ISSN (Online) 2394-2320

International Journal of Engineering Research in Computer Science and Engineering (IJERCSE)

Vol 4, Issue 11, November 2017

Frequent ITEMSET Mining Map Reduce

[1] Sanchita Sonar, [2] Shweta Kawad, [3] Karishma Murudkar, [4] Prof.Mrs.Sandhya S.Waghere
[1][2][3] BE Information Technology, [4] Assistant Professor
Pimpri Chinchwad College of Engineering, Pune, India

Abstract—Data mining deals with the extraction of hidden predictive information from large databases. Main task of data mining **is to discover action rules to take profitable decisions from huge database. However, with the exposing recent growth of size of data it is challenging to find out patterns of dataset. We have used a scalable approach for discovering action rules. The main goal of this algorithm is to build a mechanism that enables au-tomatic parallelization and data distribution for parallel mining of frequent itemsets on large clusters. We are proposing an algorithm inspired from FiDoop which run in-house Hadoop cluster. We are distributing dataset into number of blocks as each block allotted to each node for mining purpose. In this algorithm, we have proposed two parts, first part finds the frequent itemset in the dataset while the second part optimizes the efficiency of output. We also have optimized the mapper which gives high performance.**

Keywords—Association Rule Mining, Apriori, FiDoop, Map Reduce, FP-Growth.

I. INTRODUCTION

Now a day, finding out useful knowledge from the huge data is popular topic in recent years. Frequent itemset mining is process of finding the regularities between items. Basically it finds the set of items which frequently come together. It is mainly used in market basket analysis. It helps to take business decisions like which items should be selected for seal, what should be coupon policy and how to arrange the items on self in order to maximize the selling rate. During previous days several organizations used to store their data on databases. But, databases are not that much compatible or it doesn't provide the functionality that provides the exact related information that user wants. So to over-come this problem frequent itemset mining by consid-ering the association rules comes in to existence. Available methodologies for frequent itemset mining does not provide the high computational rate due to large amount of input and output data and this data is increasing day by day and so that Frequent Itemset Mining has become an important issue in sequence mining and association rule mining. To overcome this problem we are going to provide an algorithm which uses the Map Reduce programming to provide the load balancing and make system more efficient.

II. RELATED WORK

The idea of finding association rules was introduced by R. Agrawal and R. Srikant. Apriori algorithm was one of the ideal algorithm that was used to detail all the frequent itemsets. Angelina A. Tzacheva and Pranava Mummoju have given the MR-Apriori Count Distribution Algorithm for Parallel Action Rules Discovery in distributed environment. In this dataset is divided in number of blocks

[5]. Map Reduce programming model executes at each node. Map contained functions like sorting and shuf-fling. By using Apriori strategy, support and confi-dence calculates to discover the Action Rules [6]. Support is the percentage of records in dataset D which contains both A and B from the dataset.

$$\text{Support } (A \Rightarrow B) = P(AUB)$$

Confidence is the percentage of records in Dataset D which contains A where they already have B.

$$\text{Confidence } (A \Rightarrow B) = P(A|B) = P(AUB)/P(A)$$

Where,
D is a dataset containing all transactions.

A and B are subsets of itemset.
While working on huge dataset most common prob-lems are load balancing on server and time optimiza-tion. This problem can be overcome using FiDoop al-gorithm and Map Reduce programming model [2]. There are two algorithms which can be used for fre-quent itemset mining viz. Apriori and FP-growth.

A. Apriori Algorithm

Apriori is classic algorithm and uses the generation of large amount of candidate itemsets. It requires the scanning of database repeatedly. Basically it consists of two steps: candidate generation and pruning. To understand the Apriori algorithm let us consider following example.

Example of Apriori Algorithm-
Consider a database which contains the following transactions
Let the records be of 4 alphabets typed at random. The alphabets be {B,W,R,V}.

We need to find the frequently typed alphabets and consider the minimum support as 50%.

