



www.ijarcsse.com

Volume 4, Issue 7, July 2014

ISSN: 2277 128X

# International Journal of Advanced Research in Computer Science and Software Engineering

Research Paper

Available online at: [www.ijarcsse.com](http://www.ijarcsse.com)

## Map Reducing Stream Based Apriori in Distributed Big Data Mining

Y. Venkata Raghavarao \*

Research Scholar, JNTUH,  
India

L. S. S Reddy

Vice chancellor, K L University  
India

A. Govardhan

Director School of IT, JNTUH  
India

---

**Abstract**— *Distributed BigDataMinig (DBDM) is architecture to rectify the mismatch between the available centralized old data mining systems and distributed systems. One feature of DBDM is its plug-in concept to carry out different data mining algorithms on big data by the help of Map Reduce functionality. The localized apriori algorithm transformed into to a distributed type of algorithm with the help of Map Reduce technique, for generating the efficient frequent big data patterns in distributed environment, which are similar to the results obtained in centralized system. Here the important feature of Map Reduce is 'share-nothing' type of agent, while implementing in distributed big data frameworks, where each big data client node can acts as a communication agent to the server node, strictly restricting internal communication among the client nodes. Hadoop is open source platform specially designed for implementing Map Reduce technique since there are no efficient algorithms, which produce frequent streaming patterns yet. DBDM also aimed to find frequent stream data patterns by using our improved apriori algorithm.*

**Keywords**— *AprioriImp, association, communication agents, distributed Big Data mining, Big data agents*

---

### I. INTRODUCTION

This outline represents a schematic diagram of traditional data warehouse-based architecture in data mining. This model of data mining is useful to extract frequent stream patterns in stock markets, where mission critical data has been uploading daily to the Big Data server. Sometimes data may be stored in data warehouses and also in subsequent centralized data mining storages. This centralized approach is not suitable for mining knowledge in both distributed and ubiquitous systems. Henceforth data mining and stream data mining algorithms to be improved. Because of long response time, lack of proper usage of distributed resources, the fundamental characteristics of centralized data mining algorithms do not work well in distributed environments, so there is a need of providing suitable solution in distributed applications for big streaming data processing, which is mainly controlled by the available resources and human factors. For example, consider an ad hoc wireless sensor network where the different sensor nodes are monitoring some time-critical events. Central collection of data from every sensor node may create heavy traffic over the limited bandwidth wireless channels and this may also drain a lot of power from the devices. A distributed architecture for data mining is likely to reduce the communication load and also reduce the time and space complexity more evenly across the distributed homogeneous or heterogeneous data nodes in the network. Analyst can easily imagine similar responsibilities for distributed computation of data mining primitives in wireless networks of mobile data like PDAs, cell phones, and personal computers. Bigger or huge data applications include personalization, graphical views and collaborative process monitoring, intrusion detection over ad hoc wireless networks. Analysts need heterogeneous data mining architectures that pay careful attention to the distributed resources of data, computing, and communication in order to consume them in a near optimal fashion. Distributed data mining (DDM) considers data mining in this broader context

Sometimes data may be in the form of continuous streams and the computation style performed in real time for big data storages. Therefore, need to provide event-based efficient software for accessing of large big stream data files. In many situations instantaneous data mining to be done at site servers like face book and twitter where large data coming to server dynamically, this system may be extensible in order to scale up dynamically adding new servers for handling the input data coming in a big way. And also The software should facilitate a package of classification for different models. For reading a predictive model to be generated in PMML [6] this is a XML-based language for describing the association rules and different types of data mining algorithms.

### II. ANALYSIS OF APRIORIIMP IN DISTRIBUTED DATA MINING

However the data distributing algorithms need more storage cost at each remote site for storing all candidates for each scan. Performance may be degraded if not provided much memory. The task of parallelizing algorithms can get rid of this type of degrading. The task of distribution might work where BigData distribution may not work fine [4]. So a newly estimated DBDM approach that is using BigData agents for instant task distribution will predict efficient results. Investigating suitability of AprioriImp algorithm in parallelized approach, proposed 4 parallel algorithms based on AprioriImp; speed up mining of frequent item sets.

