ISSN (Online) 2394-2320

International Journal of Engineering Research in Computer Science and Engineering (IJERCSE)
Vol 5, Issue 4, April 2018

Modified Map Reduce Algorithm for Frequent Itemset Mining in Big Data

[1] K.Premchander, [2] S.S.V.N.Sarma, [3] Dr.S.Nagaprasad
[1] Research Scholar-Dravidian University, Kuppam, Andhra Pradesh
[2] Professor of CSE, Dept.of CSE, Vaagdevi College of Engineering, Warangal, Telangana State,
[3] Faculty of Computer Science, Dept. Of Computer Science, S.R.R.Govt.Arts & Science College, Karimnagar, Telangana State.

Abstract: **- Frequent Pattern Mining (FPM) is one of the most well-known techniques to extract frequent patterns from data. It plays an important role in association rule mining, finding correlations and trends etc. Finding Frequent Patterns becomes a very difficult task when they are applied to Big Data. Many researchers have proposed many algorithms to generate FIM, but the execution time and storage space plays a key difference .All the existing algorithms hold well only when the dataset is small. So there is a need to propose an efficient algorithm to find frequent itemsets from Big Dataset using constraints. In almost all FPM algorithms, Frequent 1-itemsets are generated to find the support count (occurrences) of each item in the entire database In order increase the efficiency of generating FIM, cache is introduced so that the support count can be calculated in the cache itself. For this a Modified Map Reduce algorithm has been proposed.**

Keywords**: Frequent Pattern Mining, Frequent Itemset Mining, Data Mining, Map Reduce Algorithm.**

I. INTRODUCTION

FPM means finding patterns (Itemset, sequence, structure, etc.) that occurs frequently in a data set. FPM helps us to identify the relationships or correlations between items in the dataset. For example, a set of items, such as paint and brush, which appear frequently together in a transaction data set, is a Frequent Itemset[1]. This information helps the shop keeper to arrange these frequent items together which will induce paint buyer to buy brush. Another example is Frequent Pattern discovery from Web Log data which helps to identify the navigational behaviors of the users. Consider the scenario, such as buying first a PC, then a Data Card, and then a Pen Drive, and if this pattern occurs frequently in a shopping history database, then that pattern is a frequent sequential pattern. Types of FPM are shown in Figure 1.

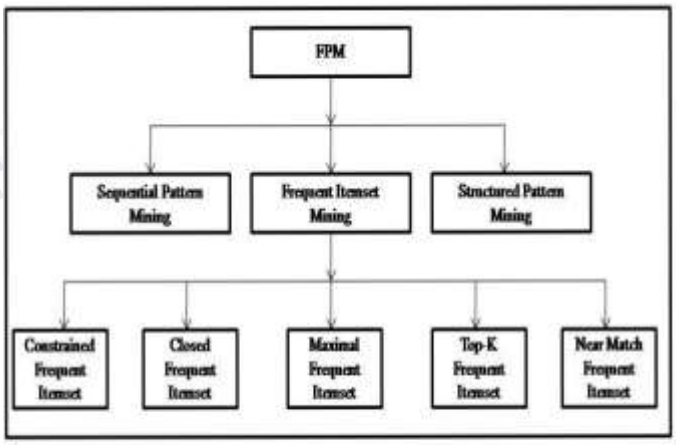

Figure 1.Types of FPM

Sequential Pattern Mining: It is concerned with finding statistically relevant patterns between data examples where the values are delivered in a sequence. The mining process finds frequent subsequences from a set of

ISSN (Online) 2394-2320

International Journal of Engineering Research in Computer Science and Engineering (IJERCSE)
Vol 5, Issue 4, April 2018

sequential data set, where a sequence records an ordering of events.

FIM: Extracting sets of products that are frequently bought together. It aims at finding regularities in the shopping behavior of customers of supermarkets, mail-order companies, on-line shops, etc.

Structured Pattern Mining: The mining process searches for frequent substructures in a structured data set. A structure is defined as a general concept that covers many structural forms, such as graphs, lattices, trees, sequences, sets, single items, or combinations of such structures.

