Study of Various Parallel Implementations of Association Rule Mining Algorithm

Sarbani Dasgupta[1], Banani Saha[2]

[1]*Department Of MCA,Techno India College Of Technology*
Kolkata 700156,India
only4sarbani@gmail.com

[2] *Department of ComputerSc and Engineering,University of Calcutta*
Kolkata 700009, India
bsaha_29@yahoo.com

Abstract-In data mining, Apriori technique is generally used for frequent itemsets mining and association rule learning over transactional databases. The frequent itemsets generated by the Apriori technique provides association rules which are used for finding trends in the database. As the size of the database increases, sequential implementation of Apriori technique will take a lot of time and at one point of time the system may crash. To overcome this problem, several algorithms for parallel implementation of Apriori technique have been proposed. This paper gives a comparative study on various parallel implementation of Apriori technique .It also focuses on the advantages of using the Map Reduce technology, the latest technology used in parallelization of large dataset mining.
Keywords: Data Mining, Association rules, Apriori, Hadoop , Map Reduce;

I. Introduction

Data mining is the process of autonomously extracting useful information or knowledge from large data stores or data sets. Association rule mining is a very popular data mining technique and it finds relationships among the different data in datasets. Frequent itemsets are a set of items, that appear frequently together in a transaction dataset. Apriori Algorithm is a well-known sequential association rule mining algorithm used for mining frequent itemsets for association rules. However with the increase of vast amount of data both in volume and dimension, application of sequential algorithms

will consume lot of time. Moreover a single processor machine does not have enough memory to hold such huge amount of data. Considering the problem of sequential algorithms, various parallel algorithms have been proposed for association rule mining. Some of the parallel algorithms are Count Distribution(CD),DataDistribution(DD),Candidate distribution[6],[7],[15].However there are

many challenges associated with these algorithms. Most of the problem occurs in case of communication and synchronization. Another approach for parallel implementation of association rule mining is the use of MapReduce technology for improving the Apriori algorithm. This paper gives a comparative study on various parallel implementation of association rule mining including the application of MapReduce framework on traditional Apriori Algorithm using Hadoop platform.

In this paper Section 2 discusses about basic concept of Association Rules Mining, Section 3 gives the overview of various parallel Association Rule Mining, Section 4 elaborately discusses about the Map reduce technology on Hadoop platform and Parallel Implementation of Apriori Algorithm based on Map Reduce technology. Section 5 gives a comparative study among various parallel implementation of Apriori Algorithm.

II. Basic Concept of Association Rules Mining

The association rule mining is used to extract the valuable knowledge from large-scale databases or datasets. Given I= {I_1, I_2…I_m} be an itemset. Let D as a transactional database where each transaction T is a nonempty itemset such that T⊆I.A unique identifier, called TID, is assigned with each transaction. We say that a transaction T contains X, a set of some items in I, if X ⊆ I. The association rules is of the following format: X=>Y, where X ⊆ I, Y ⊆ I, and X∩Y=Φ.

2.1 Support

The rule X=>Y has support s in the transaction set D if s% of transactions in D contains X ∪ Y. It describes the probability that the union set of itemsets X and Y appear in transaction database. 2.2 Confidence

The rule X => Y holds in the transaction set D with confidence c if c% of transactions in D that contain X also contains Y. It describes the probability that itemsets X and Y appear synchronously in transaction database.

An itemset is called frequent in D if its support in D exceeds a given minimal support threshold min_sup.

In general association rule mining can be viewed as a two step process:

1. Find all frequent itemsets f_k: Each of the itemsets will occur at least as frequently as a predetermined minimum support count, min_sup.
2. Generate strong association rules from the frequent itemsets: These rules must satisfy minimum support and minimum confidence.

2.3 Apriori Algorithm

The Apriori Algorithm[19] is the well known algorithm in association rule mining. Apriori uses a "bottom up" approach. The algorithm terminates when no further successful extensions are found. Apriori uses breadth-first search to count candidate item sets efficiently.

The name of the Apriori algorithm is based on the fact that the algorithm uses prior knowledge of frequent itemset property which is that all nonempty subsets of a frequent itemset must also be frequent. The main idea is to find the frequent itemsets.

