International Journal of Computer Science and Engineering (IJCSE)
ISSN(P): 2278-9960; ISSN(E): 2278-9979
Vol. 3, Issue 1, Jan 2014, 19-28

International Academy of Science, Engineering and Technology
Connecting Researchers; Nurturing Innovations

MAPREDUCE BASED ECLAT ALGORITHM FOR ASSOCIATION RULE MINING IN DATAMINING: MR_ECLAT

MANALISHA HAZARIKA & MIRZANUR RAHMAN

Department of Information Technology, Gauhati University, Guwahati, Assam, India

ABSTRACT

This Data mining is the process of extracting useful information from the huge amount of data stored in the databases. Data mining tools and techniques help to predict business trends those can occur in near future. The goal of the paper is to experimentally evaluate association rule mining approaches in the context of vertical database partitioning This paper introduces a new MapReduce based association rule miner for extracting strong rules from large datasets. Due to rapid growth of data, large scale data processing is becoming a focal point of information technique. To deal with this advancement in data collection and storage technologies, designing and implementing large-scale parallel algorithm for Data mining is gaining more interest. We design Association Rule based parallel data mining algorithm which deals with Hadoop. An association rule mining helps in finding hidden relation between the items or item sets in the given data. The association rules are developed on the basis of the frequent item set generated from the data. The frequent item set are generated following the ECLAT algorithm. As the input data and number of distinct items in the data set is large, lots of space and memory is required in traditional system, so in the modified Eclat Hadoop is used, as Hadoop provide parallel, scalable, robust framework in the distributed environment.

KEYWORDS: Association Rule, Data Mining, Eclat, MapReduce, Parallel

I. INTRODUCTION

The advent of information technology in various fields of human life has lead to the large volumes of data storage in various formats like records, documents, images, sound recordings, videos, scientific data, and many new data formats. The data collected from different applications require proper mechanism of extracting knowledge/information from large repositories for better decision making. Knowledge discovery in databases (KDD), often called data mining, aims at the discovery of useful information from large collections of data [1]. The literature available for data mining contains many definitions [2][3][4][5]. Some of them depend on the application and how data has been organized into a database whereas some of them depend on the discovery of new information from the facts in a database. Datamining is a process by which one can extract interesting and valuable information from large data using efficient techniques. Conventional data mining algorithms are developed with the assumption that data is memory resident, making them unable to cope with the exponentially increasing size of data sets. Therefore, the use of parallel and distributed systems has gained significance. Generally, parallel data mining algorithms work on tightly coupled custom-made shared memory systems or distributed-memory systems with fast interconnects. Other algorithms designed for clusters like loosely coupled systems are connected over a fast Ethernet LAN or The main differences between such algorithms are scale, communication costs; interconnect speed, and data distribution. Map-Reduce is an emerging programming model to write applications that run on distributed environments. Several implementations such Apache Hadoop are currently used on clusters of tens of thousands of nodes [6]. This paper focuses on Map-Reduce design and mathematical view of one new data mining techniques relating to associative rules [7,8] and associative classification. To deal with this advancement in data collection and storage

technologies, designing and implementing large-scale parallel algorithm [9] for Data mining is gaining more interest. This trend to use distributed [10], complex, heterogeneous computing environments has given rise to a range of new data mining research challenges [11,12,13]. The rest of paper is organized as follows. In Section II we described Related Work, including MapReduce programming model and equivalence class partitioning and vertical database transformation Section III presents Analysis of éclat algorithm section IV presents Proposed Algorithm and explains about applying improved Eclat algorithm on MapReduce. Section V analyzes the experimental results and Section VI concludes the paper followed by limitation and future research recommendation.

II. RELATED WORK

MapReduce Programming Model

MapReduce [14] introduced the easy and abstracted programming model, MapReduce. Many computation problems can be expressed using this model. It is inspired by functional programming languages. The input and output data have a specific format of key/value pairs. The users express an algorithm using two functions: the Map functions and the Reduce function. The Map function is written by the application developer. It iterates over a set of the input key/value pairs, and generates intermediate output key/value pairs. The MapReduce library groups all intermediate values by key and introduces them to the reduce function. The Reduce function is also written by the application developer, it iterates over the intermediate values associated by one key. Then it generates zero or more output key/value pairs. The output pairs are sorted by their key value.