T1 : [B,W,R]
T2 : [B,R,V]
T3 : [B,V]
T4 : [B,W,R]

Minimum Support = 50%
= (50/100) * 4
= 2

Step 1: Calculate the frequency of each alphabet in the database.
Frequencies are-

B : 4
W : 2
R : 3
V : 2

We will prune all those alphabets whose frequency is less than 2. Since frequencies of all the alphabets is greater than or equal to 2, all alphabets are accepted.

Step 2 : Construct a pairs of alphabets and find the frequency of occurrence of pair.

Frequencies are-

{B,W} : 2
{B,R} : 3
{B,V} : 2
{W,R} : 2
{W,V} : 0
{R,V} : 1

We will prune all the pairs whose frequency is less than 2. Since frequency of {B,W,R} is greater than 2, this is the frequent alphabets set i.e. frequently occurring itemset

As shown in above example it works on the key con-cept that subset of the frequent itemset must be fre-quent. As the Apriori algorithm includes the candidate generation which involves the scanning of whole da-taset every time, it has very slow processing time. Can- didate generation is in the form of pairs or triples so this also decreases the speed and makes system slow. Sometimes in Apriori algorithm if implementation goes wrong then it produces duplicate of candidates. In addi-tion to this it consumes more memory.

B. FP-Growth Algorithm

While to solve this problem of scanning the whole database, another algorithm called FP-growth is used and it doesn't generate the candidate itemsets. FP Growth is nothing but Frequent Pattern Growth. It improves all the drawbacks of Apriori algorithm and proves that frequent itemset mining can be done without candidate generation. It first builds the FP-tree and then extracts frequent itemsets from these FP trees without scanning the whole dataset again and again. Now let us consider an example of FP growth to get clear idea about it

Example of FP-Growth algorithm-
Consider a database having 9 transactions {T1,T2,T3,…,T9} of 5 items {I1,I2,..,I5}.
We need to calculate frequent itemset .Given minimum support is 30%.

TRANSACTION ID	LIST
T1	I1,I2,I5
T2	I2,I4
T3	I2,I3
T4	I1,I2,I4
T5	I1,I3
T6	I2,I3
T7	I1,I3
T8	I1,I2,I3,I5
T9	I1,I2,I3

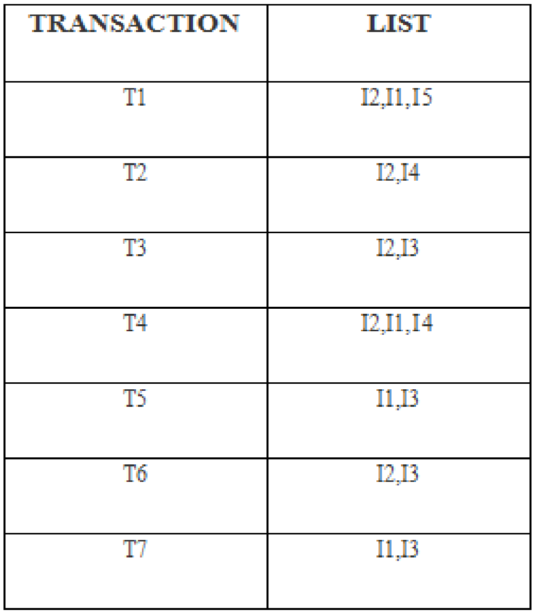
Step 1: Calculate minimum support 30%

Minimum Support = 30%
= (30/100) * 9
= 2.7~3

Step 2: Calculate priority of each item in database based on frequency of item.

ITEM	FREQUENCY	PRIORITY
I1	6	2
I2	7	1
I3	6	3
I4	2	4
I5	2	5

Step 3: Arrange the items in the transaction according to priority.



TRANSACTION	LIST
T1	I2,I1,I5
T2	I2,I4
T3	I2,I3
T4	I2,I1,I4
T5	I1,I3
T6	I2,I3
T7	I1,I3

Step 4: Construction of FP-Tree

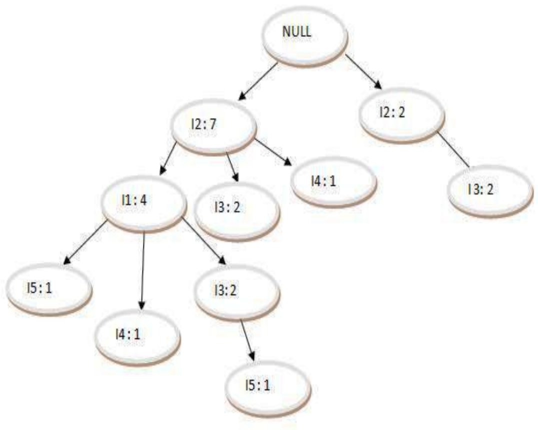


Fig. 1 FP Tree Example

If we compare FP-growth algorithm and Apriori algorithm then we observe that FP-growth algorithm is efficient than Apriori algorithm because of the following reasons:

1. FP-growth algorithm scans the dataset two times but the Apriori algorithm scans the da-taset many times.

2. There are no pairs and triples so no counting is involved in FP-growth algorithm and this makes it less complex and improve its speed.

3. It requires less memory consumption compared to Apriori algorithm.

But if we consider the huge and massive datasets then construction of FP-tree becomes difficult task. These existing parallel mining algorithms are unable to pro-vide automatic parallelization, effective load balancing, fault tolerance on large datasets. So to provide solution to this problems we design a parallel frequent itemset mining algorithm which uses Map Reduce program-ming to minimize the energy and remove irrelevant features from dataset to improve system's efficiency. Mapper provides parallelization to system by decom-posing itemsets in parallel while reducer combines these itemsets in frequent itemset trees and mining trees. We are going to implement our system on Ha-doop to provide load balancing by construction the nodes.

III. PRELIMINARY

In this section we will understand the basic concepts about Association Rule Mining (ARM) and Map Re-duce.

A. Association Rule Mining

In Association Rule Mining, frequent rules are found that define relations between unrelated frequent items in databases, and it has two main measurements: sup-port and confidence values [4]. Association rules have two parts: antecedent (if) and a consequent (then). An antecedent is an item found in the data. A consequent is found in combination with the antecedent. These rules are created by analysing data for frequent if/then pat-terns and using the criteria support and confidence to identify the most important relationships. Indication of how many frequent items appear in the database is called as Support. Number of times if/then statements have been found to be true is called as Confidence. The itemset that have support value greater than or equal to a minimum threshold support value, and frequent rules as the rules that have confidence value greater than or equal to minimum threshold confidence value are called as frequent item sets. The threshold values are traditionally assumed to be available for mining fre-quent itemsets. ARM is all about finding all rules whose support and confidence exceed the threshold, minimum support and minimum confidence values. Association Rule Mining consists of two main steps: first step is to find all itemsets with adequate supports and the second step is to generate association rules by combining these frequent or large itemsets. In tradi-tional system the threshold values are assumed to be known. It is very difficult to set the threshold values without any prior knowledge and to obtain the required results. Setting the threshold value very high may produce very small number of rules or else if the threshold value is set very low then it may produce big number of rules and it will take long time for computing the result. The output of these rules can be obtained in key-value pair and this output is then mapped. To map, Map and Reduce technique is used.

B. Map Reduce Framework

Map Reduce is a technique used for processing and it contains two important tasks, namely Map and Reduce. Map contains data and it breaks down the data into tu-ples (key/value pairs). Reduce takes the input from a mapper and combine those data tuples into a smaller set of tuples. As the name suggests the reduce task is al-ways performed after the map task. The advantage of Map Reduce is that it is easy to scale data processing over multiple computing nodes. Normally, it is consid-ered as non-trivial to decompose a data processing ap-plication into mappers and reducers but once we write an application in the Map Reduce form, scaling the application to run over hundreds, thousands, or even tens of thousands of machines in a cluster is merely a configuration change. Hadoop is used for distributed storage and processing of dataset of big data using the Map Reduce program-ming model. It is an open source software framework. It splits files into large blocks and distributes them across nodes in cluster. This uses the approach of data locality, where nodes manipulate the data they have access to. Hadoop Distributed File System (HDFS) stores data to provide high aggregate input/output bandwidth. Name Node is the master server that man-ages the file system namespace and regulates access to files. Data Node are the nodes on which actual data is stored and these nodes are managed by Name Node. Runtime system of Hadoop establishes two processes: Job Tracker and Task Tracker. Job Tracker is responsi-ble for assigning and scheduling tasks whereas each Task Tracker handles Map or Reduce tasks assigned by Job Tracker. Hadoop is widely adopted by various companies like Google, Facebook, Microsoft etc. main-ly because of features like HDFS and Map Reduce.