The process of the algorithm is as follows:

Step1. Set the minimum support and confidence according to the user definition.

Step2. Construct the candidate 1-itemsets C_1. Generate the frequent 1-itemsets f_1 by pruning some candidate 1-itemsets C_1 if their support values are lower than the minimum support.

Step3: Join the frequent 1-itemsets f_1 with each other to construct the candidate 2-itemsets C_2 and prune some infrequent itemsets from the candidate 2-itemsets C_2 to create the frequent 2-itemsets f_2

Step4: Step3 is repeated again and again until no more candidate itemsets C_k can be created.

III. Parallel Association Rule Mining Algorithms

For processing massive datasets, use of sequential algorithm will consume a lot of time so parallel algorithms have been implemented for handling massive datasets with large dimensions on different platforms and with

different configurations. Many parallel algorithms has been developed based on different criteria like load balancing, memory utilization and task decomposition.

3.1 The Count Distribution algorithm:

The Count Distribution algorithm follows a data-parallel paradigm according to which the transaction database is statically partitioned among the processing nodes, while the candidate set C_k is replicated at each node. At each iteration every node counts the occurrences of candidate itemsets within the local database partition. At the end of the counting phase, the replicated counters are aggregated, and every node builds the same set of frequent itemsets f_k. On the basis of the global knowledge of f_k, candidate set $C_{k+1}$for the next iteration is then built. Inter-Node communication is minimized at the price of carrying out redundant computations in parallel.

3.2 The Data Distribution algorithm:

The Data Distribution algorithm attempts to utilize the aggregate main memory of the whole parallel system. Not only the transaction database, but also the candidate set C_k is partitioned, in order to permit both kinds of partitions to fit into the main memory of each node. Processing nodes are arranged in a logical ring topology to exchange database partitions, since every node has to count the occurrences of its own candidate itemsets within the transactions of the whole database. Once all database partitions have been processed by each node, every node identifies the locally frequent itemsets and broadcasts them to all the other nodes in order to allow them to build the same set C_{k+1}.This approach clearly maximizes the use of node aggregate memory, but requires a very high communication bandwidth to transfer the whole dataset through the ring at each iteration.

3.3 The Candidate Distribution algorithm:

In case of Candidate Distribution algorithm, both the database and the candidate set are partitioned in such a way that it allows each processor to proceed independently. The rationale of the approach is to identify, as execution progresses, disjoint partitions of candidates supported by (possibly overlapping) subsets of different transactions. Candidates are sub-divided on the basis of their prefixes. This is possible because candidates, frequent itemsets, and transactions, are stored in lexicographical order. The approach may suffer from poor load balancing due to dependence on resulting candidate portioning schema. Once the partitioning schema for both C_k and F_k is decided, the approach does not provide communication or synchronization among the nodes.

IV. Parallel Implementation of Apriori Algorithm based on MapReduce Technology on Hadoop Platform

The above mentioned parallel algorithm suffers from communication and synchronization problem between the nodes. Since Apriori Algorithm is a well known algorithm for Association Rule Mining, so Map Reduce model is used as it automatically handles the failure hiding the complexities of fault tolerance from the programmer.

4.1 MapReduce Model

Google's MapReduce paradigm [3],[4],[7] is a distributed programming paradigm and an associated implementation to support distributed computing over large datasets. With the help of this technology a programmer without any experience in parallel and distributed system can easily utilize the resources of a large distributed system, since it hides the details of parallelization, fault-tolerance, locality optimization, and load balancing.

The ideas of map reduce technology originated from the map and reduce functions of the functional programming. The Map Reduce framework consists of large number of computers called nodes which are collectively referred to as cluster. The Map and Reduce functions of MapReduce framework are defined

with respect to data structured (key, value) pairs. Map () can be expressed as
Map (k1,v1) → list(k2,v2).
The Map function is applied in parallel to every pair in the input dataset (k1,v1). This produces a list of pairs for each call list (k2,v2). After that, the MapReduce framework collects all pairs with the same key from all lists and groups them together, creating one group for each key. The Reduce function is then applied in parallel to each group, which in turn produces a collection of values in the same domain:
Reduce(k2, list (v2)) → list(v3)
Each Reduce call typically produces either one value v3 or an empty return, though one call is allowed to return more than one value. The returns of all calls are collected as the desired result list.

