Improving the Performance of MS-Apriori Algorithm using Dynamic Matrix Technique and Map-Reduce Framework

Ms. Rachna Chaudhary
M. Tech. Scholar
Department of Computer Science and Engineering
Rajasthan Institute of Engineering and Technology, Jaipur, (Raj)

Mr. Sachin Sharma
Associate Professor
Department of Computer Science and Engineering
Rajasthan Institute of Engineering and Technology, Jaipur, (Raj)

Mr. Vijay Kumar Sharma
Associate Professor
Department of Computer Science and Engineering
Rajasthan Institute of Engineering and Technology, Jaipur, (Raj)

Abstract

Data Mining refers to the process of mining useful data over large datasets. The discovery of interesting association relationships among large amounts of business transactions is currently vital for making appropriate business decisions. This is the reason that the research in data mining is carried out largely for business decision making rather than for academic importance. Association rule analysis is the task of discovering association rules that occur frequently in a given transaction data set. Its task is to find certain relationships among a set of data (itemset) in the database. It has two measurements: Support and confidence values. Confidence value is a measure of rule's strength, while support value corresponds to statistical significance. There are currently a variety of algorithms to discover association rules. Most of the algorithms need a specification of minimum support value as user input. Specifying minimum support values of items is not recommended as it leads to very less or very large rules. With a sufficiently high support value, the less frequent elements gets eliminated, leaving only the elements which are most frequent. Thus, knives and spoons may get eliminated leaving only biscuits and milk. One approach for this problem is proposed by MsApriori Algorithm. However, both Apriori and MsApriori are computationally complex and need large computational time for large datasets over traditional machines. One solution to this problem is proposed by Dynamic Matrix Apriori which is much faster as compared to traditional Apriori in the generation of candidate sets. The contribution of this paper is twofold. It first proposed a method to use MsAprioiri using Dynamix Matrix Technique. It then proposes a framework to use the Algorithm under the Map Reduce Programming model. Experiments on large set of data bases have been conducted to validate the proposed framework. The achieved results show that there is a remarkable improvement in the overall performance of the system in terms of run time, the number of generated rules, and number of frequent items used.

Keywords: Apriori Algorithm, Association rule mining, Multiple Item Support, MapReduce

I. INTRODUCTION

Large quantity of data have been collected in the course of day-to-day management in business, administration, sports, banking, the delivery of social and health services, environmental protection, security, politics and endless ventures of modern society. Such data is often used for accounting and for management of the customer base. Typically, management data sets are sizable, exponentially growing and contain a large number of complex features. While these data sets reflect properties of the managed subjects and relations, and are thus potentially of some use to their owner, they generally have relatively low information density, in the context of association rule mining. Robust, simple and computationally efficient tools are required to extract information from such data sets. The development and understanding of such tools forms the core of data mining. These tools utilizes the ideas from computer science, mathematics and statistics.

The introduction of association rule mining in 1993 by Agrawal, Imielinski and Swami [1] and, in particular, the development of an efficient algorithm by Agrawal and Srikant [2] and by Mannila, Toivonen and Verkamo [3] marked a shift of the focus in the young discipline of data mining onto rules and data bases. Consequently, besides involving the traditional statistical and machine learning community, data mining now attracted researchers with a variety of skills ranging from computer science, mathematics, science, to business and administration. The urgent need for computational tools to extract information from data bases and for manpower to apply these tools has allowed a diverse community to settle in this new area. The data analysis aspect of data mining is more exploratory than in statistics and consequently, the mathematical roots of probability are somewhat less prominent in data mining than in statistics. Computationally, however, data mining frequently requires the solution of large and complex search and optimization problems [4] and it is here where mathematical methods can assist most. This is particularly the

case for association rule mining which requires searching large data bases for complex rules. Mathematical modeling is required in order to generalize the original techniques used in market basket analysis to a wide variety of applications. Mathematical analysis provides insights into the performance of the algorithms. An association rule is an implication or if-then-rule which is supported by data. The motivation given in [5] for the development of association rules is market basket analysis which deals with the contents of point-of-sale transactions of large retailers. A typical association rule resulting from such a study could be "90 percent of all customers who buy bread and butter also buy milk". Insights into customer behavior may also be obtained through customer surveys, but the analysis of the transactional data has the advantage of being much cheaper and covering all current customers. Compared to customer surveys, the analysis of transactional data does have some severe limitations, however. For example, point-of-sale data typically does not contain any information about personal interests, age and occupation of customers. Nonetheless, market basket analysis can provide new insights into customer behavior and has led to higher profits through better customer relations, customer retention, better product placements, product development and fraud detection. Market basket analysis is not limited to retail shopping but has also been applied in other business areas including:

1) Credit card transactions,
2) Telecommunication service purchases,
3) Banking services,
4) Insurance claims, and
5) Medical patient histories.
6) Economy, Stock Predictors

Association rule mining generalizes market basket analysis and is used in many other areas including genomics, text data analysis and Internet intrusion detection. For motivation, the focus is given on retail market basket analysis in this paper. When a customer passes through a point of sale, the contents of his market basket are registered. This results in large collections of market basket data which provide information about which items were sold and, in particular, which combinations of items were sold.

Association rule mining [1,2] is one of the most important and well-researched techniques of data mining, that aims to induce associations among sets of items in transaction databases or other data repositories. Currently, Apriori algorithms [1,2,6] play a major role in identifying frequent item set and deriving rule sets out of it. Apriori algorithm is the classic algorithm of association rules, which enumerate all of the frequent item sets. When this algorithm encountered dense data due to the large number of long patterns emerge, this algorithm's performance declined dramatically. In order to find more valuable rules, this paper proposes an improved algorithm of association rules over the classical Apriori algorithm. The improved algorithm is verified, the results show that the improved algorithm is reasonable and effective, can extract more value information.

A. Problem Statement

Association rule analysis is the task of discovering association rules that occur frequently in a given transaction data set. Its task is to find certain relationships among a set of data (itemset) in the database. It has two measurements: Support and confidence values [1, 2]. Confidence value is a measure of rule's strength, while support value is of statistical significance. Traditional association rule mining techniques employ predefined support and confidence values. However, specifying minimum support value of the mined rules in advance often leads to either too many or too few rules, which negatively impacts the performance of the overall system. In this paper, it is proposed to replace the Apriori's user-defined minimum support threshold with a more meaningful aggregate function based on empirical analysis of the database. Also, Classical Apriori Algorithm is inefficient in finding out association rules as the number of database operations is huge and every time the database is updated, the mining of association rules is to be performed from a fresh start. To overcome the issue of rescanning the database, Matrix Apriori algorithm [7] was proposed in which matrices MFI and STE are prepared to mine association rules. Matrix Apriori gives great performance improvement over classical Apriori. Further, to manage for the dynamic nature of transactional databases, dynamic matrix Apriori algorithm was proposed which manages the dynamic nature of the databases without recreation of the MFI and STE matrices. Although no provision was provided for accounting the custom defined support values for the datasets. This paper focuses on the issue of setting up custom support values over Dynamic Matrix Apriori algorithm for efficient association rule mining. It also proposes a framework for using MapReduce [8] technique in multiple node Cluster based environments so as to reduce time complexity for mining of association rules over bigdata.