The available data mining algorithms implemented in distributed mining of data are of communication intensive. Many of streaming algorithms for big data mining have been investigated for predicting knowledge at a local based data repositories, normally some of them are implemented at remote data nodes by some improved code but, in terms of efficiency of these algorithms as a part of quality, but not specially designed for complexity based distributed environment, and even big data is not addressed properly, data on the web/network are distributed by very of its nature. So for selecting the suitable distribution of data to be selected out of four types of algorithms[5]. Four types of logics, The Candidate Distribution Mining (CDM) algorithm parallelizes the task of generating longer patterns and load balancing algorithm that facilitates synchronization between the processors and segments. The big streaming data supplied by different transaction patterns. Of all these parallel algorithms which were tested among each other and CD evolved as the best for AprioriImp algorithm. CD's overhead is less than 7.05% when compared with AprioriImp, and tried to make DDM and CDM in scalability concept instead of Hybrid Distribution (HD) algorithm respectively.

CDM addresses the issues of communication solves overhead and redundant computation in by using aggregate memory to divide candidate sets and moving streamed data in a big way. CDM improves over by dynamically partitioning of candidate set to manage good load balance. Experimental results showing that the response-time of CDM is 4.4 times less than DDM on a 64-processors clients and HD is 9.6% better than CDM on 128 processors. the following graph shows comparison of the running time of above four algorithms.; so it is taken as stable in Data Distribution Mining (DDM).so proposing the AprioriImp algorithm as shown in Figure.4

### **III. SUITABILITY OF APRIORI IN DISTRIBUTED BIGDATA MINING**

The 'AprioriImproved Algorithm' (AprioriImp) is mainly applied for generating frequently occurred Item Sets (FOIS) by Using Candidate Generation  $c$ . This influential algorithm generates FOIS in Boolean association rules. The name itself elaborating it's meaning, getting input data from prior knowledge of FOIS. AprioriImp using an iterative function known as a step-wise search, where  $k$ 'th items sets are repeatedly helped to generate  $(k+1)$  items sets. In the sets of frequently Occurred  $l$ 'th-item sets generated. This set is named as  $L_1$  generally.  $L_1$  set is used to generate  $L_2$ , the set of frequent 2-item sets, which is used to generate  $L_3$  item set, repeatedly, until no more frequent  $k$ -item sets can be found. For generating each  $K$ -item set as  $L_k$  requires one full Memory usage scanned repeatedly on both the local and global Big databases .here we are going to focus mainly on Big Data . where huge log files or stock market data ,streaming data , twitter data , and huge face book trends of larger sized data storages located around the globe which are connected different wide area networks (WANS), which badly affects the bandwidth .these type of limitations and scenarios are forcing us to provide solution to handle the frequent pattern Mining on Big Data which may be called as global data mining on Big data which is nothing but Distributed BigData Mining (DBDM).

Applying the above discussed algorithm AprioriImp in this DBDM which is useful for finding frequent patterns on both local data mining locally or remotely. So AprioriImp algorithm is applied on Local sites single node cluster. or Running on cluster nodes of big data . DBDM is mainly for analyzing data partially on local sites and then sending all these calculated intermediate results to remote sites sometimes useful for calculating global outcome which is nothing but final result ,final set of frequent patterns .In order to support DBDM architectures , consider some typical situations where, central data server need to collect data from several servers for example a search engine collecting log data from many web servers distributed remotely over the globe . Local information server in our city collects data from different depts. of different cities for mining distributed BigData bases. At each server local agent processes the log files and returns the partial results as parameters to the main server creates Low network traffic because of the software agents are used for local data processing [6].

For finding suitable algorithm and useful mining architecture in distributed environments may be useful . The main objective of the paper is to develop architecture to support Distributed BigData Mining (DBDM), useful to mine the hidden predictive information from big databases, henceforth this DBDM is helping potentially for stock markets and big data handling companies located in distributed fashion. There are three types of data collections systems Conventional Data Collection System (CDCS), Decentralized Data Collection System (DDCS), Data collection with communicative Mobile agents (DCCS) are three preferred solutions [2].