(input) <k1, v1> -> map -> <k2, v2> -> reduce>

<k3, v3> (output)

Most of the parallel Association Rule Mining (ARM) algorithms are based on parallelization of Apriori [15,16] Apriori algorithm already finds all frequent itemsets by introduce parallelization using map reduce model [17], an improved Apriori algorithm is proposed [18]. The algorithm give frequent itemsets by traversing transaction database only once [19]. The improved Apriori algorithm is implemented with MapReduce Programming model [19]. It gives good result using mapreduce programming model compared to non parallel apriori algorithm.

Equivalence Class Partitioning

Let's take an example based on which we are going to use the candidate generation step using equivalence class partitioning.. Let L_2= {PQ,PR, PS, PT,PU,RS, RT, RU TU}. Then

C_3={PQR,PQS,PQT,PQU,PRS,PRT,PRU,RST,RSU} Assuming that Lk-1 is lexicographically sorted, we can partition the itemsets in L_{k-1} into equivalence classes based on their common k-2 length prefixes, i.e., the equivalence class P $\mathbf{\Sigma}L_{k-2}$ is given as:

S_P = **[P] = {r$\mathbf{\Sigma}$** L_{k-1} | p [1 : k- 2] = r[1 : k-2]}

Candidate k-itemsets can simply be generated from itemsets within a class by joining all pairs. For our example L_2 above, we obtain the equivalence classes:

S_P = [P]={PQ,PR,PS,PT,PU}PU}

S_R = [R] ={RS,RT,RU},

and S_T = [T] = {TU}

We observe that itemsets produced by the equivalence class [P], namely those in the set PQR, PQS,PQT,PQUPRS, PRT, PRU are independent of those produced by the class [R] the set RST, RSU Any class with only 1 member can be eliminated. Since no candidates can be generated from it. Thus we can discard the class [T]. This idea of partitioning Lk-1 into equivalence classes was independently proposed in [20].

Vertical Database Transformation: Horizontal to Vertical in Map-Reduce Model

We have to transform the local database from the horizontal format to the vertical tid-list format. This can be achieved in two steps. First, each processor scans its local database and constructs partial tid-lists for all the frequent 2- itemsets. Second, each processor needs to construct the global tidlists for itemsets in its equivalence classes. Each processor thus needs to send tid-lists for those itemsets belonging to other processors, while receiving tid-lists for the itemsets it is responsible for. So here while converting from horizontal to vertical we partitioned the database into the number of nodes available for computation in the above example 3 nodes are available so we portioned the database of 8-transactions into 3 nodes, then the <key, value> pairs is key consist of the item and value consist of the Transaction ID (tid). Map reduce group everything by Key, So in the map phase we make the <key, value pairs> for efficiency in communication and data send over the network we do a local reduction on the key-value pairs. After that two nodes are selected for the reduce phase lets say node 5 and node 6, the reducer specifies which key it will reduce so node 5 will only reduce a,b,c. And node 6 reduce d,e,f,g. So the reduce phase is complete. Now system). And this frequent teo keys are again partitioned based on <key, value> pairs is key consist of the item and value consist of the Transaction ID (tid). Map reduce group everything by Key, so in the map phase we make the <key, value pairs> for efficiency in communication and data send over the network we do a local reduction on the key-value pairs. After that two nodes are selected for the reduce phase lets say node 5 and node 6, the reducer specifies which key it will reduce so node 5 will only reduce a,b,c. And node 6 reduce d,e,f,g. So the reduce phase is complete. Now system). And this frequent teo keys are again partitioned based on equivalence classes and distribute them among the participating processor for map reduce phase and after finding next level frequent items we synchronize them among the processors. This process continues until we get no frequent item. We can combine the reduced data from node-5 and node-6 and find 2-itemset so that we can initialize the equivalence class eclat algorithm using map-reduce. After this the Frequent 2 items are make global for synchronization among the processors. This redistribution process is done by HDFS (Hadoop distributed file system).