B. Motivation

Association rule mining has become a popular research area due to its applicability in various fields such as market analysis, forecasting and fraud detection. Given a market basket dataset, association rule mining discovers all association rules such as "A customer who buys item X, also buys item Y at the same time". These rules are displayed in the form of $X \rightarrow Y$ where X and Y are sets of items that belong to a transactional database. Support of association rule $X \rightarrow Y$ is the percentage of transactions in the database that contain $X \cup Y$. Association rule mining aims to discover interesting relationships and patterns among items in a database. It has two steps; finding all frequent itemsets and generating association rules from the itemsets discovered. Itemset denotes a set of items and frequent itemset refers to an itemset whose support value is more than the threshold described as the minimum support. Since the second step of the association rule mining is straightforward, the general performance of an algorithm for mining association rules is determined by the first step. Therefore, association rule mining algorithms commonly

concentrate on finding frequent itemsets. For this reason, in most of the literature, "association rule mining algorithm" and "frequent itemset mining algorithm" terms are used interchangeably. Apriori and FP-Growth [9] are known to be the two important algorithms each having different approaches in finding frequent itemsets. The Apriori Algorithm uses Apriori Property in order to improve the efficiency of the level-wise generation of frequent itemsets. On the other hand, the drawbacks of the algorithm are candidate generation and multiple database scans. FP-Growth comes with an approach that creates signatures of transactions on a tree structure to eliminate the need of database scans and outperforms compared to Apriori. A recent algorithm called Matrix Apriori [7] which combines the advantages of Apriori and FP Growth was proposed. The algorithm eliminates the need of multiple database scans by creating signatures of itemsets in the form of a matrix. The algorithm provides a better overall performance than FP-Growth. Although all of these algorithms handle the problem of association rule mining, they ignore the mining for infrequent items as all these techniques employs user specified minimum support. Although, work on multiple minimum support has been carried out in recent years, no existing techniques addresses the issue of implementing multiple minimum support [10,11,12] over matrix Apriori algorithm.

The user specified support values are generally not optimal and a common minimum-support value cannot be assigned to all the items. Thus, there is a critical need to develop techniques so as to compensate for this issue.

This paper is orgranized as follows. Section 2 gives the objectives and research approach used in the paper. Section 3 discusses Matrix Apriori Algorithm and the techniques for evaluation of itemset classes for custom specified support values. It also presents the techniques through which the custom support values can be inculcated in matrix Apriori algorithm. It also proposes technique to use the MapReduce framework for mining of association rules. Section 4 shows the test results and the performance evaluations. The section begins with an illustration of walmart transaction database. Custom values are evaluated for datasets and association rules are mined. Also, 1, 2 and 3 frequent itemsets are mined based on the support values and the matrix apriori algorithm using MapReduce. Section 5 is the conclusion section. A summary of the paper and suggestions for future research are stated.

II. RESEARCH APPROACH

The objective of this paper is to provide a mechanism for user defined support values for various classes of data items for effective mining of association rules and at the same time, provide mechanism so that such technique can be inculcated into matrix Apriori technique so that multiple scans of database can be avoided and association rules corresponding to infrequent items can be accounted. It also aimed at the implementation of the proposed Dynamic Matrix MsApriori using the MapReduce framework so that the proposed algorithm can be executed in cluster based environments to mine association rules in real time on operational bigdata.

For each item of the item universe, the support value is evaluated using entire thorough scan of the database. The range of values of support of different items is tabulated in ascending order, thereby, providing a closed interval in which these support values lies. This range of support values is very large for most general market basket databases. The support values are maximum for everyday consumables like bread and butter and in contrast, very low for items such as food processor or television. Thus a common minimum specified support value cannot be assigned to all the items for association rule mining. One approach is to make classes of items belonging to specific range of support values over entire range of support values. Another approach is to provide custom specified minimum support values to be assigned to each item based on the average values of the support and the probability distribution of infrequent items. The derived Minimum Item Support (MIS) can be used in matrix apriori algorithm for generation of MFI and STE matrices. Also, techniques are provided so that these custom specified support values can be used with Matrix Apriori algorithm. A method is then proposed to inculcated dynamicity into the technique so that new transactions can be added into the existing transaction matrix and results can be mined. It then propose an architecture to use MapReduce for creation of MFI and STE for mining of Association rules over a cluster based system using the <key, value> pairs and the corresponding reduce operation.

III. PROPOSED WORK

A. Msapriori: Apriori With Multiple Minimum Support

Mining association rules with multiple minimum supports is an important generalization of the association-rule-mining, which was proposed by Liu et al. This generalization is named as Multiple Support Apriori or MsApriori. The implementation of MsApriori is straightforward except that it requires a large number of iteration as compared to standard Apriori.

The MsApriori algorithm can find rare item rules without producing a huge number of meaningless rules. Each item in the database can have its minimum support value called minsup, which is expressed in terms of minimum item support (MIS). Users can specify different MIS values for different items. By assigning different MIS values to different items, one can reflect the natures of the items and their varied frequencies in the database.

Let $I=\{a_1, a_2,..., a_m\}$ be a set of items and $MIS(a_i)$ denote the MIS value of item a_i. Then the MIS value of itemset $A=\{a_1, a_2,..., a_k\}$ $(1\leq k\leq m)$ is equal to

$$\min [MIS(a_1), MIS(a_2).... MIS(a_k)]$$

Consider the following items in a database, bread, shoes and clothes. The user-specified MIS values are as follows: MIS(bread) = 2%, MIS(shoes) = 0.1%, MIS(clothes) = 0.2%. If the support of itemset{clothes, bread} is 0.15%, then

itemset{clothes, bread} is infrequent because the MIS value of itemset{clothes, bread} is equal to min[MIS(clothes), MIS(bread)]=0.2%, which is larger than 0.15%.

B. Matrix Apriori Algorithm

The Matrix Apriori Algorithm is a frequent itemset mining algorithm that combines the advantages of both Apriori and FP-Growth algorithms. Resembling Apriori, algorithm Matrix Apriori consists of two steps. First, discover frequent patterns, and second, generate association rules from the discovered patterns. The first step determines the performance of the algorithm.

Let $L = \{i_1, i_2, \ldots, i_m\}$ be a set of items, *D be* a repository containing a set of transactions, δ a *minimal support* predefined by the user, *T* a transaction, where each transaction $T \subseteq L$. Algorithm Matrix Apriori employs two simple structures for generating frequent patterns: a matrix called MFI (matrix of frequent items) which holds the set of frequent items identified during the first traversal of repository *D*, and a vector called STE which stores the support of candidate sets. Frequent items will be represented by columns of MFI, however, they could also be represented by rows. The total number of frequent items is stored in variable NFI and the total number of candidate sets NCS.