II. PRESENTED SYSTEM:

FPM has proved to be one of the promising fields in carrying out the research work because of its wide use in all Data Mining tasks such as clustering, classification, and prediction and association analysis. Mining frequent itemsets enables humans to take better decisions in a wide range of applications including market basket analysis, traffic signals analysis and in Bioinformatics identify frequently co-occurring protein domains in a set of proteins[2]. Many researchers have proposed many algorithms to generate FIM, but the execution time and storage space plays a key difference in different algorithms means there is no efficiency of generating FIM.

III. PROPOSED SYSTEM

Modified MapReduce algorithm has been proposed. In this algorithm cache has been included in the Map phase to maintain support count tree for calculating the frequent-1 itemset of each mapper[3]. This reduces the total time of calculating Frequent-1 itemsets since it bypasses the shuffle, sort and the combine task of each Mapper in the original MapReduce tasks. This in-turn reduces the execution time of generating Frequent Itemsets of the entire database.

The flow chart of this algorithm is given in Figure 2.

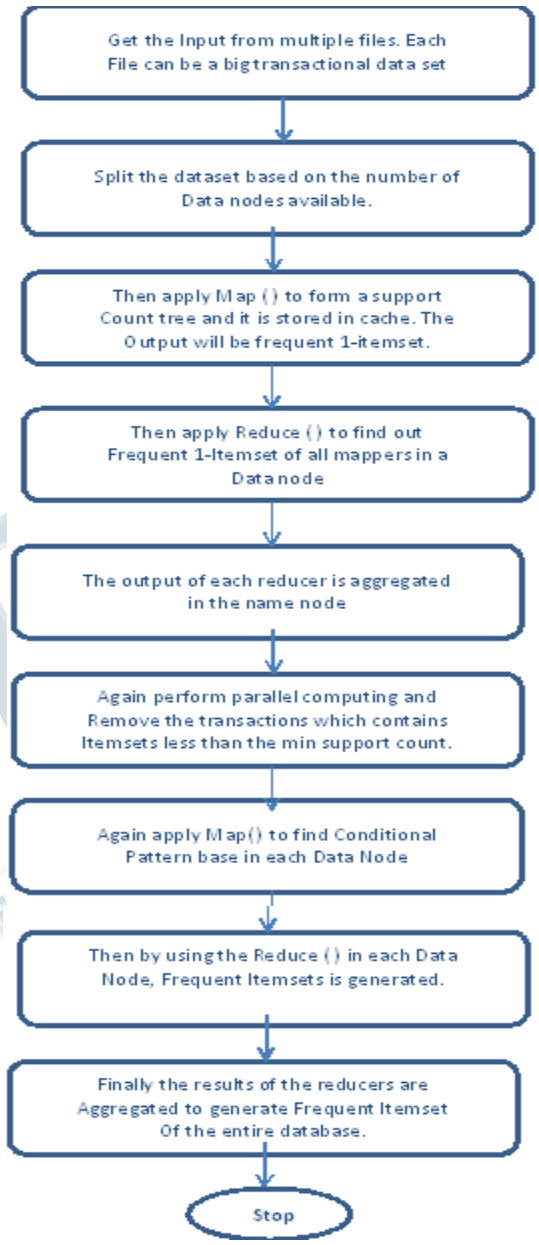

Figure 2. Flow Chart for FIM using Modified MapReduce

The initial step of frequent itemset generation is to generate Frequent 1-itemsets for the given database. For this support count tree algorithm has been proposed which is explained in detail in further section it has been shown how MapReduce is used to find frequent 1-itemsets and to generate frequent itemsets using constraints[4]. To increase the efficiency of map reduce task a cache has been included in the map phase to

ISSN (Online) 2394-2320

International Journal of Engineering Research in Computer Science and Engineering (IJERCSE)
Vol 5, Issue 4, April 2018

maintain support count tree for calculating the frequent-1 itemset of each mapper which is shown in Figure 3. As the data in cache can be quickly fetched it reduces the total time of calculating
Frequent-1 itemsets, since it bypasses the shuffle, sort and the combine task of each Mapper in the original MapReduce tasks.