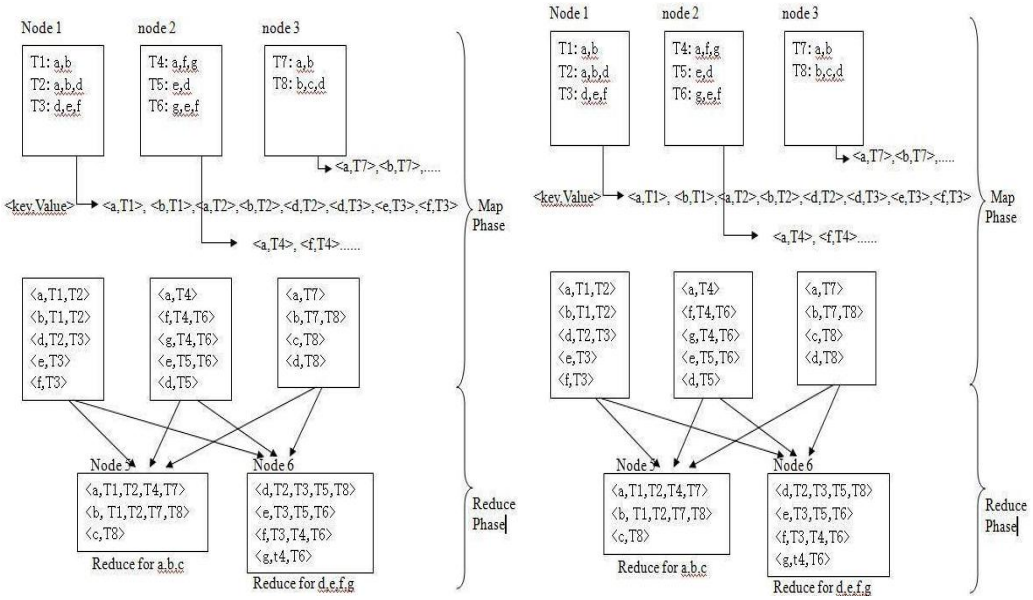

Figure 1: Map-Reduce Technique for Horizontal to Vertical Database

III. ANALYSIS OF THE ALGORITHM

To find how much of parallelism this algorithm, first we have to find the part of algorithm that can not be parallelize i.e. the strand [15], here intersection of any two items PQ and QR to produce PQR can not be parallelize more, so the asynchronous phase can not be parallelize. Now creating a directed acyclic graph G= (V,E), if a strand has two successor one of them is spawned (on mapped) and after computation a synchronization (or reduce) statement join them back. Thus, set V contains strands, and the set of edge E represent directed edge of dependencies Strands induced by parallel control Figure 2 shows the tree.

In figure 2 A,C,D, T are 4 items, A's equivalence Class forms AC,AD,AT which can put into one processor. The rectangle boxes represent the itemsets that can be put into a separate processor. As we are assuming we have P processor.

where P>n, where n is the total items

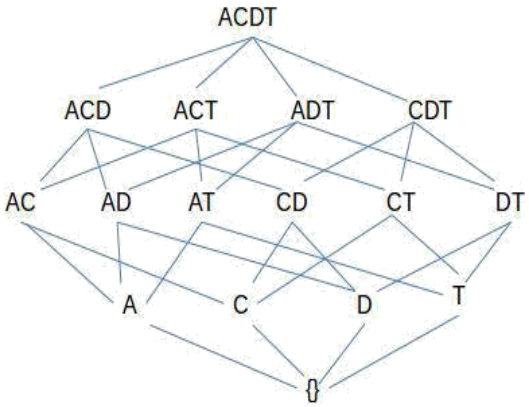

Figure 2: ACDT Tree

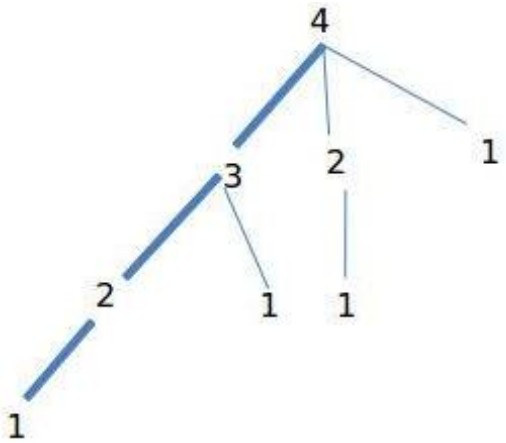

Figure 3: Tree of Figure 3

Figure 3 represent the tree from figure 2, in no of items allocated to a processor. as we can see in figure 3 level 1(considering levels from below) can be assign to four processor and then equivalence class of A can be assign to another processor which have three items in it, like wise we created the tree. If we consider T1 (n), running time on a single processor with n items then T1(n) will be O(2n) as the items forms a power set.

