Distributed Frequent Itemset Mining using Hadoop

Dr. Dineshkumar B. Vaghela
Information Technology, Shantilal Shah Engineering College, Bhavnagar

Abstract-*In the present days, distributed environment has been growing and gain more attention in the data mining area. In the data mining area, frequent pattern mining is the most active research topic. In this paper, a survey on frequent itemset mining in distributed environment has been discussed. The frequent itemset mining and association rule mining has been growing so fast in the environment of distributed data mining. For frequent itemset mining and association rule mining number of algorithms are used. Some of those algorithms are discussed and their characteristics have been shown through comparison matrix. Also the proposed algorithm is discussed. By using proposed system the issues of communication overhead as well as time reduction has been solved.*

Keywords-frequent itemset, ARM, trie, distributed mining

I. INTRODUCTION

Data mining is the extraction of hidden predictive information from large databases, is a powerful new technology with great potential to help companies as well as research focus on the most important information in their data warehouses. Data mining tools predict future trends and behaviors, allowing businesses to make proactive, knowledge-driven decisions.

Mining frequent itemset in the distributed environment is a distributed problem and must be performed using a distributed algorithm that does not need raw data exchange between participating sites.(Ansari et al. 2008)

Distributed data mining is the operation of data mining in distributed data sets. According to Zaki (1999), two dominant architectures exist in the distributed environments which are listed as distributed and shared memory architectures.

In distributed memory each processor has a private DB or memory and has access to it. In this architecture, access to other local DB is possible only via message exchange. This architecture offers a simple programming method, but limited bandwidth may reduce the scalability.

In distributed memory each processor has a private DB or memory and has access to it. In this architecture, access to other local DB is possible only via message exchange. This architecture offers a simple programming method, but limited bandwidth may reduce the scalability.

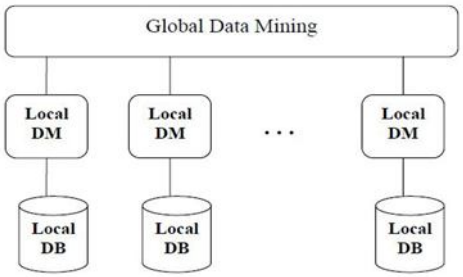


Figure 1: Distributed memory architecture for distributed data mining [11]

In the shared DB architecture, each processor has direct and equal access to the DB in the system (Global DB). Parallel programs can be implemented on such systems easily. The figure below indicates architecture for shared memory systems.

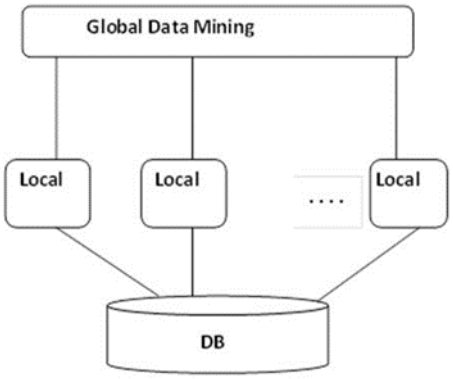


Figure 2: Shared memory architecture for distributed data mining [11]

The association rule mining (ARM) is very important task within the area of data mining Given a set of transactions, where each transaction is a set of literals (called items), an association rule is an expression of the form X ->Y, where X and Y are sets of items. The intuitive meaning of such a rule is that transactions of the database which contain X tend to contain Y.

Apriori is one of the most popular data mining approaches for finding frequent itemsets from transactional datasets.

II.GUIDELINES

In the following section we will discuss different algorithms proposed by different author in increasing order of efficiency. We will see the most basic distributed version of apriori and gradually we will discuss the comparison matrix and in future work we will propose one algorithm that will give less communication overhead and more efficiency.

1 .Count Distribution (CD): [6]

In this algorithm each site finds local support count of Ck Candidate itemsets in its own local database. Each Site exchanges its local support with other sites to obtain entire support for all candidate itemsets. Each site obtains the entire support for all candidate itemsets its local support with other sites to obtain the entire support for all candidate itemsets. Each site obtains LK, the entire frequent itemset, the candidate itemset with length of k+1 obtained from each site by execution of Apriori gen() function on Lk.