Initially the database is scanned in order to determine frequent items. These items are sorted in a descending support count and trimmed to those that are above the minimum support value to create the frequent items list. The sorted frequent items list is the basis for the order of columns of the MFI. Subsequently, in the second scan, MFI and STE are built. The first row of the MFI is left empty. This row will be updated later in the modification. Therefore, inserting rows to MFI begins after this empty row. For each new transaction in the database, a row of zeros and ones is inserted according to the following rule. The row is constructed by using the order in the frequent items list. For each item in the list, either "1" or "0" is added to the column of the row if the transaction contains the item or not. If the transaction is already included in the MFI, then it is not stored again in a new row, but its STE is incremented by "1".

Consider the following illustration for Matrix Apriori algorithm:

Table - 3.1
Hypothetical Data of Transactions

Transaction Id	*ITEM SET*
001	*Coffee, dish-wash, eggs, gum, honey, ice-cream, ketchup, pasta*
002	*bread, eggs, flour, honey, ice-cream, pasta*
003	*coffee, eggs*
004	*Aluminum-foil, bread, Coffee, dish-wash, flour, honey, pasta*
005	*Aluminum-foil, bread, coffee, dish-wash, eggs, flour, gum, ice-cream, pasta*
006	*bread, eggs, flour, gum, honey, ice-cream, pasta*
007	*Aluminum-foil, bread, coffee, dish-wash, eggs, pasta*
008	*Aluminum-foil, Coffee, dish-wash, eggs, flour, honey, ice-cream, pasta*
009	*Aluminum-foil, Coffee, dish-wash, eggs, ketchup, pasta*
0010	*Aluminum-foil, Coffee, dish-wash, eggs, flour, ice-cream, pasta*

As specified earlier, the support count of an element is the number of times it appears in all the transactions. The tabulation of the support values of various items is shown in table 3.2.

Table - 3.2
Item Support Count

Item	*Support*
Aluminum-foil	*6*
Bread	*5*
Coffee	*8*
Dish-wash	*7*
Eggs	*9*
Flour	*6*
Gum	*3*
Honey	*5*
Ice-cream	*6*
Ketchup	*2*
Pasta	*9*

Let the minimum support value be 6, then 1- Frequent itemset list is the list of all the elements having the support values greater than or equal to 6. Table 3.3 lists frequent items sorted according to the list of frequency and eliminating those with less than minimum support.

Table - 3.3
Sorted List of Frequent Items with Minimum Support

Item	*Support*
Eggs	*9*
Pasta	*9*

Coffee	*8*
Dish-wash	*7*
Aluminum-foil	*6*
Flour	*6*
Ice-cream	*6*
Bread	*5*
Honey	*5*
Gum	*3*
Ketchup	*2*

The shaded items are the items which have support values less than the minimum specified support. These elements are to be eliminated from further analysis

It is important to note at this point that if the support values are used specified, as in the case of MsApriori, then the list of 1 frequent itemsets is to be prepared accordingly. The number of frequent items is stored in variable NFI; in this example, the number of frequent items is 7.

Table 3.4 lists the transaction id and transactions sorted by the order of support and elimination those elements which have less than the minimum support.

Table - 3.4
Transaction Lists with Sorted Support

Transaction Id	*Sorted Item-Set*
001	*Eggs, pasta, coffee, dish-wash, ice-cream*
002	*Eggs, pasta, flour, ice-cream*
003	*Eggs, coffee*
004	*Pasta, coffee, dish-wash, Aluminum-Foil, Flour*
005	*Eggs, pasta, coffee, dish-wash, Aluminum-Foil, Flour, ice-cream*
006	*Eggs, pasta, flour, ice-cream*
007	*Eggs, pasta, Coffee, dish-wash, Aluminum-Foil*
008	*Eggs, pasta, Coffee, dish-wash, Aluminum-Foil, Flour, ice-cream.*
009	*Eggs, pasta, Coffee, dish-wash, Aluminum-Foil*
010	*Eggs, pasta, Coffee, dish-wash, Aluminum-Foil, Flour, ice-cream*

Initially, MFI is prepared by second scan of the entire database.

Each column of MFI represents an elements and each row represents a transaction. The columns are assigned to the elements in the decreasing order of the support count as specified in table 3.5.

The binary representation of the transactions in terms of frequent items is shown in table 3.5

Table - 3.5
Transactions Represented As Binary Vectors

Items/ Transaction No.	*Eggs*	*Pasta*	*Coffee*	*Dish- wash*	*Aluminum-foil*	*Flour*	*Ice-cream*
001	*1*	*1*	*1*	*1*	*0*	*0*	*1*
002	*1*	*1*	*0*	*0*	*0*	*1*	*1*
003	*1*	*0*	*1*	*0*	*0*	*0*	*0*
004	*0*	*1*	*1*	*1*	*1*	*1*	*0*
005	*1*	*1*	*1*	*1*	*1*	*1*	*1*
006	*1*	*1*	*0*	*0*	*0*	*1*	*1*
007	*1*	*1*	*1*	*1*	*1*	*0*	*0*
008	*1*	*1*	*1*	*1*	*1*	*1*	*1*
009	*1*	*1*	*1*	*1*	*1*	*0*	*0*
010	*1*	*1*	*1*	*1*	*1*	*1*	*1*

Each of the row vectors is to be included in MFI as per the Matrix Apriori specification. This is performed as shown in table 3.5. The first row of MFI is left blank for computations to be performed later.

Transactions 1 to 5 are all distinct so added in the MFI as these are in Table 3.5 and Row candidate set is set to 1 for each of these transactions. Transaction 6 is identical to 5, so the count of STE for row 5 is incremented by 1. Transaction 7 is then added to the matrix in row no. 7. Transaction 8 is identical to transaction 5 so count is STE in row five is again incremented by 1. Transaction 9 is identical to transaction 7 so the STE row for 7 is incremented by 1. Transaction 10 is identical to 5 so the row 5 of STE is again incremented by1.

Table - 3.6
Initial Matrix of Frequent Items (MFI) and STE

MFI

	1	2	3	4	5	6	7
1							
2	1	1	1	1	0	0	1
3	1	1	0	0	0	1	1
4	1	0	1	0	0	0	0
5	0	1	1	1	1	1	0
6	1	1	1	1	1	1	1
7	1	1	0	0	0	1	1
8	1	1	1	1	1	0	0
	Eggs	Pasta	Coffee	Dish-wash	Aluminum-foil	Flour	Ice-cream

STE

Row Candidate Set	Count
1	1
2	1
3	1
4	1
5	1+1+1
6	1
7	1+1

The resulting values in matrix MFI and vector STE, after full traversal of repository *D*, are presented in table 3.6. The fourth candidate set (stored in row 5) has support equal to 3, that is, the candidate set is present in three transactions of repository *D*. This value is stored in the fourth position of vector STE. The next procedure to be executed is modified MFI, which begins its process in the first row of MFI and has the goal of writing indexes to accelerate the search for frequent patterns. In the first row of matrix MFI, the row number are stored where the *j*-th frequent item appears for the first time. For example, frequent item flour appears for the first time in the third row, hence MFI[1, 6] = 3. The value 3 inserted is thus an index, and following it, the row number where frequent item flour appears for the second time will be stored in row 3 of the matrix, that is in MFI[3, 6] the inserted value will be 5. The next row where flour is found is row 6, so this value is stored in MFI[5, 6]. Finally, element MFI[6, 6] stores the value 1 to indicate that no more rows contain item flour. Figure 3.6 shows the result obtained after modifying the MFI. The reason for insertion in 3, 5 and 6th row in case of element flour being this that these are the nonzero cells in the column vector of these columns.