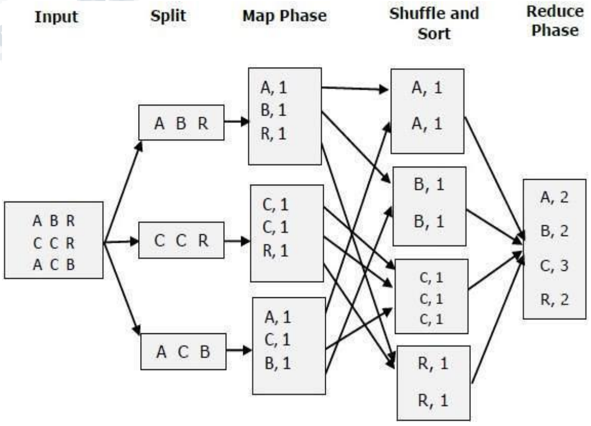


Fig. 2 Map Reduce process

IV. PROPOSED METHOD

In the light of Map Reduce framework and Association Rule Mining, we propose a parallel frequent itemsets mining algorithm. This algorithm is inspired from a

FiDoop algorithm. The goal of this algorithm is to build a mechanism that enables automatic paralleliza-tion and data distribution for parallel mining of fre-quent itemsets on large clusters. The algorithm consists of basically two parts: 1. FiDoop based construction of frequent itemsets 2. Optimization of output.

A. Frequent Itemsets

The process of finding frequent itemsets is based on Map Reduce and construction of Frequent Itemset Tree. This algorithm consists of two Map Reduce after which the algorithm constructs the Frequent Itemset Tree. It aims at improving the output of the algorithm and efficiency by incorporating the concept of FI-tree rather than traditional FP trees. Let us consider the al-gorithm in 2 phases: 1. First two Map Reduce 2. Tree Construction.

Phase 1

First Map Reduce

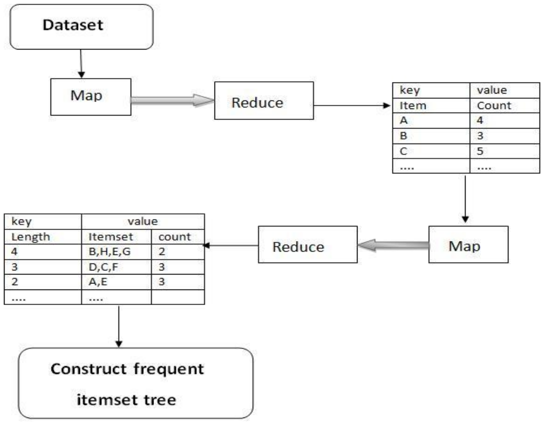
The dataset is provided as input to the first Map Re-duce and is responsible for creating all frequent item-sets. The dataset is split into multiple files and is stored by HDFS in data nodes on Hadoop cluster. The map-per computes frequency of each itemset in the file and generate local key value pair data which are stored lo-cally in files named F-list. These local pairs are then applied to reducer and then reducer generates the glob-al one itemsets. The infrequent pairs are pruned by ap-plying the minimum support to local pairs and thus the global itemsets are generated. Finally frequent one itemset along with its count is stored in files.

Second Map Reduce

The output of the first Map Reduce is given as input to Second Map Reduce. It performs a second round of scanning to prune the itemsets. In this step, the records are created of k frequent itemsets (2<k<M where M is maximal value of k in the pruned record). Mapper function emits the records containing k item in the itemsets which can be further combined. After performing the combination operation, each reducer emits key/value pairs, where the key is the number of each itemset and the value is each itemset and its count.