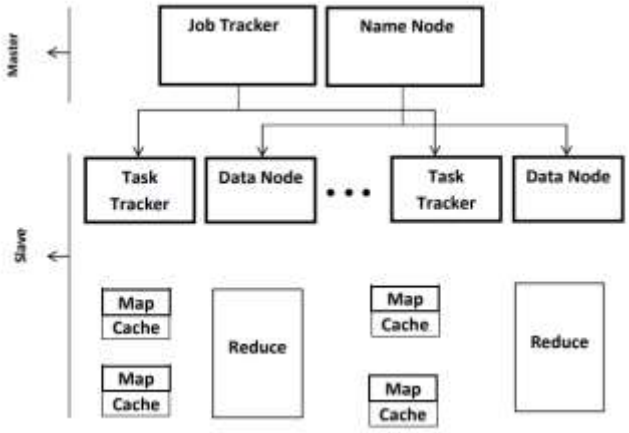

Figure 3. Proposed Architecture of MapReduce for generating frequent 1-itemsets

IV. MODIFIED MAPREDUCE

In each map function for finding the support count of each item the support count tree code has been embedded. The tree is stored in cache. As the items are read from the transaction database, it becomes easier to fetch the respective items data, as it is stored in the cache[8]. Thus at the end of map phase, the support count of each item is calculated by bypassing the sort and combine phase of the original MapReduce tasks which is shown in Figure 4 and Figure 5. The output of each Mapper is then given to the Reducer which finds the cumulative Frequent-1 itemsets of all mappers belonging to the same Data Node. The output is then stored in HDFS. In HDFS the outputs of the all the reducers are aggregated which gives the Frequent -1 Itemsets of the entire database.

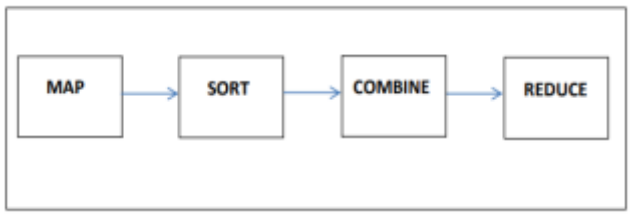

Figure 4.Flow Diagram of Map Reduce Task

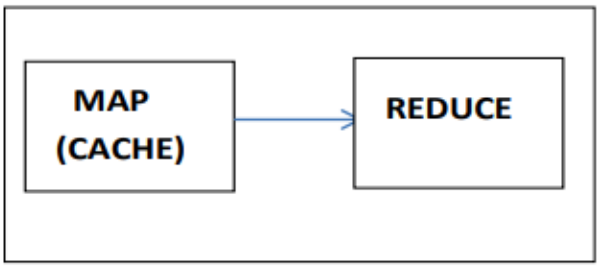

Figure 5 .Flow Diagram of Modified Map Reduce Task

Thus using cache and Support count tree the support count of each item is calculated quickly without undergoing sorting and combining steps[5]. Hadoop combiners require all map outputs to be serialized, sorted, and possibly written to disk. To overcome this, a cache has been introduced to store the frequent 1-itemset values.

Table1. Transaction Database 2

Transaction ID	Transactions
1	Z, Y, C, D, G, I, X, P
2	Y, B, C, Z, L, X, O
3	B, Z, H, J, O
4	B, C, Q, K, S, P
5	Y, Z, C, E, L, P, X, N

If the transaction database is given the item number, then the support count tree can be formed immediately. If not each item has to be numbered and then the support count tree has to be formed[9]. After finding the support count each item name has to be mapped to the item name and an example is shown below:

Table2. Numbering each item

Item Name	Item Number
B	1
C	2
D	3
E	4
G	5
H	6
I	7
J	8
K	9
L	10
N	11
O	12
P	13
Q	14
S	15
X	16
Y	17
Z	18

Next step is to form a support count tree. A support count tree for the above Table 2.is shown in Figure 5.