The CD algorithm's main advantage is that it doesn't exchange data tuples between processors-it only exchanges the counts. The algorithm's communication overhead is O (|C|*n) at each phase, where |C| and n are the size of candidate itemsets and number of datasets, respectively. [1]

2. Fast distribution mining (FDM): [7]

In each site play different roles, in the beginning a site consider as "home site" for produced set of candidate sets and subsequently it changes to a polling sites to get response time from other sites, it changes to a polling site to get response time from other sites, it become remote site. The different stages for FDM algorithm with consideration of different roles for each site.

FDM's main advantage over CD is that it reduces the communication overhead to O (|Cp|*n), where |CP| and n are potentially large candidate itemsets and number of sites, respectively. FDM generates fewer candidate itemsets compared to CD, when the number of disjoint itemsets among various sites is large. [1]

3. Optimized distributed association mining (ODAM):[1]

ODAM calculates 1-itemset from each site and broadcast those itemsets and discovers global frequent

1-itemsets.Each site generates candidate 2 itemsets and computes its support count and same time eliminate infrequent itemsets ODAM generates globally frequent 2 itemsets and iterates through main memory transaction and generates the support count of that respective length and the final frequent itemset is generated, The efficiency is more than CD and FDM algorithm.

The ODAM's message exchange size increases linearly as we increases the number of sites. It exchanges fewer messages than FDM.ODAM requires minimal number of comparison and update operation to generate support count.

4. Distributed trie frequent itemset mining (DTFIM):[2]

In this algorithm each site scans its local database and determines local count (1-itemsets) a vector is kept for making support count of every item. At end each site synchronize their data structure, using L1 the trie copies are alike. In second pass the candidate 2-itemsets are calculated,2-d array is used for this purpose, at end count are synchronized and global support count for candidate 2-itemset are calculated and trie copies are updated. From here on, in each pass k (k>=3) a candidate itemsets are calculated and the process is repeated, the pruning is performed simultaneously at each stage. The final output is frequent itemset. The complexity of DTFIM is O(n2) which is less than CD, FDM,ODAM. As the algorithm uses trie structure it is useful for pruning at local site.

5. Mining distributed frequent itemsets using a gossip Based protocol: [3]

In this algorithm, gossip based communication mechanism is used that is purely based on random communication between sites. The trie based apriori structure is used to improve the performance. First the local frequency is computed and support count is checked then after gossip Based global aggregation is done. In this algorithm improvement is achieved through employing trie data structure and grouping of nodes is done.

The complexity of this algorithm is O(nlogn) which outperforms all above algorithm. Gossip based communication can reduce the computation overhead of finding frequent itemset. The scalability is high as it requires minimum communication and comparison costs. As the complexity is low compared to DTFIM it is faster.

III.COMPARISION MATRIX

Algorithm	Complexity	Message passing overhead
CD	High(O(|ck|n) where ck is candidate itemset is number of site)	less
FDM	High, less than CD(O(|cp|n),where Cp is union of all local candidate Itemsets	less
ODAM	Low compare to CD,FDM($O(\|C_R + P(F_D\|*n)$, where C_R is the intersection of all local frequent itemsets, $P(F_D)$ is the total number of disjoint local frequent itemsets that have higher probability, and n is the total number of sites)	high
DTFIM	Low compare to CD,FDM,ODAM ($O(n^2)$)	high
GOSSIP BASED DISTRIBUTED FREQUENT ITEMSET MINING	Low compared to all above (O(nlogn))	high

IV.PROPOSED METHODOLOGY

After reviewing, we can propose such algorithm which will use trie structure and grouping methodology but it will locally prune dataset at base level as well as at intermediate level by grouping node into cluster. At the end final aggregation of frequent itemset is done at global level. This algorithm will have low complexity as well as message passing overhead.