Phase 2

In this step k FI-trees are constructed. For construction of the trees the itemsets are decomposed obtained from second Map Reduce job into a list of small sized sets where the number of items in the set lies between 2 to k-1. The sets are the passed on to constructing trees by merging local decomposition results of same length. The decomposition of each mapper is independent of other mapper and this makes it highly scalable. This shows that multiple mappers can perform decomposi-tion in parallel. Third Map Reduce job is performed on this data where the Map function of the third job gener-ates a set of key/value pairs, in which the key is the number of items in an itemset and the value is an FIU-tree that is comprised of nonleaf and leaf nodes. Leaf nodes include item-name and its support and nonleaf nodes include item-name and node link. Itemsets with the same number of items are delivered to a single re-ducer. It mines all frequent itemset based on its count value.



key	value
Item	Count
A	4
B	3
C	5
....	....

key	value	
Length	Itemset	count
4	B,H,E,G	2
3	D,C,F	3
2	A,E	3
....	....	

Fig 3. Overview of Frequent Itemsets

Construction B. Optimization

Optimization of process is extremely important as if the length of itemsets to be decomposed is large, the decomposition cost will exponentially increase. The algorithm exhibits high performance for the dataset of small size. To process datasets of huge size it is necessary to optimize to mappers of the algorithm. As mentioned, the output of mappers is stored locally where all k-itemsets are recorded in a file named k-file and all (k−1)-itemsets are written to another file named (k−1)-file. Optimization algorithm decomposes the list of itemsets in a decreasing order of itemset length. After reading M-itemsets from a cache file, it decomposes the M- itemsets into a list of (M − 1) itemsets where M is the maximal length of itemsets. It then loads file to store (M−1)-itemsets, which are decomposed into (M−2)-itemsets to be unioned into the original (M−2)-itemsets. This procedure is iterated until the entire documenta-tion process is accomplished. The node sequentially loads and processes the files if multiple files are stored on a data node.

V. CONCLUSION

To overcome the challenges of existing system which lacks in distributed computing, we have proposed an algorithm inspired from FiDoop which runs on in-house Hadoop cluster. Map Reduce programming model is used for mining frequent itemsets from da-taset. We have proposed an algorithm which has two phases. In first phase, we find the frequent k-itemsets with its count and build FP-tree by merging of two smallest itemset. In second phase, we have proposed a technique for optimization which increases the effi-ciency of algorithm when the dataset is very big. We have optimized the mapper which gives high perfor-mance. This system provides scalability and parallel distribution features.

VI. ACKNOWLEDGEMENT

Our sincere gratitude to Mrs. Sandhya Waghere, Assis-tant Professor in Department of Information Technolo-gy for her valuable support and guidance.

REFERENCES

[1] Yaling Xun, Jifu Zhang, and Xiao Qin, “Fi- Doop: Parallel Mining of Frequent Itemsets Using MapReduce” in IEEE Transactions on Systems, Man, and Cybernetics: Systems, 2016

[2] Sandhya S Waghere, Pothuraju Rajarajeswari, “Parallel Frequent Dataset Mining and Feature Subset Selection for High Dimensional Data on Hadoop using Map-Reduce” in International Journal of Applied Engineering Research ISSN 0973-4562 Volume 12, Number 18 (2017) pp. 7783-7789.

[3] Sandhya S. Waghere, Pothuraju Rajarajeswari, “A Survey on Achieving Best Knowledge from Frequent Item set Mining using Fidoop” in In- ternational Journal of Computer Applications (0975 – 8887) Volume 171 – No. 9, August 2017.

[4] lugendra Dongre, Gend Lal Prajapati, S. V. Tokekar, “The Role of Apriori Algorithm for Finding the Association Rules in Data Mining” in 2014 International Conference on Issues and Challenges in Intelligent Computing Tech-niques (ICICT)

[5] Angelina A. Tzacheva, midhun M.Sunny and Pranava Mummoju , “ MR-Apriori Count Dis-tribution Aigorithm for Parallel Action Rules Discovery” in 2016 IEEE International Confer-ence on Knowledge Engineering and Applica-tions

[6] Sandhya Harikumar, Divya Usha Dilipkumar, “Apriori Algorithm for Association Rule Min-ing in High Dimensional Data” in 2016 IEEE International Conference on Data Science and Engineering (ICDSE)