Table - 3.7
Modified Matrix of Frequent Items (MFI) and STE

MFI

	1	2	3	4	5	6	7
1	2	2	2	2	5	2	2
2	3	3	4	5	0	0	3
3	4	5	0	0	0	5	6
4	6	0	5	0	0	0	0
5	0	6	6	6	6	6	0
6	7	7	8	8	8	7	7
7	8	8	0	1	0	1	1
8	1	1	1	0	1	0	0
	Eggs	Pasta	Coffee	Dish-wash	Aluminum-foil	Flour	Ice-cream

STE

Row Candidate Set	Count
1	1
2	1
3	1
4	1
5	1+1+1
6	1
7	1+1

The last step is to find Frequent Patterns, which takes as input matrix MFI and vector STE as presented in table 3.7. This procedure applies the method of growing patterns, basically consisting of combining frequent items, determining the support associated to this combination, and comparing the support of a combination with the minimal support to decide whether or not it is a frequent pattern.

In the process of searching for frequent patterns, item ice-cream is combined with the frequent items found on its left hand side (flour, aluminum-foil, dish-wash, coffee, pasta and eggs). The support for each of the combinations is calculated. For this purpose, all the cells of both the vector are analyzed for matching values. The matching values in the columns of both ice-cream and flour are 2, 6 and 7. The corresponding values in STE are 1, 5 and 6. This is important here to note that STE vector count is one less than the row count in MFI. The summation of the count of these entries in STE table is 1+3+1 = 5 which is the support value of this itemset. The support values if the two itemset, of element ice-cream with all other elements is listed in table 3.8

Table - 3.8
Combinations Obtained Based On Conditional Pattern i

Combined Items	*Support*
flour-icecream	*5*
Aluminiumfoil-icecream	*3*
dishwash-icecream	*4*
Coffee-icecream	*4*
pasta-icecream	*6*
eggs-icecream	*6*

Table 3.9 presents the sorted lists of frequent items having minimum support.

Table - 3.9
Combinations Obtained Based On Conditional Pattern i with Minimum Support

Combined Items	*Support*
pasta-icecream	*6*
eggs-icecream	*6*
flour-icecream	*5*
dishwash-icecream	*4*
Coffee-icecream	*4*
Aluminiumfoil-icecream	*3*

If the minimum support considered for any element pair be 6, then only the twp pairs; viz, <pasta, ice-cream> and <eggs, ice-cream> are considered as frequent pairs whereas the last three in the table, which are shown highlighted are discarded from further analysis.

In the next iteration of the search process, these two combinations will be taken as conditional patterns.

1) For conditional pattern <pasta, ice-cream>, there is only one item to the, namely item eggs, hence combination <eggs, pasta, icecream> with a support of 6, becomes a frequent pattern.
2) For conditional pattern <eggs, icecream>, no items exist to the left of eggs, therefore no combination can be generated.

Once the analysis on item ice-cream has concluded, the process continues with conditional pattern flour. Combinations are made with the items situated to the left of flour (aluminium-foil, dish-wash, coffee, pasta and eggs), obtaining the combinations with their corresponding support values.

Matrix Apriori works without candidate generation, scans database only twice and uses Apriori Property. Also, the management of matrix is easy. Also, Matrix Apriori can be tabulated with Multiple support values with slight modifications as per the specifications given in section 3.1.

C. Dynamic Matrix Apriori

It is impractical to scan database to construct MFI and STE every time the database is updated. Also, the transactional database of a typical market basket analysis frequently changes with a number of transactions being added every day. This gives motivation for development of techniques to inculcate dynamicity in the matrix apriori algorithm so as to dynamically generate association rules based on the overall transactional database under consideration.

Dynamic Matrix Apriori Works on the similar lines as the matrix apriori algorithm. When new transactions arrive, they are scanned and 1-itemset ordered list of items are updated to include the 'now frequent' items.

The new items in the additions are included to the MFI by adding new columns. The MFI is updated as follows:

First, the new transaction is checked whether it is existed in the MFI or not. If it exists, its STE is incremented by “1”; if it does not exist, it is added to the MFI. Adding to the modified MFI is done by setting the cell value to the row number of transaction where the “1” value is stored in the MFI. When the item does not exist in the remaining transactions of the incremental database, the cell value is set to “1”. Finally, according to the change of the 1-itemset ordered list of items, the order of items in the MFI is changed. Dynamic Matrix Apriori can also be processed for Multiple Minimum Support of the itemsets. The same sets of techniques are also applicable to implement Dynamic Matrix MsApriori.

D. MapReduce Framework for Dynamic Matrix MsApriori

The MapReduce framework is applicable to Matrix Apriori technique as per the following proposed scheme shown in figure 3.1. The implementation platform for MapReduce framework is Hadoop which is considered for execution in single server mode. It is important to note that most of the operations of Apriori Algorithm for databases involves counting operation which are most suitable to implement using MapReduce Model for multiprocessing environments. In Matrix Apriori Algorithm, there are only two database scans followed by matrix operations to mine frequent itemsets. This can be readily implemented in Hadoop as

MapReduce implementation. As it is clear that the resultant MFI matrix of real dataset is large, the problem of mining association rules from n-frequent itemsets can be solved using MapReduce Framework.

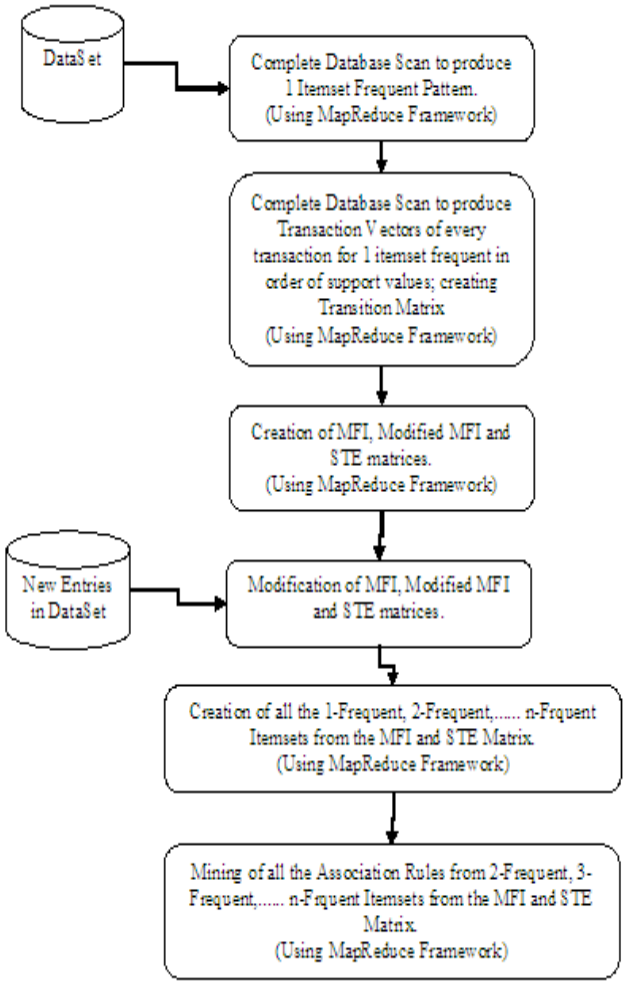

Fig. 3.1: Proposed Framework of Dynamic Matrix MsApriori using MapReduce

1) Setting up of the MFI Matrix

Consider the transaction dataset as specified above with N transactions and I distinct items. Consider a k node cluster with the implementation of MapReduce, the key value pair allotment and computation assignment can be done in the form as shown in table 3.10.