### **IV. ARCHITECTURE FOR DISTRIBUTED BIG DATA MINING**

The module DBDM is responsible for predicting frequent stream patterns of Big Data nodes and to scale up the data nodes into the right format, such that it can be used as a plug in. After extraction, it has to commit the predicted data mining models to the DBDM. Subsequently the frequent item stream patterns must be converted to pmml models

The Proposed architecture DBDM uses XML and Hadoop standards which are important. They run data mining algorithms for predicting pmml also, which are useful to different knowledge discovery systems distributed in remote locations. The main components and the functionalities of these components are installed. HDFS deployment in dedicated server is for running only on the NameNode. Where the other clients in the cluster specially running as one copy of the DataNode software, as though the DBDM architecture allows you to run several DataNodes on the same server. The NameNode is managing metadata repository and control; it never handles user private data. The NameNode using a special kind file, with name *EditLog*, consisting of metadata.

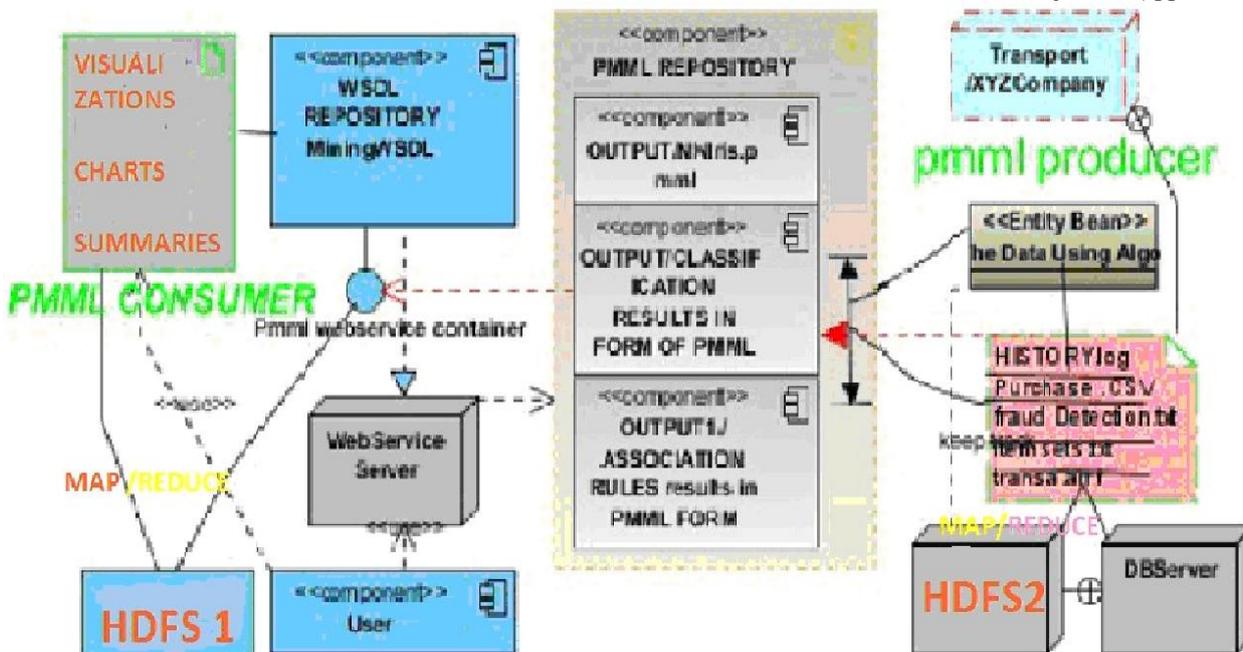


Fig. 1 Proposed DBDM Architecture

Describing components from top-left the visualizations are achieved through PMML is an XML exchangeable format describing different data mining models. In DBDM the PMML module is an outcome of different data mining algorithms. As PMML is an open source, each model can be subsequently used by another applications or data mining tools. As this PMML applied to BigData, log files, continuous data, web collected log files, stock market data, collected at parallel, webService server. And Log files are collected at each sessionizer which automatically generated by DBserver. The Mining Repository (MR) is a part of HDFS functionality which is suited for exporting and importing pmml models which are produced as outcomes of different data mining algorithms as discussed above. and finally integrated to generate stream based patterns in the form of graphs and visualizations, Here MapReduce played a vital role while generating final outcome showed as visualizations in following manner, each module results collected by different DBDM agents are to be processed by another name node Agent for achieving target output. The final out puts may be shown as graphs with use of visualization tools as shown in the Figure.1.

