A Novel Technique for Parallelization of Genetic Algorithm using Hadoop

Ms. Kanchan Sharadchandra Rahate (Khedikar)#1 Prof. L.M.R.J. Lobo*2

Student M.E.(CSE)
Computer Science & Engineering Department. Solapur University.
Walchand Institute of Technology, Solapur, Maharashtra, India.

* *Prof. L.M.R.J. Lobo.*
Professor & Head,
Department of Information Technology,
Walchand Institute of Technology, Solapur, Maharashtra, India.

Abstract—**Document categorization is used in education, government sectors, art, industry etc. Categorizing a document to enable immediate finding of it in the future motivated the concept of Classification involving Document categorization. Manual document classification involves a lot of effort and is time consuming. The basic idea implemented in this paper speeds up processing and reduces manual intervention, by atomizing this categorization. This idea is an edge over the existing classification systems. The implementation of the system basically involves getting into to parallelize Genetic Algorithm (GA) thus improving the processing speed. The use of Hadoop MapReduce and HDFS (Hadoop Distributed File System) framework helps to store big data and speeds up the calculations involved in the computation of genetic algorithm. The motivation of this work has reason from mapreduce fare well in terms of scalability, fault tolerance, and ease-of-use. This is adjoined by hadoop being an open-source and hadoop being written in Java.**

Keywords— **Hadoop, HDFS, MapReduce, PGA, OlexGA.**

I. INTRODUCTION

Parallel genetic algorithm may be implemented by many methods and produced better results than sequentially run GA in terms of space and time. The approach expressed in this paper enables us to develop Parallel Genetic Algorithm (PGA). Cloud computing framework could be used. This has been recognized as one of the latest computing paradigm where applications, data and IT services are provided over the Internet [1]. A plugin to data mining packages OlexGA is used for developing classification. Hadoop technology along with its components can be used to store big data and process computations at a faster rate. Hadoop provides various technologies like MapReduce framework, HDFS (Hadoop Distributed File System), Hive, H-base, Pig, Chukwa, Avro, ZooKeeper etc. [2]. We concentrate on developing a system that increases processing speed and capability to process huge amount of data. This contribution has an excellent demand in today's academic, medical, scientific and regular business. Document categorization is needed in the above fields; therefore it has a generalized application for commercial approach. We try to parallelize GA to improve the processing speed. We use Hadoops MapReduce and HDFS (Hadoop Distributed File System) framework approach.

The rest of the paper is organized as follows. Basic System Architecture of the developed system is explained in section II. Section III explains Algorithm of system. Methodology used in our project is given in section IV. OlexGA is explained in section V. Section VI shows Results of our implementation. Concluding remarks and Future work are given in section VII.

II. BASIC SYSTEM ARCHITECTURE

The possible operational environment for the proposed methodology has proved to be excellent and has achieved appropriate dimensions of the predicted results. The basic architecture for the proposed system is shown in Fig.1.

The blocks used in this architecture involved have been elaborated as bellow.

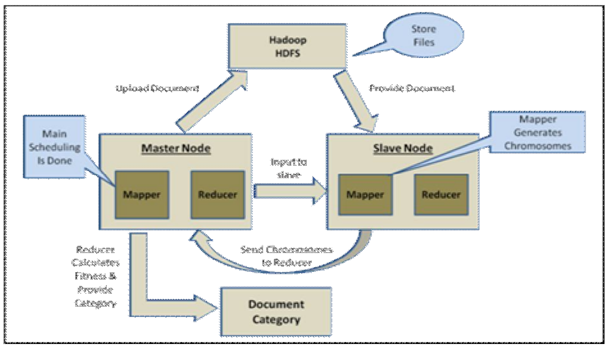


Fig 1. Basic Architecture

A. Master Node

Master node functions to upload the document to HDFS. Main scheduling is performed by this master node. The reducer at master node calculates fitness value and provides a category to documents.

B. Slave Node

Slave node takes a document from HDFS. The mapper at slave node generates the chromosomes and sends to master node for further operation.

C. Hadoop HDFS

It accepts the documents from master node and store as a file. For further calculation HDFS provides a file to slave node.

III. Algorithm

The algorithm required for this implementation is given in pseudo code

```
Step 1: Start all HDFS servers i.e. NameNode, DataNode, JobTracker, and TaskTracker on Master Node.
Step 2: Start HDFS servers i.e. DataNode and TaskTracker on Slave Nodes.
Step 3: Start GUI Model
Step 4: Click on Train Model
    • Directory is already set where system IP addresses, HDFS port number and Input folder is already specified.
    • Put Max Generation
    • Click on Extract Keywords button once which gives Top Keywords.
    • Click on Execute OlexGA button
    • While clicking we get Category and Fitness Value of document
    • Close Train Model
Step 5: Click on Test Model
Step 6: Give File Name whose category we want to know
Step 7: Click on Find Category button
    • Category Field shows Category of document.
Step 8: Exit from the Window.
```