Table - 3.10
Combinations Obtained Based On Conditional Pattern i with Minimum Support

Node	*Operation*
#1	*Transactions: First t Transactions Elements : First set of items*
#2	*Transactions: Next t transactions Elements : First set of items*
#3	*Transactions: Next t transactions Elements : First set of items*
#k	*Transactions: Next t transactions Elements : First set of items*

Consider the operation done by i^{th} node in the first phase of computation. The i^{th} node operates on the set first of items of the unordered item list. Also, the i^{th} node operates on the transaction labeled from (i-1)*t+1 to i*t.

For the first set of items, the support values of these items are computer over subsets of transaction by the corresponding nodes. Let the time required for processing of specified set of items overt transactions is s and the number of rounds needed for entire database to be scanned be r, then the time required for processing of first set of items is s*r.

The reduce operation on the set can be performed as follows:

The support values of the items at the end of each round is added and stored. The beginning of next round takes a fresh start of the item count starting from zero and scans the database transactions which are allotted next. This again followed by subsequent addition to the previous list which finally gives the support count of the subset of items considered at the end of the r^{th} round.

The process is repeated for the next subset of items and followed until all the items in the list are computed over the entire database. If there a p subsets of items, then the time required for scanning of transaction database over the entire set of items is s*r*p.

The items can then be arranged in the ascending order of support values. The representation of the entire database in terms of binary sparse matrix can also be done through assignment over the cluster node in the following way.

The item list in the descending order of support values is stored in a vector v and the set of transactions is represented as binary vectors of length v. The checking of duplicity within the transaction group computed by a cluster node in any round can be done through a flag or key value which stores the count as how many times the vector has been repeated in the set of nodes, and increasing the count when the same vector is encounter again, and then skipping it.

At the end of the first round all binary vectors are stored as separate subsets of vectors. For k such sets, a total of C(k,2) pairs are possible. Taking any two subsets at a time, a cumulative checking of duplicates is performed to remove the duplicates. This can be checked parallel by assignment of pairs to cluster nodes, and removing duplicates, while at the same time, increasing the count in the lower order set. This can only be done in k rounds in which at the first round, set 1 is checked amongst all other set and so on for subsequent rounds.

Finally, at the end of these rounds, the transactions are added to MFI matrix and the count are stored in STE array. In the next round, the same process is repeated. as duplicates are eliminated and then stored in the MFI matrix. Finally, when the MFI matrix is complete, the vector sets representing individual transactions are assigned to cluster nodes in groups for eliminating redundancy as a whole using aforementioned techniques and making a corresponding change in STE.

2) Setting up of the Modified MFI Matrix

The modified MFI matrix can be built up by processing the MFI matrix as per the rules to compute modified MFI. However, the large size of MFI restricts it from being read as a whole. However, one factor that accounts of its storage in efficient way is this MFI is a typical sparse matrix whose most entries are zero. The nonzero entries in the individual columns can be counted and modified MFI can be tabulated.

The non-zero entry just above the occurrence of 1 in any column corresponding to any element gives the support values of that element in the transactions. At this stage, multiple minimum support can be specified by the user and the items no fulfilling the minimum support criteria can be eliminated.

3) Computing Frequent Itemsets

The pairing of the item with least support with the other items to the left of it is made by assigning such pairs to the cluster nodes till the pairing of the leftmost with all other items is being made. For any particular grouping of the item with the other item, the minimum support count of the set is the minimum of the support count of any of the two item. However, the confidence remains a universal measure and has to be specified for the computation of association rules. More strong association rules are computed with higher values of minimum confidence.

Till the computation of two frequent itemsets, the Apriori property is not used. For three of more frequent itemsets, Apriori property is used to mine association rules.

Section 4 analyzes the execution results on real database of WalMart Megastore and gives the implications.

IV. ANALYSIS OF PROPOSED WORK

A. Analysis of Transaction Data

A part of local grocery store database, is shown below in table 4.1. The complete mysql dump in the form of csv (filename: grocery.csv) of the database is included in the CD-ROM enclosed.

Table - 4.1
Subset Of Transactions From The Transaction Data Of Grocery Store

Transaction	*Item-list (Representative)*
T1	*citrus fruit, semi-finished bread, margarine, ready soups*
T2	*tropical fruit, yogurt, coffee*
T3	*whole milk*
T4	*pip fruit, yogurt, cream cheese, meat spreads*
T5	*other vegetables, whole milk, condensed milk, long life bakery product*
T6	*whole milk, butter, yogurt, rice, abrasive cleaner*
T7	*rolls/buns*
T8	*other vegetables, UHT-milk, rolls/buns, bottled beer, liquor (appetizer)*
T9	*pot plants*

T10	*whole milk, cereals*
T11	*tropical fruit, other vegetables, white bread, bottled water, chocolate*
T12	*citrus fruit, tropical fruit, whole milk, butter, curd, yogurt, flour*
T13	*Beef*
T14	*frankfurter, rolls/buns, soda*
T15	*chicken, tropical fruit*
T16	*butter, sugar, fruit/vegetable juice, newspapers*
T17	*fruit/vegetable juice*
T18	*packaged fruit/vegetables*
T19	*Chocolate*
T20	*specialty bar*
T21	*other vegetables*
T22	*butter milk, pastry*
T23	*whole milk*
T24	*tropical fruit, cream cheese , processed cheese, detergent, newspapers*
T25	*tropical fruit, root vegetables, other vegetables, frozen dessert, rolls/buns, flour, sweet spreads*
T26	*bottled water, canned beer*
T27	*Yogurt*
T28	*sausage, rolls/buns, soda, chocolate*
T29	*other vegetables*
T30	*brown bread, soda, fruit/vegetable juice, canned beer, newspapers, shopping bags*

A set of 9,835 records are considered for association rule mining using proposed techniques. The item list and the corresponding support values for the transactions are provided in table 4.2 and in figure 4.1.