Now if we have ∞ no of processors to parallel the algorithm then we have to generalize the figure 3.this is shown in next figure 4. In figure3 the dark line shows critical path, that is, it is the largest necessary path where successor is dependent on it's parent.

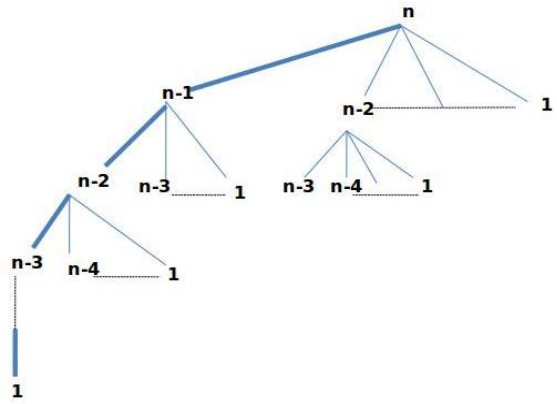

Figure 4: Generalization of the Tree Shown in Figure 3

And as we can see it is the height of the tree. For a four item system the height is four and as from figure 3 we can say that the critical path length is n, so if we have $T\infty(n)$ i.e. time required when we have infinite many processor and n items will be O(n). The ratio $T_1/T\infty$ gives the parallelism of the algorithm, which is 2n/n very big parallelism. This ratio provide a limit on the possibility of attaining perfect linear speed up. Specifically, once the number of processors exceeds the parallelism, the computation can not possibly achieve perfect linear speed up [16,17].

In the above analysis we have considered a ideal scenario, without considering data movement time in the n/w and many more computational bottleneck, also we have considered spawning(or mapping)than goes way down with fewer intersection to perform the height of the tree will be shorter in reality as it is no good practice to parallelize the algorithm to that much granularity. It will actually slow the algorithm down as the initialization of a mapper (or a processor) is very costlier with respect to time.

IV. PROPOSED MODEL: MR_Eclat

At the end of transformation phase the database has been redistributed among processes, so that the transaction ID (tid) lists of all 2-itemsets in its local equivalence classes reside on the local disk. Each processor can independently compute all the frequent itemsets, and do not synchronize with other processors. We read the tidlists for 2-itemsets within each equivalence class directly from the disk. We then generate all possible frequent itemsets from that class and make the items global for all the participating processors using HDFS (hadoop distributed file system) before moving on to the next class. This step involves scanning the inverted local database partition only once. This benefit us from huge I/O savings and from the locality perspective as well. Within each equivalence class we look at all pairs of 2- itemsets, and intersect their corresponding tid-lists. If the cardinality of the resulting tid-list exceeds the minimum support, the new itemset is inserted in L_3.After this we redistribute them among the processors for make the result global. Then we split the resulting frequent 3-itemsets, L_3 into equivalence classes based on common prefixes of length 2. All pairs of 3-itemsets within an equivalence are intersected to determine L_4 This process of partitioning finding frequent items and global of the result is repeated until there are no more frequent k-itemsets to be found. Note that once L_k has been determined, we can delete L_{k-1}. We need main memory space only for the itemsets in L_{k-1} within one equivalence class. The Eclat algorithm is therefore extremely main memory space efficient.

Design Model

Eclat algorithm is proposed which is shown:

- For making parallel scan, we first divide the transaction database horizontally into n data subsets and distribute it to m nodes.