ISSN (Online) 2394-2320

International Journal of Engineering Research in Computer Science and Engineering (IJERCSE)
Vol 5, Issue 4, April 2018

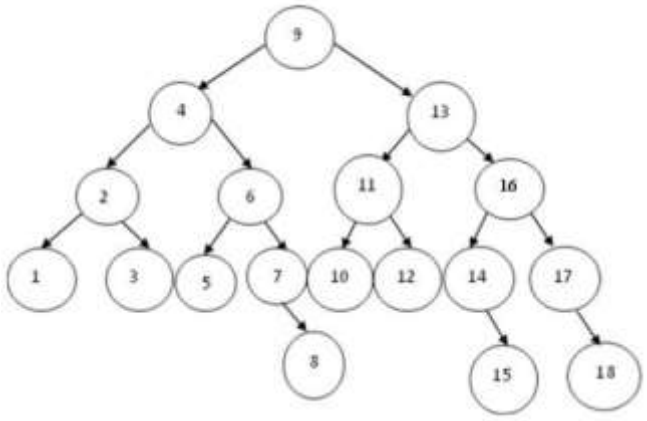

Figure 6. Support Count Tree

Each node in the support count tree has a count value associated with it. This gives the frequent 1-itemset of each item. Items whose support count is less than the minimum support threshold are removed and a Support Count Table (SCT) is formed which is shown in table 3.

Table 3 Support Count Table

Item Identifier	C	Z	Y	B	X	P
Support Count	4	4	3	3	3	3

V. RESULTS

The dataset which is being considered is T10I4D100K [36]. It contains 100,000 transactions of 3.93 MB with 999 different items. Each unique item in the dataset is considered as a node in the support count tree which has four attributes namely the name, count value, left link and the right link. The cache sizes for storing various numbers of items are given in Figure 7.

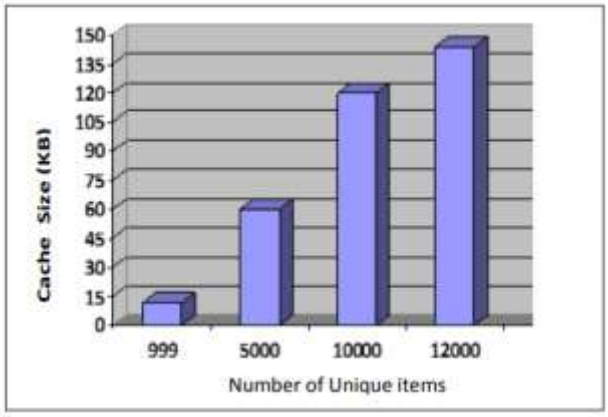

Figure7. Cache size required for storing different number of items

From Figure 8 and Table 4, it is clearly shown that the execution time to generate Frequent Itemsets using modified MapReduce is less when compared to the original MapReduce method[6]. The graph clearly shows that as the number of cores increases the execution time decreases considerably because the database is split evenly among the cores. So each core has fewer amounts of transactions to find frequent itemsets as the number of cores are increased. This in turn reduces the total execution time.

Table 4.Comparison of MapReduce and Modified MapReduce with respect to the execution time

S.No	Number of cores	Execution Time (sec) for MapReduce	Execution Time (sec) for Modified MapReduce
1	2	35	30
2	3	20	15
3	4	10	8

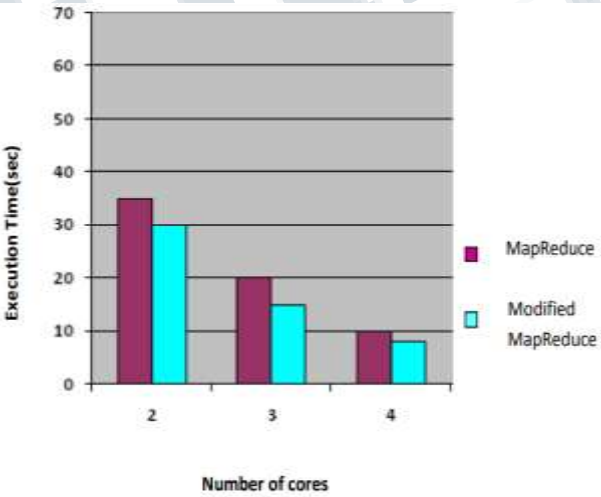

Figure8. Performance Comparison of MapReduce and Modified MapReduce

The database can be a single file or it can be from multiple files. If cumulative frequent itemsets are to be generated from multiple files that can also be done by appending the second file with the first file and the rest of the procedure is the same[7]. Two files namely T10I4D100K and T10I4D1000K have been considered. These two files are merged into a single file. Cumulative frequent itemsets for 1100000 transactions are generated which is shown in Figure 9.