4.2 Hadoop

Hadoop is a software framework of map reduce system which provides a distributed filesystem (HDFS)[1] that can store data across thousands of servers, which provides a means of running work (Map/Reduce jobs) across those machines, as well as running the work near the data.
The Hadoop Map/Reduce framework has a master/slave architecture. It has a single master server or jobtracker and several slave servers or tasktrackers, one per node in the cluster. The jobtracker is the point of interaction between users and the framework. Hadoop's Distributed File System (HDFS) is designed to reliably store very large files across machines in a large cluster.

4.3 PApriori algorithm

This algorithm[18] needs one kind of MapReduce job. The map function performs the procedure of counting each occurrence of potential candidates of size k and thus the map stage realizes the occurrences counting for all the potential candidates in parallel way. Then, the reduce function performs the procedure of summing the occurrences counts. For each round of the iteration, such a job is carried out to implement the occurrences computing for potential candidates of size k.
The input dataset of the map () is stored on HDFS as a sequence file of <key, value> pairs, each of which represents a record in the dataset. The key is the offset in bytes of this record to the start point of the data file, and the value is a string of the content of this record. The dataset is splitted and globally broadcasted to all mappers. Consequently, the occurrence computations are parallel executed. For each map task, once the items in the candidate itemsets occur in the transactions, the <'key', 1> pair will be outputted, where 'key' is the candidate itemsets. The input of the reduce ( ) is the data obtained from the map function of each host. In reduce step, all the values with the same key is summed up and the final result is obtained.

V. Comparison among different parallel implementation of Association rule mining:

The various parallel algorithms are compared with respect to the response time of the algorithm with the increase in the number of processors as well as storage capacity. Figure 1 shows the response time of the four algorithms with respect to the datasets of Table 1.

Name	T	I	D_1	D_{16}	D_{32}
D4587K.T5.I2	5	2	3278K	52448K	104896K
D3107K.T10.I2	10	2	2016K	32256K	64512K
D3102K.T10.I4	10	4	2016K	32256K	64512K
D2367K.T10.I4	15	4	1456K	23296K	46592K

Table 1:Datasets Parameters

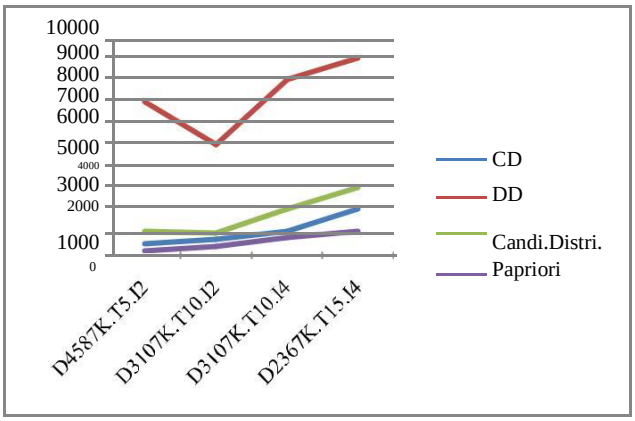

Figure 1:Performance of the four algorithms

Both Count Distribution (CD) algorithm and Candidate distribution algorithm shows that the response time is near about the time taken by the serial algorithm. The overhead of the Count Distribution algorithm is less than that of the serial algorithm. But some overhead occurs due to waiting for synchronization amongst the processor. Candidate distribution algorithm communicates with the entire dataset during the redistribution pass. It performs only one redistribution pass. Data distribution algorithm does not perform well as the other two algorithms because of the overhead of extra communication among the nodes. In Data Distribution algorithm every node processes each and every single transaction. It is almost entirely CPU-bound. CD shows less overhead. But the synchronization cost can increase in case of CD if the dataset is distributed among large number of nodes and the nodes are not equally capable. PApriori algorithm provides efficient memory utilization, minimization of communication and synchronization among different nodes, load balancing among various processes. The main advantage of the algorithm is that it automatically handles failure hiding complexity and fault tolerance from the programmer. The parallel algorithms can be compared with respect to the scaleup, speedup and size up criteria as shown in Figure 2.The dataset c of Table 1 is used to show comparison among the algorithm using 4,8,16,32 cores.