### V. MAP REDUCING APRIORI IN BIG DATA MINING

The essential functionality of data mining is for extracting patterns or knowledge from files or big data bases. The generated mined out put generally helping in making market improvement decisions. important knowledge extracting algorithms for mining are classification, sequential pattern, clustering and clustering, association rules, and the interesting and popular algorithms generally implements in industry is frequently occurred patterns mining, where extracting of FOIS. The main logic of FOIS is to generate frequent patterns from a data file or log file / big data file, even from big data base cluster also. Association rule in the form of  $X \Rightarrow Y$ , here each X and Y are sets of items. In this association rule based mining there are mainly two essential algorithms for frequent pattern mining or even for frequently occurred patterns mining FOIS Apriori based approach Mining (Agarwal & Srikanth 1994) and Frequent Pattern Growth approach Mining (Han et al 2004).

Association rule mining, mining, and. The most important and popular topic in data mining is frequent pattern generation. The basic concept of frequent pattern mining is to discover patterns from database that are more frequent than the specific threshold. As per association rule is defined as  $X \Rightarrow Y$ , where X and Y are the set of items. There are two main primitive algorithms in frequent mining are Apriori algorithm (Agarwal & Srikanth 1994) and Frequent Pattern Growth approach (Han et al 2004). Apriori algorithm facing two limitations.

First problem is the memory of the system should be large to save many candidate item sets. and second problem is number of times of scanning the data base is very huge. so for overcoming this limitations Han(2004) produced his new algorithm by facilitating a new data structure for storing data item sets is Frequent -Pattern tree (Fp-tree), all transactions are saved as compressed formats in this Fp-tree structures.

This algorithm, frequently occurred patterns item sets have been generated from big databases, or csv files. But increase in saving data size and frequency of saving effects the running time and execution time is more. The demand of memory is also very high in case of single task is running on big data Clusters where the given threshold is very low. and takes long time to execute. Also demand for memory was the main concern in case of single task computing because to mine large databases more memory is required to handle large number of item sets especially when the threshold is low.

VI. MAP REDUCING PROGRAMMING MODEL

A Map-Reduce algorithm is consisting of a map() function which is used for filtering and sorting such as sorting employees by first in first out as emp-name into data structures queues, and one queue for each emp-name another function reduce() which generates a summary function which is counting the number of employees in each queue, resulting emp-name frequencies. The Map-Reduce resembles by serializing the remotely located servers, running different jobs in parallel, managing all communications and data transfers between different modules of the system, facilitates for redundancy, fault tolerance. Map Reduce implementation provides a template model for processing and outputting large sized data sets. The coder also providing such map function which computing a key\value pair to produce a set of intermediate key\value pairs, even a reduce function used to combine all intermediate results associated for the same intermediate key.