- Then each node scans its own data setstrans form the horizontal data to vertical database Format and generate set of Candidate itemset C_p. The support count of each Candidate itemset is set to 1.
- Now create all 2-itemset from the vertical database just created. We do not check the frequency of 1-itemset as it will be another database scan and 2-itemset (L_2) can be created just by one pass.
- This is followed by a sum-reduction among all the processors. After this a redistribution of the frequent items is done by HDFS for synchronization. At the end of the initial phase, all processors have the global counts of the frequent 2- itemsets, L_2 in the database.
- Now, L_2 is partitioned using the equivalence class partitioning. The partitions are then assigned to the nodes. So now the tid-lists of all 2-itemsets in its local equivalence classes reside on the local disk. Each processor can independently compute all the frequent itemsets, eliminating the need for synchronization with other processors.
- We then generate all possible frequent itemsets from that class before moving on to the next class. So we will check the intersection of two 2-itemsets and if it is greater then the minimum support then it will be put into the next levelHere redistribution of the frequent items is done by HDFS for make the result global to all the processors.
- And in the next level we get itemsets which can be divided into partitioned using equivalence class. In each level we have to global the result using HDFS. And it will go on until no further frequent itemset can be created on all the nodes.
- At the end we aggregate the result of all the nodes and get the final frequent items. From the above improved Eclat algorithm we may considerably reduce the time as in this algorithm we are getting frequent itemsets by traversing transaction database only once.

Combining Equivalence Class Implemented Eclat Algorithm and Map Reduce

In the map phase as the database is portioned with the equivalence class so for example node-1 get [P]={PQ,PR,PS,PT} i.e. equivalence class2,then we observe that itemsets produced by the equivalence class [P] namely those in the set {PQR, PQS, PQT, PRS, PRT, PST} so in the map-reduce model we will take <PQ, trans_ids> ∩<PR,trans_ids>, here the trans_ids list will be get intersected,and if the output of this intersection is more then the minimum support then <PQR,trans_ids> will be created. And so on for other member of the class. The procedure can be stated as:

Compute Frequent (E_k)

- For all itemsets I_1 and I_2 in E_{k-2}.If (($I_1 \cap I_2$)>= minsup).
- Add I_1 and I_2 to L_{k+1}
- Partition L_k into equivalence classes
- For each equivalence class E_k in L_k 6.Compute_frequent (E_k)

At the end we aggregate the result of all the nodes and get the final frequent items.

The MR_Eclat Algorithm

In our algorithm there are five phases.

Initialization Phase

In this phase we scan local database.

Transformation Phase

In the second phase

- Transform the database to vertical database.

Synchronous Phase

1. Compute local counts for all 2-itemsets 2.Construct global L_2 count Partition L_2 into equivalence classes 3. Schedule L_2 over the set of processors P

Asynchronous Phase

In third phase for each equivalence class E_2 in local L_2,*Compute Frequent(E_2)*

Reduction Phase

In the last phase of our algorithm, aggregate Results and Output Associations. After this again synchronous phase for L_3asynchronous phase for L_3 and reduction phase for L3 then again synchronous phase for L_4 and so on. The procedure of compute frequent E_2 can be stated as:

1. Compute Frequent (E_k) for all itemsets I_1 and I_2 in E_k2.If (($I_1 \cap \mathbf{I}_2$)>= minsup) add I_1 and I_2 to L_{k+1} Partition LK into equivalence classes for each equivalence class Ek in L_k Compute_frequent (E_k).

V. EXPERIMENTS AND RESULTS

In order to test the performance of the parallel data mining strategy, first experiment has done on Hadoop platform Experiment has been performed with 2 node with the software environment having Hadoop installed on Ubuntu. The performance of different data sets is tested on both MR_Eclat and NHadoop algorithm as shon in the following table.. This gives an idea on how different data set would perform for the algorithm. The threshold is taken so that time taken can be compared among the data sets each having different number of total number of transactions(as they will have different minimum support count).The time output is in millseconds which is converted to seconds for comparing the performance. Figure 5 shows the time effect of the parallel algorithm on Hadoop. As shown in the figure, the parallel algorithm has good performance on mining frequent itemsets from the mass data. We use different size of database for both the algorithms and find the frequent itemsets. The time taken for both the algorithm has very little difference. We implement our algorithm in JAVA.