IV. Methodology

After going through the related work we come to a state which shows Genetic Algorithm and Hadoop gives better results than other technologies. Hadoop provides a standard framework, there is no need to manage various nodes manually, Hadoop can also manage failures automatically. In our work OlexGA is used which is parallel. In GA Fitness calculation is a very time consuming task so in this model Hadoop MapReduce methodology is used to reduce processing time. Our implementation is related to finding a category of documents which helps in many fields like Education, Medical, IT Industries, and Government etc. This model has two phases named as Train and Test. Train model provides a classifier which is input to test model which provides category of document. The detail methodology is explained in the sections bellow:

A. Train Model

The Train model takes documents and keywords from Hadoop HDFS which then provides to Parallel Genetic Algorithm (PGA). PGA gives a population which is given to an Input Format. The Input Format splits data and provides it to the mapper. There are number of mappers, they generate chromosomes and pass to reducer. Reducer is responsible for calculating fitness. Then output of reducer is forwarded to Output Format after that new population is obtained which is given to PGA and finally the output is directed to a classification model. The Classifier is output of train model.

B. Test Model

Output of train model i.e. classifier, keywords and test document are given to test model. Test model then finds chromosomes with maximum fitness and returns the category of fittest chromosome. Final output of test model will show the category of a given text document.

V. OlexGA

OlexGA is a plugin for implementing parallel GA in a data mining package. Olex GA gives faster text categorization than others like Naive Bayes, C4.5, Ripper etc. All these algorithms are complex

& time consuming. Olex genetic algorithm classify document **d** under category **c** if

t1 € d or … t n € d and not (t n+1 € d or … or t n+m € d) holds [8].

Where c is a category, d is a document and each **ti** is a term. Pos indicate positive term and Neg indicate negative terms. Olex-GA relies on an efficient several-rules-per-individual binary representation and uses the F-measure as the fitness function.

```
Step 1: Start OlexGA
Step 2: Configure Hadoop
        Configuration conf=new Configuration (true);
        conf.set("mapred.job.tracker", "192.168.1.2:54311");
Step 3: Create job for every generation
        Job job = new Job(conf);
Step 4: Configure current job
        job.setJarByClass(OlexGA.class);
        job.setInputFormatClass(OlexInputFormat.class);
        job.setOutputFormatClass(OlexOutputFormat.class);
        job.setPartitionerClass(OlexPartitioner.class);
        job.setMapOutputKeyClass(Chromosome.class);
        job.setMapOutputValueClass(DoubleWritable.class);
        job.setGroupingComparatorClass(OlexGroupingComparator.class);
        job.setSortComparatorClass(OlexGroupingComparator.class);
        job.setMapperClass(OlexMapper.class);
Step 5: Give this job to hadoop
Step 6: Call two methods for submission of job and
        for completion of job
        job.submit();
        job.waitForCompletion(true);
Step 7: Fitness of chromosome is obtained after
        completion of job
Step 8: Perform Sort chromosomes using fitness as
        sort key
Step 9: Perform Selection of best chromosomes
Step10: Perform Crossover on selected
        Chromosomes
Step 11: Perform Mutation on randomly selected
        Chromosomes.
```

Initially OlexGA is run and then Hadoop is configured by creating a configuration object. While configuring hadoop we need to set IP address of nodes (192.168.1.2) and port number (54311) of HDFS.

A Job is created for every generation. Job signifies some work is given to hadoop. It passes hadoop configuration to job (Job job = new Job(conf);). A class name is set for every job the OlexGA package formed contains classes named InputFormatClass, OutputFormatClass, PartitionerClass, Chromosome, DoubleWritable, OlexGroupingComparator, SortComparator, OlexMapper using (job.setInputFormatClass(OlexInputFormat.class);). To give these job to hadoop submit method (job.submit();) is used. Once the job is submitted a wait for the completion of job for this is looked for by method (job.waitForCompletion();). After completion of a job fitness of chromosome is obtained which is the output of our Train model.

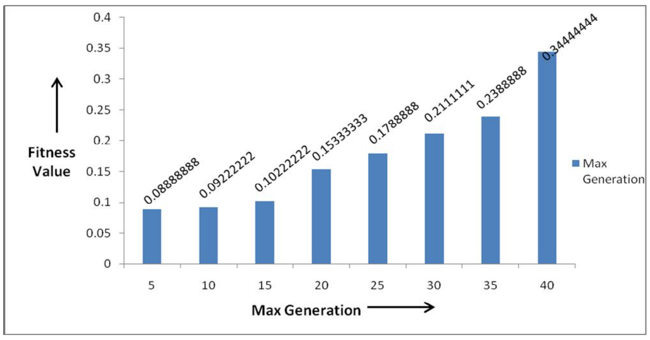
A. Hadoop Configuration:

The files for Hadoop need to be configured. Some of the files which need this process to be done manually are core-site.xml, mapred-site.xml, masters and slaves files.