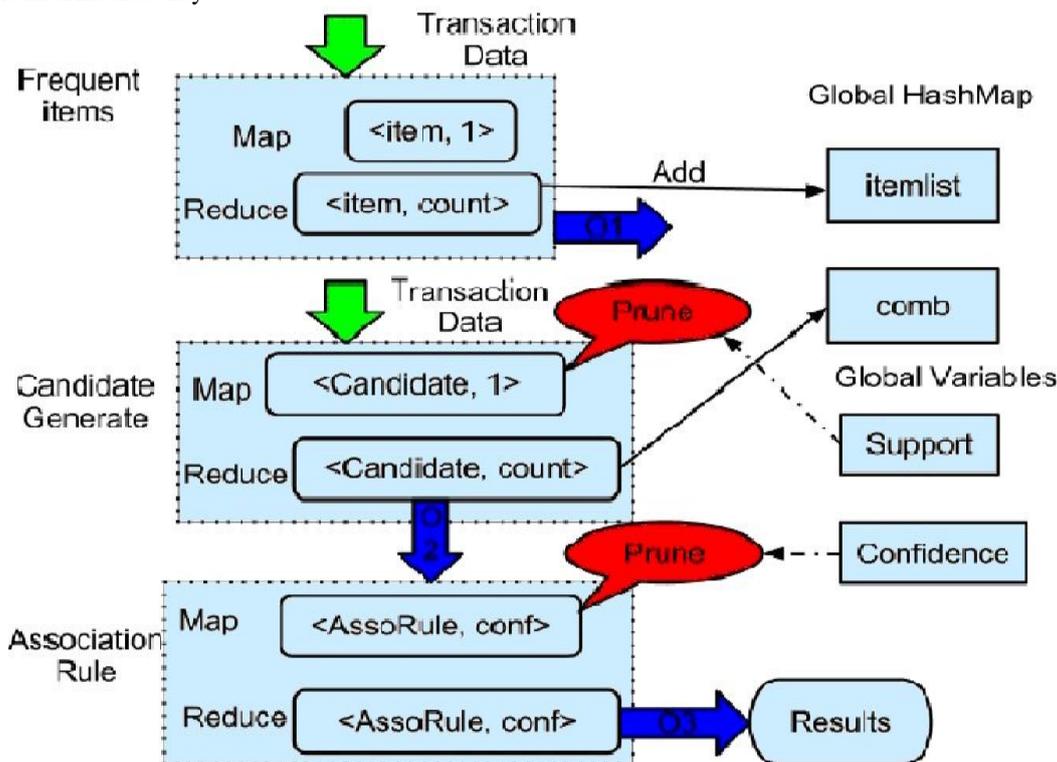


Fig. 1 Proposed Map Reduce for DBDM Architecture.

In the above Figure 2 job1 tool is for producing frequent 1 items sets like  $\langle item, count \rangle$  key,value pair (here count is showing frequency of item), and mapped to the list called HashMap. *Ofreq1* is the output generated. The job2 component is for producing candidate sets. This source data file is taken as input for this type of code generation.. Firstly map phase, pruning the items those occurred less than the support provided by using the global HashMap item list, and repeatedly calling the algorithm function *GenCandidatesRecur()* to produce candidate sets. The important reducing phase joining candidate item sets and to add the count by  $\langle candidate-item set, count \rangle$  key-value pair. The third job tool named as Association Rule by taking the output produced by candidate generation job as input and Producing BigDataMining association rules on the candidate item- sets. here pruning the left hand side of an association rule by observing its count in the HashMap so as discussed above the pseudo code for apriori improved is as follows.

ALGORITHM AprioriImp\_Bigdata ():-

```

For all DataNodes if (DataNode=true) {
    I1 =items set A; I2=items set B; Procedure combine ()
    For all up to k-1 {
        Select p1.item1... I1.itemk-1, I2.itemk-1 from Lk-1I1, Lk-1I2 where I1.item1 = I2.item1 and I1.itemk-2 =
        I2.itemk-2 and I1.itemk-1 < I2.itemk-1;
    }
    If (agent=false)
    // remove items sets such as that some (k-1) th subset not present in Lk-1
    for all c in Ck
    Process _eliminate ()
    For loop up to {(k-1)-subsets of c if [(s is not in Lk-1)! = true]}
    Eliminate c from ck; break;
}
}
    
```

Fig. 3 Proposed Map Reduce Apriori for DBDM Architecture.

## VII. IMPLEMENTATION AND RESULTS

### Experimental Setup

The implementation of AprioriImp\_Agent() algorithm with Agent class, Resulting reader and a CsvconvertToData classes and writing map and build reduce XML files, build successful after running Hadoop fs -put output. data output1.data,now by running hadoop@ubuntu:~\$<HADOOP\_INSTALL>/bin/GenerateAll,now dispatching the mining agent to another host, the following graphical user interface created and our data mining agent ready to do AprioriImp mining on the Web server Log files which are downloaded and processed through a sessionizer and save that file as LOGML,another host located in distributed environment may asked to supply support values and the task of apriori mining is completed at remote host as shown in Figure.3 in the similar way any remote host supply datasets like log files, weblogs,LOGMLs,data.txt as input to our Distributed Data Mining (DDM) with the help of aglets which are event based mechanisms.

PMML allows a single style to represent a predictive solution in data mining, regardless of the environment in which it was built. Part 3 of this paper described important phases for obtaining the predictive unique solution, they are mainly data pre-preprocessing and model building finally targeting at post-processing of model scores. PMML is nothing but a single file which represents this important phases. It can also represent solutions that include multiple models or a model ensemble.PMML is XML-based. Its schema follows a well-defined structure in which elements and attributes are used to define: Data Dictionary elements useful for input data ,Invalid, missing and outlier value handling strategies through element Mining Schema Data pre-processing achieved through element TransformationsDictionary A myriad of modeling techniques via specific model elements such as: Neural Network, Tree Model, SupportVectorMachineModel, Scorecard, and Regression Model Post-processing of model outputs via element Output PMML also contains other language structures including specific elements for model verification and model explanation and evaluation. This export PATH=\$PATH:[HADOOP\_DIR]/bin: PATH:/bin.