From the comparison matrix one can say that a more work should be carried out for making the algorithm more efficient in the sense for reducing message passing overhead as well as issue of fault tolerance.

The problem associated with this approach is message passing overhead as it includes gossip based strategy for communication. The approach used is not working properly in the condition of fault. It means if any node fail taking part in communication then wrong result may be generated due to lack of communication. Naturally, for the reasons mentioned over will degrade the performance. So work can be carried out by addressing these problems on mind.

Overview of Proposed Work

The propose system can work out on reducing message passing overhead. Like in gossip based mining approach overall message passing overhead occurs because it follows random approach for communication and if any update occurs then the message passing should be occur randomly for all nodes while in our proposed strategy the global dataset generated is centralized so it will take only one message for one node to communicate.

The issue of fault tolerance is automatically resolved since we are using hadoop technology for implementation. Since hadoop works by name node and data node and replication is provided by this platform.so automatically efficiency is achieved.

Flow of Proposed Work

Our proposed algorithm will follow the steps starting from dividing the data nodes in clusters based on any nominal clustering criteria. The nodes that are geographically near area bounded in one cluster. First the local itemset for each node is calculated and stored in local trie. The same process is repeated for all nodes in cluster. At the end we will find intermediate global itemset for particular cluster.

Environmental setup:

For implementation of our propose algorithm hadoop distributed file system (HDFS) can be used.

It is used to perform operation on unstructured data, the data in form of logs. Data mining performs operation on structured data that is stored in form of tables in that row and column.

Apache hadoop is a framework that allows for distributed processing of large dataset across the clusters of commodity computers using simple programming model.

Hadoop components:

1) HDFS- That is used for storage in distributed environment.
2) Map Reduce-This approach is used for processing in distributed environment.

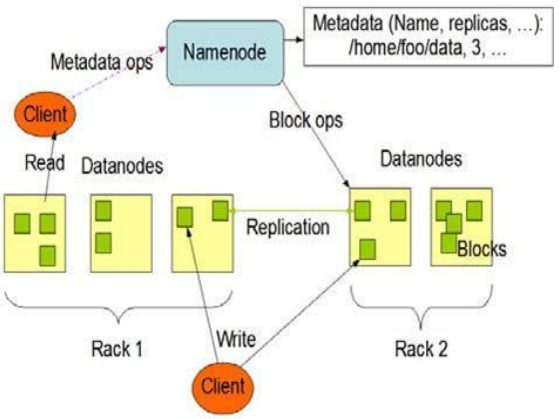


Figure 3: HDFS Architecture [15]

HDFS components:

Name node:

They are master of system. That maintains and manages the blocks which are present on data node. Its implementation requires expensive hardware.

Data node:

They are slaves which are deployed on each machine and provide actual storage. It is responsible for serving read and write request for client.

The implementation strategy followed is mentioned as below:

Proposed work can be deployed in distributed environment after installation of apache hadoop.

By implementing our propose system in single node cluster environment in which one Name node and one data node is there. Here single machine configuration is required but the problem with this approach is if name node fails then whole system fails same as if data node fails then also system crashes. Better way is to implementing our propose system in multi node clusters environment in which one name node and many data nodes are available. Here multiple machines can be configured together. Benefit of this approach is fault tolerance is achieved by having replica of name node as supporting name node and replication of data node can be achieved by name node as it stores the two or more copies of data node.

Outline of the Proposed Approach

The proposed methodology can be divided into two phases namely:

1. Local communication.
2. Grouping Nodes.

Local communication:

In this phase every node maintains a trie data structure. Nodes synchronize at end of each node so that all the nodes have same data at end of each round. The pruning operation is performed after the communication between the nodes is terminated. Hence as result the node maintains the similarity among them.

Grouping Nodes:

After the first phase, the local candidates along with their corresponding supports are sent to the central nodes. The central node at each group aggregates the candidate item sets and their supports for their groups. This phase incorporates to the concept of parallel processing as the central nodes from each group enter into the gossip for determination of the global support. This global support is taken into account for pruning the infrequent item set.