Table - 4.2
Item List and the Support Values

whole milk	*2513*	*Grapes*	*220*	*sauces*	*54*
other vegetables	*1903*	*chewing gum*	*207*	*Jam*	*53*
rolls/buns	*1809*	*Detergent*	*189*	*spices*	*51*
Soda	*1715*	*red/blush wine*	*189*	*curd cheese*	*50*
Yogurt	*1372*	*white wine*	*187*	*cleaner*	*50*
bottled water	*1087*	*pickled vegetables*	*176*	*liver loaf*	*50*
root vegetables	*1072*	*semi-finished bread*	*174*	*male cosmetics*	*45*
tropical fruit	*1032*	*baking powder*	*174*	*Rum*	*44*
shopping bags	*969*	*Dishes*	*173*	*meat spreads*	*42*
Sausage	*924*	*Flour*	*171*	*ketchup*	*42*
Pastry	*875*	*pot plants*	*170*	*brandy*	*41*
citrus fruit	*814*	*soft cheese*	*168*	*light bulbs*	*41*
bottled beer	*792*	*processed cheese*	*163*	*Tea*	*38*
Newspapers	*785*	*Herbs*	*160*	*specialty fat*	*36*
canned beer	*764*	*canned fish*	*148*	*abrasive cleaner*	*35*
pip fruit	*744*	*Pasta*	*148*	*skin care*	*35*
fruit/vegetable juice	*711*	*seasonal products*	*140*	*nuts/prunes*	*33*
whipped/sour cream	*705*	*cake bar*	*130*	*artif. sweetener*	*32*
brown bread	*638*	*packaged fruit/vegetables*	*128*	*canned fruit*	*32*
domestic eggs	*624*	*Mustard*	*118*	*syrup*	*32*

Frankfurter	580	frozen fish	115	nut snack	31
margarine	576	cling film/bags	112	snack products	30
Coffee	571	spread cheese	110	Fish	29
Pork	567	Liquor	109	potato products	28
Butter	545	frozen dessert	106	bathroom cleaner	27
Curd	524	Salt	106	cookware	27
Beef	516	canned vegetables	106	soap	26
Napkins	515	dish cleaner	103	cooking chocolate	25
Chocolate	488	flower (seeds)	102	pudding powder	23
frozen vegetables	473	condensed milk	101	tidbits	23
Chicken	422	roll products	101	cocoa drinks	22
white bread	414	pet care	93	organic sausage	22
cream cheese	390	photo/film	91	prosecco	20
Waffles	378	mayonnaise	90	flower soil/fertilizer	19
salty snack	372	sweet spreads	89	ready soups	18
long life bakery product	368	chocolate marshmallow	89	specialty vegetables	17
Dessert	365	Candles	88	organic products	16
Sugar	333	specialty cheese	84	honey	15
UHT-milk	329	dog food	84	decalcifier	15
hamburger meat	327	frozen potato products	83	cream	13
Berries	327	house keeping products	82	frozen fruits	12
hygiene articles	324	Turkey	80	hair spray	11
Onions	305	Instant food products	79	rubbing alcohol	10
specialty chocolate	299	liquor (appetizer)	78	liqueur	9
Candy	294	Rice	75	make up remover	8
misc. beverages	279	instant coffee	73	salad dressing	8
frozen meals	279	Popcorn	71	whisky	8
Oil	276	Zwieback	68	toilet cleaner	7
butter milk	275	Soups	67	baby cosmetics	6
specialty bar	269	finished products	64	frozen chicken	6
Beverages	256	Vinegar	64	bags	4
Ham	256	female sanitary products	60	kitchen utensil	4
Meat	254	kitchen towels	59	preservation products	2
ice cream	246	dental care	57	baby food	1
hard cheese	241	Cereals	56	sound storage medium	1
sliced cheese	241	sparkling wine	55		
cat food	229	Softener	54		

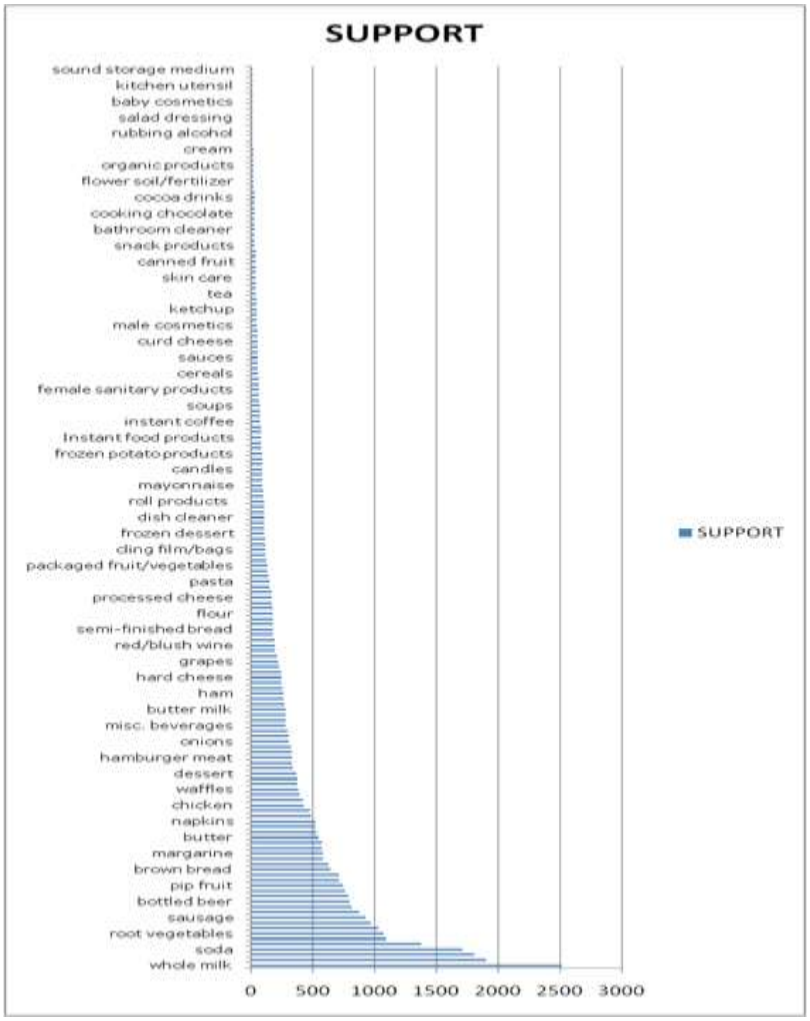

Fig. 4.1: Plot of Support Values of all the items in the transactions listed in the 9835 transaction

The binary transition matrix of transaction database consists of a matrix with 9835 rows and 164 columns. The column values correspond to the maximum number of items in any of the transaction of the database record.