### Some code snippet: In Hadoop based java

```
public class ConvertCsvconvertToDat {
    public static void main(String args[]) throws Exception {
        if (args.length != 1) {
            BufferedReader csvinput = new BufferedReader(new FileReader(csvFilename));

            // with the first line we can get the mapping id, text

            FileWriter dataWriter = new FileWriter ("output1.dat");
            int transaction count = 0;
            While (true) {
                line = csvinput.readLine();
                if (line == null) {
                    break;
                }

                dataWriter.close ();
                System.out.println ("Wrote" + transaction Count + "transactions.");
            }

            public class ResultingReader {
                public static Map<Integer, Long> read Frequency(Configuration config, String filename) throws Exception {

                    Map<Integer, Long> frequency = new HashMap<Integer, Long>();
                    Text ourkey = new Text();

                    While (frequencyReader.next (ourkey, value)) {
                        frequency. put (Integer.parseInt(ourkey.toString(), value. get());
                    }
                    return frequency;
                }
            }

            public static Map<Integer, String> readMapping(String fileName) throws Exception {
                Map<Integer, String> itemById = new HashMap<Integer, String>();
                BufferedReader csvinput = new BufferedReader (new FileReader (fileName));
                while(true) {
                    String line = csvinput.readLine ();
                    if (line == null) {
                        break;
                    }
                }
            }
        }
    }
}
```

Fig. 4 (Algorithm): Implementation of MapReduce apriori.

By using the above algorithm and the following results are obtained at hadoop as shown in Figure .5 by executing all experiments at the command prompt , by including all the required jar files.

```
[Sweet Corn] => Hot Dogs: supp=0.037, conf=0.518, lift=0.013, conviction=1.859
[Eggs] => Hot Dogs: supp=0.053, conf=0.450, lift=0.002, conviction=1.616
[Hot Dog Buns] => Hot Dogs: supp=0.042, conf=0.452, lift=0.005, conviction=1.819
[White Bread] => Hot Dogs: supp=0.040, conf=0.437, lift=0.003, conviction=1.463
[Eggs] => 2pct. Milk: supp=0.052, conf=0.467, lift=0.003, conviction=1.676
[White Bread] => 3pct. Milk: supp=0.041, conf=0.480, lift=0.003, conviction=1.562
[Hot Chips] => 2pct. Milk: supp=0.035, conf=0.409, lift=0.002, conviction=1.538
[White Bread] => Tomatoes: supp=0.040, conf=0.611, lift=0.004, conviction=2.265
[White Bread] => Eggs: supp=0.045, conf=0.449, lift=0.003, conviction=1.599
[2pct. Milk] => Eggs: supp=0.062, conf=0.425, lift=0.003, conviction=1.549
[...]
```

Fig. 5 frequent item sets of improved apriori

The Algorithm DBBM performance in hadoop with single node is much greater than Hadoop with multiple nodes. the performance DBDM on Hadoop also depends on the nodes in hadoop provided to the cluster. this performance touches almost 1.2 in double nodes cluster than a single node. if we compare to eight nodes and the performance raised to 8 time than a single node hadoop based DBDM. The iteration number for generating Frequent item sets is also effecting the performance. for reading assigning to each node is  $O(n)$  where  $n$  is the number of items for a transaction  $m$  is the number of pairs. for sorting  $O(n \log n)$  on merge sort, for conversion into pairs  $\langle \text{items}, \text{key} \rangle$  running time complexity is  $O(n \times m)$ . so the final time complexity for mapping functionality is  $O(n + n \log n + n \times m)$ .