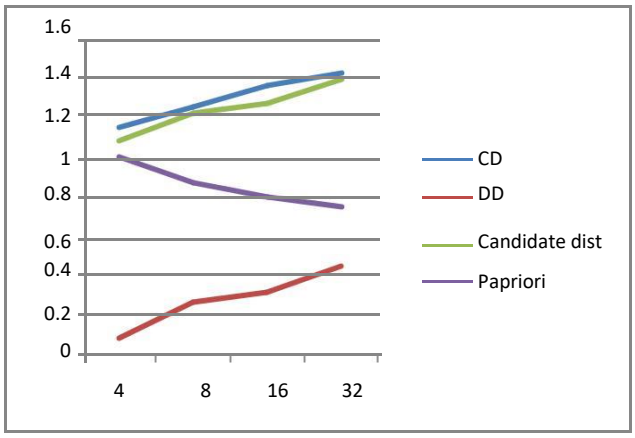

Figure 2:Scaleup for dataset D4587K.T5.I2

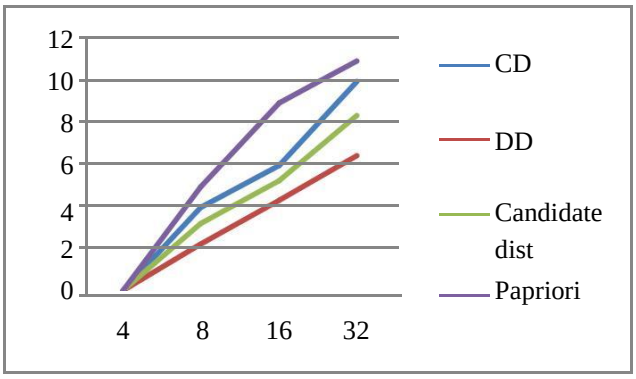

Figure 3:Sizeup for dataset D4587K.T5.I2

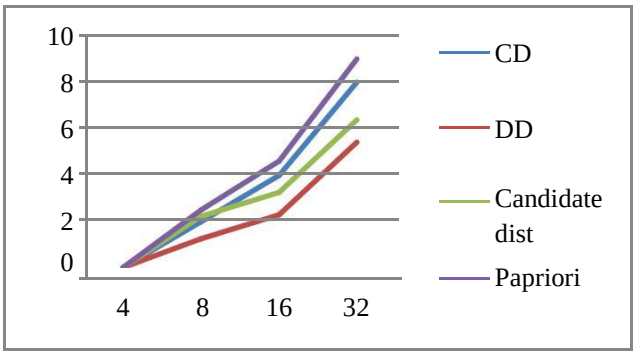

Figure 3:Speedup for dataset D4587K.T5.I2

The CD algorithm shows better performance with respect to the above factors as shown in the Figure 1, Figure2 and Figure 3. The other two algorithms shows an inferior performance. PApriori algorithm performs even better than the CD because it reduces the percentage of overall time spent in communication among the processors.

The parallel algorithms can be classified based on load balancing, data layout, memory system and the type of parallelism as shown in Table 2

Algorithm	Load Balancing	Data layout	Memory System	Typeof parallelism
CD	Static	Horizontal	Distributed	Data
DD	Static	Horizontal	Distributed	Data
Candidate distributio n	Static	Horizontal	Distributed	Task
PApriori	Dynamic	Horizontal	Distributed	Task

Table 2:Comparasion among different algorithms

Discussion

In Count Distribution algorithm each processor generates the same set of global frequent itemsets at each step. As a result it degrades the performance of the system because suffers from replication of calculation. Moreover, communicating the generated candidate set at each step increases the cost of the system. Data distribution Algorithm also suffers from communication overhead as well as overhead of partitioning the transaction among various processors. In case of the Candidate distribution algorithm cost increases due to redistribution of the transaction database. In PApriori algorithm it has been experimentally found that if the size of the dataset increases the algorithm will perform better. The performance of the PApriori algorithm has been measured based on several factors like scale up , speedup and sizeup. As the algorithm performs well on increasing the size of the datasets so PApriori algorithm can be consider as a more efficient algorithm among all the distributed algorithms discussed in this paper.