The category information from the products can be obtained by the level information provided in the transactions. For the data considered in this analysis, the level information of a subset of data is depicted in table 4.3

Table - 4.3
Level Data from Item List

	Labels	*level2*	*level1*
1	*frankfurter*	*Sausage*	*meet and sausage*
2	*Sausage*	*sausage*	*meet and sausage*
3	*liver loaf*	*sausage*	*meet and sausage*
4	*Ham*	*sausage*	*meet and sausage*
5	*Meat*	*sausage*	*meet and sausage*
6	*finished products*	*sausage*	*meet and sausage*
7	*organic sausage*	*sausage*	*meet and sausage*
8	*Chicken*	*poultry*	*meet and sausage*
9	*Turkey*	*poultry*	*meet and sausage*
10	*Pork*	*pork*	*meet and sausage*
11	*Beef*	*beef*	*meet and sausage*
12	*hamburger meat*	*beef*	*meet and sausage*
13	*Fish*	*fish*	*meet and sausage*

14	citrus fruit	fruit	fruit and vegetables
15	tropical fruit	fruit	fruit and vegetables
16	pip fruit	fruit	fruit and vegetables
17	Grapes	fruit	fruit and vegetables
18	Berries	fruit	fruit and vegetables
19	nuts/prunes	fruit	fruit and vegetables
20	root vegetables	vegetables	fruit and vegetables
21	Onions	vegetables	fruit and vegetables
22	Herbs	vegetables	fruit and vegetables
23	other vegetables	vegetables	fruit and vegetables
24	packaged fruit/vegetables	packaged fruit/vegetables	fruit and vegetables
25	whole milk	dairy produce	fresh products
26	Butter	dairy produce	fresh products
27	Curd	dairy produce	fresh products
28	Dessert	dairy produce	fresh products
29	butter milk	dairy produce	fresh products
30	Yogurt	dairy produce	fresh products
31	whipped/sour cream	dairy produce	fresh products
32	Beverages	dairy produce	fresh products
33	UHT-milk	shelf-stable dairy	fresh products
34	condensed milk	shelf-stable dairy	fresh products
35	Cream	shelf-stable dairy	fresh products

Table 4.4 lists all the level-2 which exists in the entire grocery store transaction dataset.

Table - 4.4
Levels (Level-2) In the Database Groceries

1	baby food	29	hard drinks
2	Bags	30	health food
3	bakery improver	31	jam/sweet spreads
4	bathroom cleaner	32	long-life bakery products
5	Beef	33	meat spreads
6	Beer	34	non-alc. drinks
7	bread and backed goods	35	non-food house keeping products
8	Candy	36	non-food kitchen
9	canned fish	37	packaged fruit/vegetables
10	canned fruit/vegetables	38	perfumery
11	Cheese	39	personal hygiene
12	chewing gum	40	pet food/care
13	Chocolate	41	pork
14	Cleaner	42	poultry
15	Coffee	43	pudding powder
16	Condiments	44	sausage
17	Cosmetics	45	seasonal products
18	dairy produce	46	shelf-stable dairy
19	Delicatessen	47	snacks
20	dental care	48	soap

21	*detergent/softener*	*49*	*soups/sauces*
22	*Eggs*	*50*	*staple foods*
23	*Fish*	*51*	*sweetener*
24	*frozen foods*	*52*	*tea/cocoa drinks*
25	*Fruit*	*53*	*vegetables*
26	*games/books/hobby*	*54*	*vinegar/oils*
27	*Garden*	*55*	*wine*
28	*hair care*		

Table 4.5 lists all the level-2 which exists in the entire grocery store transaction dataset.

Table - 4.5
Levels (Level-1) In the Database Groceries

S. No.	*Level 1 / Category*
1	*canned food*
2	*Detergent*
3	*Drinks*
4	*fresh products*
5	*fruit and vegetables*
6	*meet and sausage*
7	*non-food*
8	*Perfumery*
9	*processed food*
10	*snacks and candies*

In total, there are 9835 transaction and 169 different items. Five items, viz; bags, kitchen utensils, preservation products, baby food and sound storage medium are excluded from further analysis as these have support values 4,4,2,1 and 1 respectively which is below the threshold considered, which is 6. The items in each of the transaction are sorted in the order of descending support values. Thereafter, the transaction matrix can be prepared with rows indicating transactions and columns indicating items in the order or ascending support count. The second level grouping of items gives 55 groups and the first level grouping gives 10 groups/categories as shown in table 4.4 and 4.5.

The visualization of the sparce transition matrix of the complete grocery store transaction data (9835 rows and 164 columns) is shown in figure 4.2.

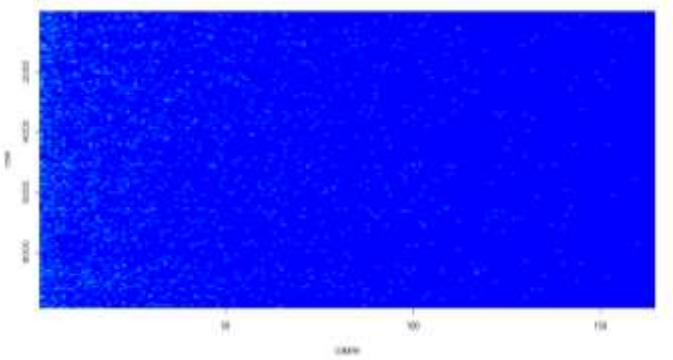

Fig. 4.2: Visualization of sparse transaction matrix of complete grocery transaction data.

The number of association rules generated depends upon the values of support and confidence of the rules. The number of rules generated keeping all items separated, by grouping in level 2 groups and by grouping in level 1 group is shown in table 4.6.

Table - 4.6
Number Of Rules Generated As Function Of Support Value (Confidence = 0.5)

SUPPORT	*R*	*R2*	*R1*
0.001	*5668*	*1160*	*144*
0.002	*1098*	*304*	*54*
0.003	*421*	*142*	*32*

0.004	*197*	*75*	*21*
0.005	*120*	*49*	*18*
0.006	*67*	*33*	*13*
0.007	*45*	*20*	*8*
0.008	*30*	*13*	*6*
0.009	*25*	*13*	*6*
0.01	*15*	*9*	*4*

The visualization of table 4.6 is shown in figure 4.3.

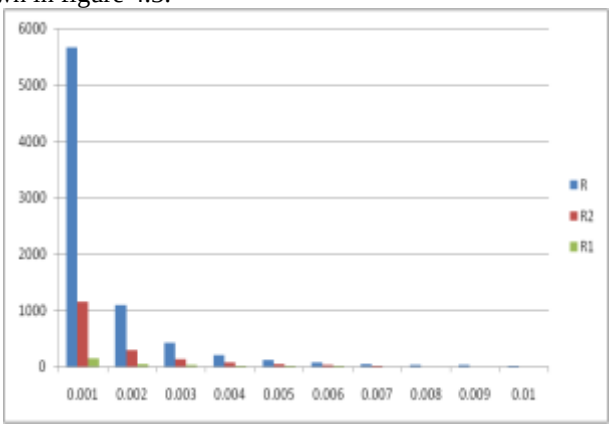

Fig 4.3: Number of Rules Generated (R, R2 and R1) as a function of Support (keeping confidence = 0.5)

Table 4.7 lists the number of rules as a function of confidence by keeping the support value fixed at 0.001. The visualization of table 4.7 is shown in figure 4.4.