Table 1: Dataset for the MR_Eclat and NHadoop Algorithm

Transaction	Items	Size (kb)	Time (Millisecond) MR_Edat	Time (Millisecond) NHadoop
84	**276**	**1.4**	**2000**	**2100**
168	552	2.8	2100	2250
336	**1104**	**5.6**	**2150**	**2300**
134	2208	11.2	2200	2400

As we see from the graph that both algorithms works well. There is a little difference between the implementation in hadoop and without Hadoop. We can contrast the performance of different data sets on single non Hadoop machine for the same algorithm implementation. The performance of the algorithm in single node is worse than Hadoop on double

node. The tasks like dividing data and writing result, so that it is available to all the participating nodes maps, the mapped key value to the reducer,etc. Because of double node due to lack of multiple processes the implementation results almost similar with the non hadoop éclat algorithm. If we use the same algorithm MR_Eclat for number of nodes having very large dataset it results in better compared to without hadoop implementationWe actually need lots more processes. The parallel algorithms can solve the problems in various domains and give us efficient throughput. Lots of parallel ARM algorithms are developed so far but some are strong in some points while others are strong in some others[25]Also most of the algorithms are designed for homogeneous system with static load balancing which is far from reality. Algorithm for heterogeneous system with dynamic load balancing is required to develop with high performance. So, there is a great need of the algorithm with the minimum constraints discussed above. We proposed this algorithm which will try to find out the discovery of frequent itemset using eclat algorithm in less time complexity, less synchronization, less load balance and high throughput and we implement the algorithm

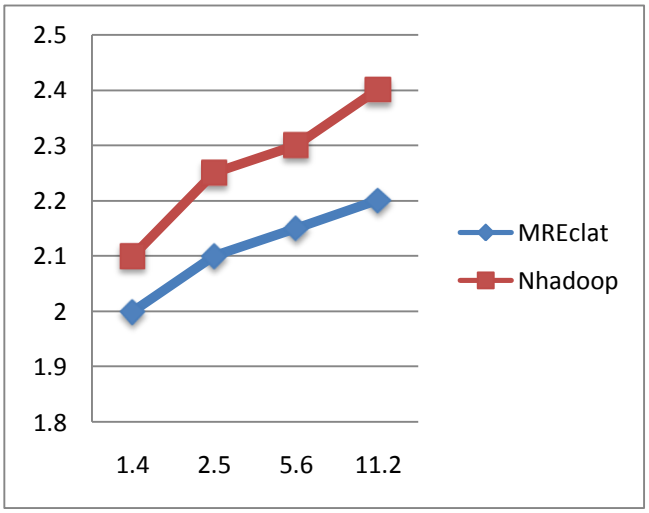

Figure 5: Graph Showing the Performance of Algorithm for Data Set Single Node Hadoop System and on Single Node Non Hadoop System

X axis:Size of database(kb)

Y axis:Time(Second)

VI. CONCLUSIONS

Parallel ARM algorithms are developed so far but some are strong in some points while others are strong in some others Also most of the algorithms are designed for homogeneous system with static load balancing which is far from reality. Algorithm for heterogeneous system with dynamic load balancing is required to develop with high performance. The key factors that affect the parallelism varies for different problems or even different algorithms of the same problem For example, due to the dynamic nature of association rule mining, the workload balance is a big issue for many algorithms that use static task scheduling mechanisms As data accumulates in large volumes and goes beyond the processing power of single-processor machines, parallel data mining techniques become more and more important for scientists as well as business decision-makers to extract concise, insightful knowledge from the collected information in an acceptable amount of time. In this paper we discussed a modified éclat algorithm MR_Eclat which employs a MapReduce approach. This approach is the underlying technique for hadoop, which is very robust and scalable The implementation of association rule in the distributed systems can be efficiently done on Hadoop. It can scale up to large data set with comparatively less cost and provide good performance. It can also cater to the distributed nature of the input data. The input data is divided among

the nodes. Further, the data transfer among the nodes and the situations like a storage node dies or, what would happen if some nodes in the cluster does not run, are taken care by Hadoop. This adds a great deal of robustness and scalability to the system.

