that the genetic operators are embedded in the Map and Reduce functions which provide parallel processing. DNA Sequencing is one of the most practical and important problem in DNA computing and combinatorial optimization. Given the basic fragment of DNA a sequence is generated and matched against the original sequence.

The proposed Hybrid MapReduce Algorithm is compared with the existing Genetic Algorithm on the parameter of execution time and fitness values of the individuals. It is observed that as the number of generations increases performance of the Hybrid MapReduce Algorithm is better for the fixed set of parameters like population size, number of generations, chromosome length, rate of mutation and crossover, etc.

REFERENCES

[1] M. Srinivas, L. M. Patnaik,"GeneticAlgorithms:A Survey", Proceeding of the IEEE International Conference 1994.

[2] J.H. Holland, Adaptation in Natural and Artificial Systems, Univ. of Michigan Press, Ann Arbor, Mich., 1975.

[3] G. Xu and F. Xu, "Deploying and Researching Hadoop in Virtual Machines", Proceeding of the IEEE International Conference on Automation and Logistics Zhengzhou, China, August 2012.

[4] M. Zaharia and T. Black "Improving MapReduce performance in heterogeneous environments," Proc. Proceedings of the 8th USENIX conference on Operating systems design and implementation, USENIX Association, 2008.

[5] C. Lin , and G. Hirst, Data-Intensive Text Processing with MapReduce. Morgan and Claypool Publishers, 2010.

[6] J. Dylan Spalding, C. MacNish, "A Genetic Algorithm to Sequence DNA using Sequencing by Hybridisation Experimental Data" 2003 IEEE.

[7] A. Zec, S.Konjicija and N. Nosovic "Hybrid approach in design of GA implementation for MapReduce" International Symposium on Telecommunications (BIHTEL) October 25-27, 2012.

[8] J.Zhang , H.Chung and W. L.Lo, "Clustering-Based Adaptive Crossover and Mutation Probabilities for Genetic Algorithms", IEEE Transactions on Evolutionary Computation vol.11, no.3.

[9] A. Verma, X. Llora, R. H. Campbell, and D. E. Goldberg, "Scaling Genetic Algorithms using MapReduce," 2009, vol. 2009007. T. White, Hadoop: The Definitive Guide, Third ed. O'Reilly Media.

[10] D. Whitley and T. Starkweather, "Genitor-11: A Distributed Genetic Algorithm,"J. Experimental Theoretical Artificial Intelligence,Vol. 2,1990.

[11] D.E. Goldberg, B. Korb, and K. Deb. "Messy genetic algorithms: Motivation, analysis, and first results". Complex Systems, 5(3):493-530, October 1989.

[12] F. Jun "Evaluating I/O Scheduler in Virtual Machines for Mapreduce Application," Proc. Grid and Cooperative Computing (GCC), 2010 9th International Conference on, 2010, pp. 64-69.

[13] G. Mackey, S. Sehrish, J. Bent, J. Lopez, S. Habib and J. Wang, Introducing map- reduce to high end computing. Proceedings of the 2008 3rd Petascale Data Storage Workshop, PDSW 2008.

[14] J. Dean and S. Ghemawat, MapReduce: Simplified Data Processing on Large Clusters. In proceeding of the 6th Symposium on Operating systems Design and Implementation (OSDI 2004), pp 137-150. USENIX Association, 2004.

[15] M. Elteit, H. Lin and W. Feng, Enhancing Mapreduce via Asynchronous Data Processing. Proceeding of International Conference on Parallel and Distributed Systems ICPADS, pp. 397-405. 2010.