Master and slave configuration of hadoop shows IP addresses of master and slaves. In this implementation one master and two slaves are present. Master can act as a slave also. The Master node IP address is 192.168.1.2 and IP address of slave nodes are 192.168.1.2 and 192.168.1.3.

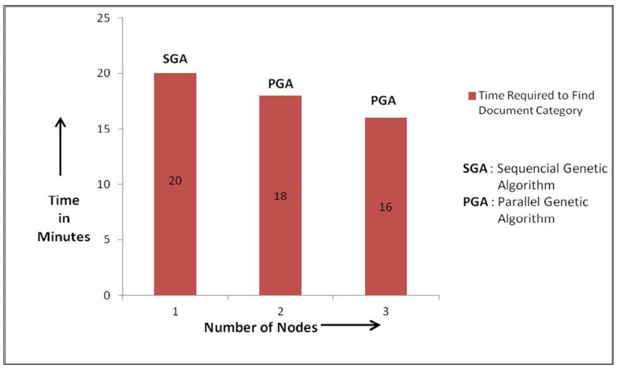
VI. RESULTS

The implementation involves collecting data from many reputed journals like IEEE, ACM and Science Direct. Then converting that data into text format using a converter tool Tika. Once this is done a text document is available. This stored into hadoop HDFS. In this implementation we parallelize GA to improve the processing speed. We use map and reduce components of hadoop for



fitness evaluation. We have stored 4 GB (Gigabytes) of data into hadoop distributed file system. Based on these data entries, results are generated. Results are of two types Administrator Model and User Model.

A. *Administrator Model Result*



Our model shows different subject categories and fitness values. Suppose observation is for one subject named as web semantic. Fig. 2 show as we increase max generations, fitness value is also increased.

Fig.2 Administrator Model Result for Single Subject

B. User Model Result

Fig. 3. User Model Result

Fig.3 show Results obtained by using single node will take maximum time to find document category. But results obtained by using two or three nodes will give better results in terms of time. This is known as parallelization of genetic algorithm using hadoop mapreduce framework.

VII. Conclusion Remarks and Future Work

In this implemented work we developed a model which provides a category to documents. This involves storage of big data.

Here Parallel Genetic Algorithm (PGA) i.e. OlexPGA is made use of. In genetic algorithm the fitness evaluation is a time consuming task. Hadoop mapreduce framework is used to reduce processing time required for fitness calculation. Hadoop HDFS (Hadoop Distributed File System) is used to store documents. This model has a generalized application for commercial used.

The future work to this implementation wold involve developing two phases train phase and test phase. Train phase to generate a classifier which will be provided as a service to client in a cloud.

References

[1] Kanchan A. Khedikar and Dr. Mrs. S. S. Apte. "Latest Technology In Networking: Cloud Architecture", in *ICETT 2010*.

[2] Konstantin Shvachko. The Hadoop Distributed File System, 978-1-4244-7153-9/10/$26.00 published at *©2010 IEEE.*

[3] Apache Hadoop. http://hadoop.apache.org/

[4] Genetic Algorithms in the Cloud" from *MENTION*.

[5] Dhruba Borthakur. The Hadoop Distributed File System: Architecture and Design.

[6] Introduction to Genetic Algorithm http://lancet.mit.edu/mbwall/presentations0/IntroToGAs/

[7] Mariusz Nowostawski And Riccardo Poli. Parallel Genetic Algorithm Taxonomy published in *KES'99, May 13, 1999*.

[8] Adriana Pietramala, Veronica L. Policicchio, Pasquale Rullo and Inderbir Sidhu. A Genetic Algorithm for Text Classification Rule Induction Appears in W. Daelemans et al. (Eds.*): ECML PKDD 2008, Part II, LNAI 5212, pp. 188–203, 2008. @ Springer-Verlag Berlin Heidelberg 2008.*

[9] Sandeep Tayal. Tasks Scheduling Optimization for the Cloud Computing Systems. In (IJAEST) International Journal Of Advanced Engineering Sciences And Technologies, Volume No. 5, Issue No. 2, 111 – 115.

[10] Linda Di Geronimo, Filomena Ferrucci, Alfonso Murolo, Federica Sarro. A Parallel Genetic Algorithm Based on Hadoop MapReduce for the Automatic Generation of JUnit Test Suites. *2012 IEEE Fifth International* Conference on Software Testing, Verification and Validation. 978-0-7695- 4670-4/12 $26.00 © 2012 IEEE DOI 10.1109/ICST.2012.177

[11] Lecture on MapReduce access on http://hadoop.apache.org/mapreduce/.

[12] Lecture on hadoop HDFS access on http://hadoop.apache.org/hdfs/.

[13] Hadoop: The Definitive Guide by Tom White, First Edition. Copyright © 2009 Tom White. All rights reserved. Printed in the United States of America. Published by O'Reilly Media. (e-book)

[14] Hadoop MapReduce Cookbook by Srinath Perera and Thilina Gunarathne. Copyright © 2013. Published by Packt Publishing Ltd., *ISBN 978-1-84951-728-7*.