VI. Conclusion

Apriori is the simplest sequential Association Rule Mining. It has many drawbacks. Specially when the size of the data increases sequential algorithm may slow down the system. So many parallel algorithms have been developed like Count distribution, Data Distribution and Candidate Distribution. Sometimes these algorithms faces communication and synchronization problem. To overcome this problem Apriori algorithm is implemented with MapReduce technology where communication and synchronization problem is minimized between the nodes as well as hides the details of parallelization, fault-tolerance, locality optimization, and load balancing. In this paper a comparative study has been done between different the parallel implementation of the Apriori Algorithm. The results shows that the parallel implementation of Apriori Algorithm using MapReduce technology implemented on Hadoop platform shows better result.

References:

[1] White Tom, ”Hadoop :The Definitive Guide”, O’reilly,3rd edition ISBN: 978-1-449-31152-0

[2]Associationrulelearning.http://en.wikipedia.org/wiki/Association_rule_learning

[3] J.Ekanayake, S.Pallickara, G.Fox. Map-Reduce for Machine Learning on Multicore. IEEE International Conference on In eScience, 2008.

[4] J.Ekanayake, S.Pallickara, G.Fox. MapReduce for data intensive scientific analyses. Proceedings - 4th IEEE International Conference on eScience

[5] Lammel, R. Google’s MapReduce Programming Model - Revisited.Science of Computer Programming 70, 1–30, 2008

[6] Agrawal Rakesh, shafer John C.. Parallel Mining of Association Rules,IEEE transactions on knowledge and data engineering, Vol. 8, No.6,pp.962-969,1996

[7] Yanbin Ye, Chia-Chu Chiang, A Parallel Apriori Algorithm for Frequent Itemsets Mining, Proceedings of the Fourth International Conference on Software Engineering Research, Management and Applications (SERA’06), pp. 87-93,2006

[8] M. J. Zaki, “Parallel and Distributed Association Mining: A Survey”, IEEE, 1999, pp. 2, 3, 13, 14, 15.

[9] R. Agarwal and J. Shafer, “Parallel mining association rules”, IEEE Trans. OnKnowledge and Data Engg., 8(6):962-969, December 1996, pp. 4-6, 14.

[10] M. J. Zaki, S. Parthasarathy and W. Li., "Parallel data mining for association rules on shared memory multi-processors". In Supercomputing 96, Pittsburg, PA, November 1996, pp. 17-22.
[11] M. J. Zaki, S. Parthasarathy, M. Ogihara and W. Li, "New algorithms for fast discovery of association rules", in Proc. of 3rd Int'l. Conference on Knowledge Discovery and Data Mining, August 1997, pp. 283-296.
[12] Masaru Kitsuregawa and Takahilus Shintani, Masahisa Tamura and Iko Pramudiono, "Parallel Data Mining on large scale PC Cluster", H. Lu and A. Zhou (Eds.): WAIM 2000, © Springer-Verlag Berlin Heidelberg 2000. LNCS 1846, pp. 15–26, 2000.
[13] A. Mueller, "Fast sequential and parallel algorithms for association rule mining: A comparison". Technical CS-TR-3515, University of Maryland, College Park, August 1995, pp. 1-5.
[15]Rakhi garg,P.K Mishra,Exploiting Parallelism in Association Rule Mining Algorithm,International Journal of Advancement in Technology ISSN 0976-4860
[16]Xin Yue Yang,Zhen Lu,Yan Fu, Map Reduce as Programming Model for Association Rules Algorithm on Hadoop,IEEE Conference.
[17] Neng Li,Li Zeng,Qing He and ZhongzhiShi,Parallel Implementation of Apriori Algorithm Based on MapReduce,ACIS International Coference on Software Engineering,Artificial Intellegence,Networking and parallel Distributed Computing 2012
[18]Han Jiawei, Micheline Kamber ,Pei Jian, "Data Mining Concepts and technique",3rd edition,2012,ElsevierISBN978-93-8093