We can process these files parallel by placing the files on HDFS and running a Map Reduce job. The processing time theoretically improves by the number of nodes in the cluster.

The advantage of the Map Reduce abstraction is that it permits a user to design his or her data-processing operations without concern for the inherent parallel or distributed nature of the cluster itself.

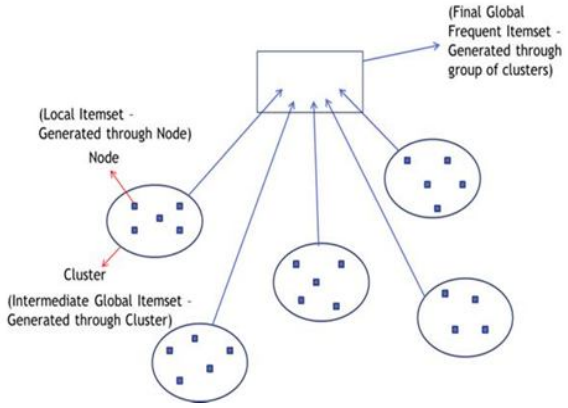


Figure 4: Proposed Methodology Process algorithm in Hadoop

(1) First the transaction database needs to be converted into Boolean matrix. If the transaction database contained m items and n transactions the Boolean matrix will have m+1 row and n+2 columns. The first column registers “items” and the first row records “TID” of the transactions.

(2) Second, the min-support is compared with the support of the item sets, if the support of the item sets is smaller than the min-support the row of the item sets will be deleted.

(3) To know the k-frequent item sets, the "AND" operation will just be carried out on the k rows. Finally all the frequent item sets can be found out.

(4) Usually there are a large number of transactions in the transaction database, so the Boolean matrix is very large. And the algorithm is carried out on the Hadoop platform.

(5) According to the number of the Data Nodes of the Hadoop, the Boolean matrix is divided into several parts based on the columns. And each part is located on each Hadoop Data Node.

(6) By doing this the algorithm can be executed parallel.

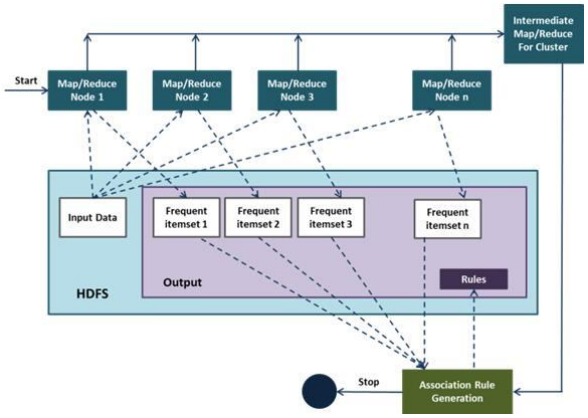


Figure 5: Data flow diagram showing n iterations of a cluster

The HDFS stores the input data as it can store huge chunks of data. HDFS also helps in data localizations for the map/reduce task. The input data is divided into parts and allotted to the mapper. The output from the mapper is key-value pair. The key is itemset and value is support count then output is passed the combiner. It combines all the count value related to a particular item which is known as key. The result from this is taken in by the reducer which combines and sum up of the values corresponding to an item. After getting the sum of values for an item the reducer is checked whether the value exceeds the given threshold value. If the value exceeds the threshold value for an item then the item with support count is written as the output. The item is discarded if it is less than the minimum support threshold value. This procedure would generate frequent-1 itemset. Frequent -1 itemset consists of set of one item pattern. The same process is repeated for generation of frequent itemset in all chunks of HDFS and then the results are combined in intermediate map reduce function the algorithm is applied again and final results will be generated.