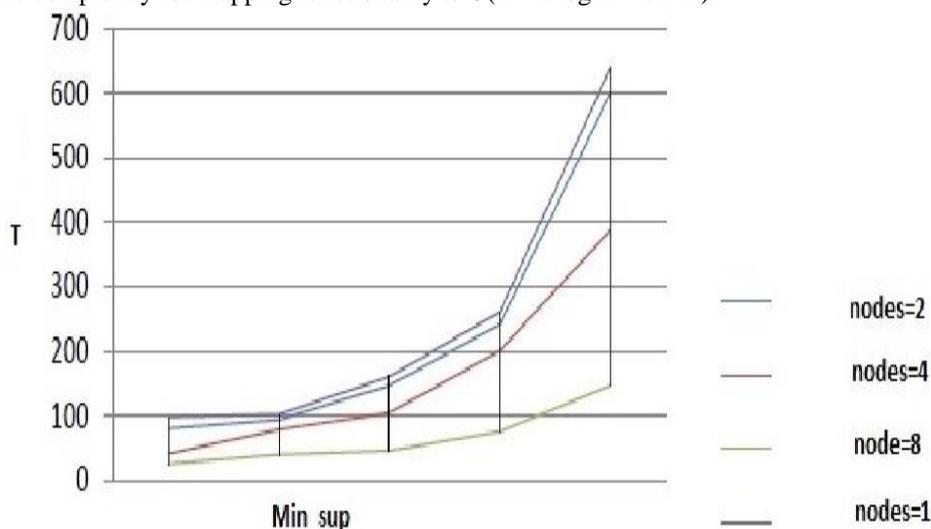


Fig. 5.1. Chart for Execution Time

## VIII. CONCLUSIONS

This paper proposes the solutions for the issues of knowledge discovery in distributed BigData mining with less load. The developed scheme can be extended to DW. As it is limited with DW usage, it uses flat files with some huge amounts of data. The data files can be taken as input by taking all the files in distributed environment. It can also be extended within and out of the domain, and from one LAN to another LAN. Computing and multi-core based concurrent programming are recent advancements in the field of solving larger problems. Multi-agent simulations when scaled up to several millions of agents is one such problem posing several challenges - like scalability of framework running such massive simulations; a lot of computation power is needed to execute agents along with primary and secondary storage to store simulation data. And fast agent execution and message delivery along with latest agent-environment state perception. In our work, we addressed these challenges and provided separate solutions for them. The challenge with scalability is handled with Hadoop-cum-Lucene solution and agent-state perception challenge is tackled by utilizing Big data systems. So Authors are working in these areas.

## REFERENCES

- [1] Abounaga, P. Haas, M. Kandil, S. Lightstone, G. Lohman, V. Markl, I. Popivanov, and V. Raman. Automated statistics collection in DB2 UDB. In VLDB, 2004.
- [2] 2nd International Work- shop on Data Mining Standards, Services and Platforms kdd 2004. | [6] Agents and Stream Data Mining: A New Perspective May/June 2005 (vol. 20 no. 3)
- [3] Han, J., and Kamber, M.: Data Mining: Concepts and Techniques (2nd version). Morgan Kaufmann (2006) Ahmad and L. Dey. A k-mean clustering algorithm for mixed numeric and categorical data. Data and Knowledge Engineering, 63(2), 2007.

- [4] 2nd International Workshop on Data Mining Standards, Services and Platforms KDD 2004. Agents and Stream Data Mining: A New Perspective May/June 2005 (vol. 20 no. 3)
- [5] R. Agrawal, J. Shafer: "Parallel mining of association rules". IEEE Transactions on Knowledge and Data Engineering, 8(6) 962-969, 1996.
- [6] Cao, L., Gorodetsky, V. and Mitkas, P. Editorial: Agents and Data Mining. IEEE Intelligent Systems (2009).
- [7] Han, J., and Kamber, M.: Data Mining: Concepts and Techniques (2nd version). Morgan Kaufmann (2006)
- [8] R Core Team. R: A Language and Environment for Statistical Computing. R Foundation for Statistical Computing, 2009.
- [9] Apache Hadoop Project, <http://hadoop.apache.org/>,
- [10] Z. Zheng, H. Zha, T. Zhang, O. Chapelle, K. Chen, and G. Sun. A general boosting method and its application to learning ranking functions for web search. In NIPS, 2007
- [11] Introduction to Cloud Computing", Jongwook Woo, the 10th KOCSEA 2009 Symposium, UNLV, Dec 18-19, 2009