VII. LIMITATION

The present algorithm can be made more effective by experimenting on number of splits, so that each map runs. It shouldn't happen that some maps are idle while other is over worked because of huge data being allotted to it. But if we implement our algorithm in multi nodes it results more effectively. The algorithm works perfectly fine with increased number of data and resources [21]. A small data set may not give good performance in the given set up. It might give worse or similar run time than the non-Hadoop system for the same algorithm.

VIII. FUTURE RESEARCH AND RECOMMENDATION

In future we will implement our algorithm using number of nodes for parallel the database so that more of items are screened out of candidate items in the initial stage in different nodes which are fasterSo that we can run our big database in a very small amount of time.

REFERENCES

1. Heikki, Mannila. 1996. Data mining: machine learning, statistics, and databases, IEEE.
2. R. Agrawal, T. Imienski and A. Swamy, Database Mining: A Performance Perspective, IEEE Tran. On Knowledge and Data Engg., December,1991. [3]M-S Chen, J Han and P. S. Yu, Data Mining: An Overview from a Database Perspective, IEEE Tran. On Knowledge and Data Engg., December, 1996.
3. A.Y. Zomya, T.E Ghazawi and O. Frieder, Parallel and Distributed Computing for Data Mining, IEEE Concurrency, Oct./Nov. 1999
4. Jong Soo Park, Ming-Syan Chenand Philip S. Yu. An effective hash-based algorithm for mining association rules. In Proceedings of 1995.
5. Piatetsky-Shapiro, Gregory. 2000. The Data-Mining Industry Coming of Age. IEEE Intelligent Systems.
6. Rakesh Agrawal, Tomasz Imielinski, and Arun N. Swami.Mining association rules between sets of items in large databases. In Proc. of the ACM SIGMOD Int'l Conf. on Management of Data, pages 207–216, May1993
7. Jiawei Han, Jian Pei, and Yiwen Yin. Mining frequent patterns without candidate generation. In Proc. of the ACM SIGMOD Int'l Conf. on Managementof Data, pages 1–12, May 2000.
8. Mohammed J. Zaki. Parallel and distributed association mining: A survey.IEEE Concurrency, 7(4):14–25, 1999
9. Nirali R. Sheth, J. S. Shah, International Journal of Advanced Research in Computer Science and Electronics Engineering Volume 1, Issue 3, May 2012.Implementing Parallel Data Mining Algorithm on High Performance Data Cloud
10. J. Dean and S. Ghemawat**,** ―MapReduce: simplified data processing on large clusters, communications of the ACM, vol. 51, Jan. 2008, pp. 107–113
11. Lingjuan Li and Min Zhang, The Strategy of Mining Association Rule Based On Cloud Computing, 2011 IEEE International Conference on Business Computing and Global Informatization.

12. Lingjuan Li and Min Zhang, The Strategy of Mining Association Rule Based On Cloud Computing, 2011 IEEE International Conference on Business Computing and Global Informatization.

13. Dhruba Borthaku. "The hadoop distributed file system: Architecture and design".

14. J. Han, and M. Kamber, 2000. Data Mining Concepts and Techniques. Morgan Kanufmann.

15. C. Borgelt. Efficient Implementations of Apriori and Eclat. Proc. 1st IEEE ICDM Workshop on Frequent Item Set Mining Implementations (FIMI 2003, Melbourne, FL). CEUR Workshop Proceedings 90, Aachen, Germany 2003. [18]

16. M. Zaki, S. Parthasarathy, M. Ogihara, and W. Li. New Algorithms for Fast Discovery of Association Rules. Proc. 3rd Int. Conf. on Knowledge Discovery and Data Mining (KDD'97), 283–296. AAAI Press, Menlo Park, CA, USA 1997.

17. R. Agrawal and J. Shafer. Parallel mining of association rules. In IEEE Trans. on Knowledge and Data Engg., pages 8(6): 962–969, 1996.

18. M. J. Zaki, M. Ogihara, S. Parthasarathy, and W. Li. Parallel data mining for association rules on shared-memory multiprocessors. In Proc. Supercomputing'96, Nov. 1996.

19. A compendium on Data Mining Algorithms and future comprehensive Manalisha Hazarika Mirzanur Rahman International Conference on Recent Advances in Mathematical Statistics and Its Applications in Applied Sciences December 31, 2012 - January 2, 2013