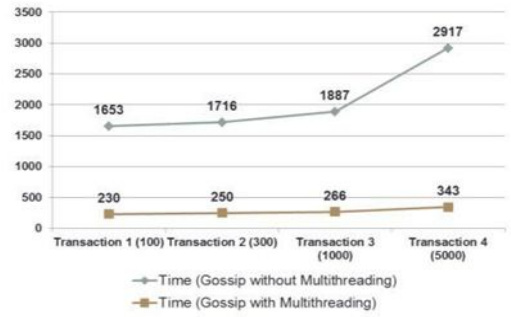
V.RESULT ANALYSIS

The major parameters to be consider are performance time and communication cost. Since the existing system suffers from more number of messages pass (i.e. n (n-1)) for getting final result and node failure.

Since hadoop resolves the issue of node failure by providing replication and data localization concept. The time taken by existing system for getting results is more which will be reduced by propose approach as it prunes infrequent itemset at local and intermediate level. By this way it gives better results as it need only n number of message passed for getting final result.so time will automatically reduce.

In experiments done during the implementation done on MUSHROOM dataset. In propose system also these data set will be used.

The mushroom dataset having approximately 8124 number of records with length of transaction 23 and has been applied to existing system. The graph shown below depicts that the performance by using gossip with multithreading is better than the performance without using multithreading.



The Proposed strategy with hadoop approach will generate better result than existing system due to the factors of reduced communication overhead and reduced time. In addition it provides reliability as protection against failure of node.

VI. CONCLUSION

In this paper, after reviewing various algorithms the conclusion can be drawn such as when we use the mostbasic algorithm in distributed environment the complexity is high as well as message passing overhead is high as we move forward

to higher algorithm the complexity decreases but the overhead is more. As CD is distributed version of apriori it is simple. The FDM is higher version with fast distribution while ODAM combines the CD and FDM and outperforms than both in terms of complexity as well as communication overhead. The DTFIM is using trie structure and it enables local pruning easy. The gossip based uses trie structure as well as gossip protocol for communication and gives highest performance.

The proposed system addresses the issues of communication overhead as well as time reduction. Due to the latest technology usage the issues of fault tolerance has been addressed. By this way the conclusion can been that hadoop technology and propose system can outperform then rest of the system.

REFERENCES

[1] Ashrafi, Taniyar and K.smith 2004,Optimized Distributed Association Rule Mining, IEEE distributed system online, 5:3
[2] E Ansari, M.keshatkaran march 2008,Distributed Trie Frequent Itemset Mining, IMECS Vol II
[3] Mining distributed frequent itemset using gossip based protocol,IEEE,2012 9th international conferenceFerenc bodon, A trie based apriori implementation for mining frequent itemset.ACM,NEW, NEW YORK, USA, 2005, 56-65
[4] A decentralized approach for mining event correlations in distributedsystem monitoring, J. Parallel Distrib. Comput. 73 (2013) 330–340
[5] Distributed Count Association Rule Mining Algorithm, International Journal of Computer Trends and Technology- July to Aug Issue 2011
[6] A fast distribution algorithm for mining association rule, Department of computer science, university of Hong KongMohammed j zaki Parallel and distributed association mining :A surveyPerformance Analysis of Distributed Association Rule Mining withApriori Algorithm, International Journal of Computer Theory and Engineering, Vol. 3, No. 4, August 2011
[7] Communication Efficient Distributed Mining of Association Rules, presented at ACM SIGMOD, BOSTON, MAY 2001
[8] Minor Thesis on association rule mining prepared by: ShamilaMafazi Supervisor: Abrar Haider, June 2010
[9] DH-TRIE Frequent Pattern Mining on Hadoop using JPA-2011,IEEE International Conference on Granular ComputingPSON: A Parallelized SON Algorithm with MapReduce for Mining Frequent Sets, 2011 Fourth International Symposium on Parallel Architectures, Algorithms and Programming
[10] Distributed Algorithm for Frequent Pattern Mining using HadoopMap Reduce Framework-ACEEE, Association of Computer Electronics and Electrical Engineers, 2013
[11] http://www.michael-noll.com/tutorials/running-hadoop-on-ubuntu-linux-single-node-cluster
[12] http://www.michael-noll.com/tutorials/running-hadoop-on-ubuntu-linux-multi-node-cluster/