ISSN (Online) 2394-2320

International Journal of Engineering Research in Computer Science and Engineering (IJERCSE)
Vol 5, Issue 4, April 2018

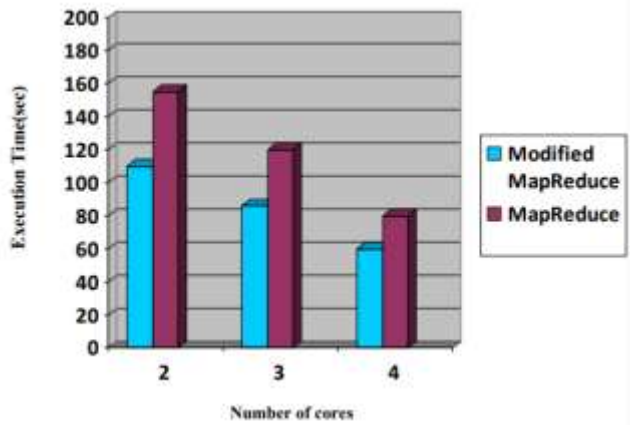

Figure9. Performance Comparison of MapReduce and Modified MapReduce for merged files

VI. CONCLUSIONS

In this paper we proposed a Modified Map Reduce algorithm in order increase the efficiency of generating FIM. In terms of memory as well as in execution speed, tree based Pattern growth algorithm is considered more most efficient than the other Frequent Itemset Mining (FIM) methods. Another important thing which is to be considered in Frequent Itemset Mining is that it often generates a very large number of itemsets, which reduces not only the efficiency but also the effectiveness of mining. So Constraint-based FIM has been proved to be effective in reducing the search space in the FIM task and thus improves the efficiency. In addition to this, in almost all Frequent Pattern Mining algorithms generates frequent 1-itemsets inorder to find the support count (occurrences) of each item in the entire transactions. Modified MapReduce algorithm has been proposed. In this algorithm cache has been included in the Map phase to maintain support count tree for calculating the frequent-1 itemset of each mapper. This reduces the total time of calculating Frequent-1 itemsets since it bypasses the shuffle, sort and the combine task of each Mapper in the original MapReduce tasks. This in-turn reduces the execution time of generating Frequent Itemsets of the entire database.

REFERENCES

1. Agrawal, R. and Shafer, J. C. "Parallel Mining of Association Rules", IEEE Transaction on Knowledge and Data Eng. , Vol. 8, No. 6, pp. 962- 96, 1996.

2. Agrawal, R. and Srikanth, R. “Fast algorithm for mining association rules”, International conference on Very large databases, 1994.

3. Alzoubi, W. A., Abu, Bakar, A. and Omar, K. “Scalable and Efficient Method for Mining Association Rules” International Conference on Electrical Engineering and Informatics, pp. 5-7, 2009.

4. Bakshi, K. “Considerations for Big Data: Architecture and approach”, in Aerospace conference, IEEE Aerospace Conference, pp. 1-7, 2012.

5. Banga, Devender and Cheepurisetti, S. “Proxy Driven FP growth based Prefetching”, International Journal of Advances in Engineering and Technology, 2014.

6. Do, T. D., Hui, S. C. and Fong, A. C. M. “Mining FequentItemsets with Category-Based Constraints”, In the proceedings of 6th International Conference on Discovery Science, 2003.

7. Dong, Jie and Han, M. “BitTableFI: An efficient mining frequent itemsets algorithm”, Knowledge based Systems, Elsevier, 2006.

8. Duggal, Puneet, Singh and Paul, S. “Big Data Analysis: Challenges and Solutions” in International Conference on Cloud, Big Data and Trust, RGPV, pp. 269-276, 2013.

9. Elteir, M., Lin, H. and Chun, Feng, W. “Enhancing MapReduce via asynchronous data Processing” in IEEE 16th International Conference on Parallel and Distributed Systems, pp. 397–405, 2010.