Table - 4.7
Number of Rules Generated As Function of Confidence Value (Support = 0.001)

CONFIDENCE	*R*	*R2*	*R1*
0.1	*32791*	*5053*	*390*
0.2	*21634*	*3481*	*318*
0.3	*13770*	*2383*	*243*
0.4	*8955*	*1633*	*176*
0.5	*5668*	*1160*	*144*
0.6	*2918*	*742*	*105*
0.7	*1279*	*425*	*75*
0.8	*410*	*191*	*50*
0.9	*129*	*78*	*27*
1	*28*	*24*	*13*

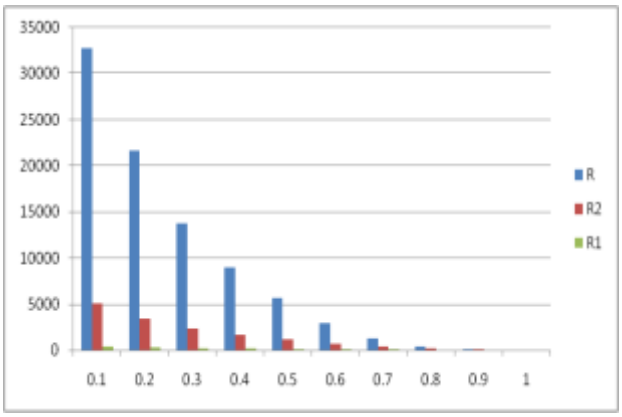

Fig. 4.4: Number of Rules Generated (R, R2 and R1) as a function of Confidence (keeping support = 0.001)

The scatter plot of all the 5663 rules obtained by keeping minimum support = 0.001 , and minimum confidence = 0.5 , are shown in the figure 4.3.

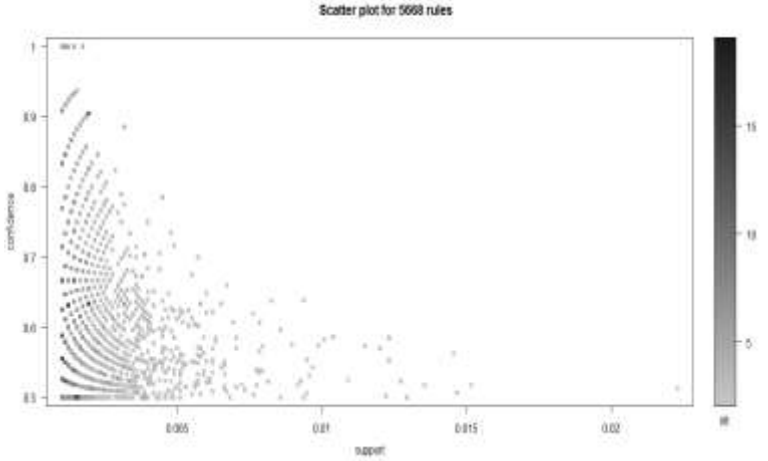

Fig. 4.5: Scatter Plot for all rules

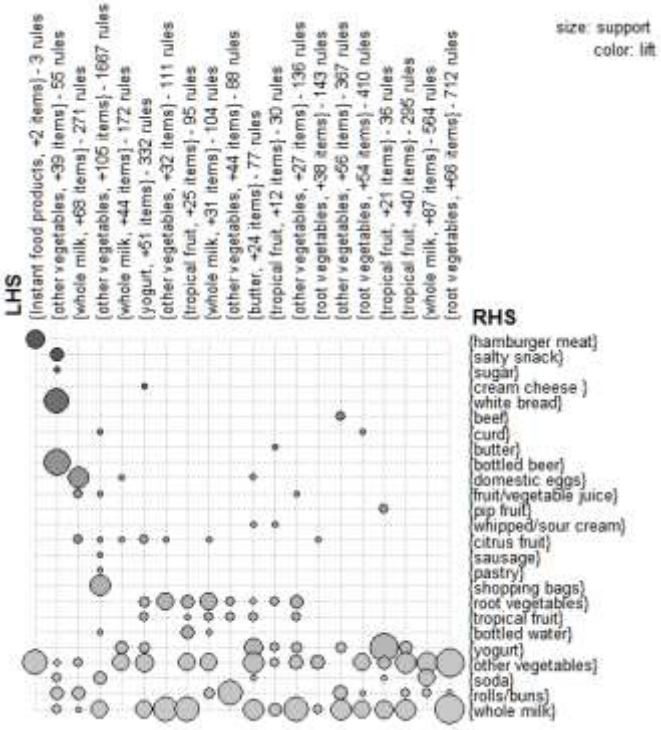

Fig. 4.6: Grouped Plot for all the Association Rules

B. Using MapReduce Framework in Association Rule Mining in Dynamic Matrix MsApriori

1) Setting up of the MFI Matrix

Consider the transaction dataset as specified above with N = 9835 transactions and I = 169 distinct items. The first step is to compute the support values of individual items and to sort them in the order of support values. Assuming that the list of elements is unordered and considering the first 20 items of the list, the support count of individual items can be done in the way as depicted in table 4.8.

Consider a k=10 node cluster for implementing parallel processing environment. The computation workflow for MFI for setting up item support vector is as shown:

Table - 4.8
Computation of Support Values of Different Items in DB

Node /Time	*40,000*	*200*			
Node 1	*Transactions*	*1-1000*	*<key1, value1>*		
	Items	*1-20*			

Node 2	*Transactions*	*1001-2000*	*<key2, value2>*		
	Items	*1-20*			
Node 3	*Transactions*	*2001-3000*	*<key3, value3>*		
	Items	*1-20*			
Node 4	*Transactions*	*3001-4000*	*<key4, value4>*		
	Items	*1-20*			
Node 5	*Transactions*	*4001-5000*	*<key5, value5>*	*REDUCE*	*<key, value>*
	Items	*1-20*			
Node 6	*Transactions*	*5001-6000*	*<key6, value6>*		
	Items	*1-20*			
Node 7	*Transactions*	*6001-7000*	*<key7, value7>*		
	Items	*1-20*			
Node 8	*Transactions*	*7001-8000*	*<key8, value8>*		
	Items	*1-20*			
Node 9	*Transactions*	*8001-9000*	*<key9, value9>*		
	Items	*1-20*			
Node 10	*Transactions*	*9001-9835*	*<key10, value10>*		
	Items	*1-20*			

The time description at the top of the table 4.8 can be understood by table 4.9

Table - 4.9
Time Complexity Using Map Reduce

Operation Descriptions		
		Time
Node i	*One item scan through DB and adding supports: 1000+1000*	*20*2000 = 40,000 time units*
Additions	*10*20*	*200*
Loops	*9*	*9*(40000+200)*

It is assumed that it takes a unit operation time in scanning a specific item in a transaction (on an average). Also, it takes unit operation time in adding two numbers. Thus, to scan 20 items over a database of 1000 records, it takes 20000 units operation time. Also, for summing these values consecutively after each scan, an additional 1000*20 operation time is need. For combining these values as a single vector, 10*20 unit operational time is needed. Thus a total of 40200 unit operation time is needed for phase 1. At the end of this phase, one has the support count values of the first 20 items of the unordered item list, in which the key values are the list of items and the value holds the corresponding support values. A total of 9 rounds are needed, thus giving a total time of 40010*9 = 361800. After this support value computation, the sorting operation on the itemset consisting of about 169 items which requires time of the order of O(n*logn) which equals to about 377. Thus, a total of about 362177 time units.

On the other hand, if the same operation is performed over a uniprocessor system under same settings, the time required would be 9835*169*2 = 3324230. This is because each item is checked in all the transactions, and for each transaction, the support count, (either 1 or 0) is added to the computed number till that scan. The improvement in the time complexity in computing the support values of distinct items and sorting them in the order of support values is depicted in figure 4.7.

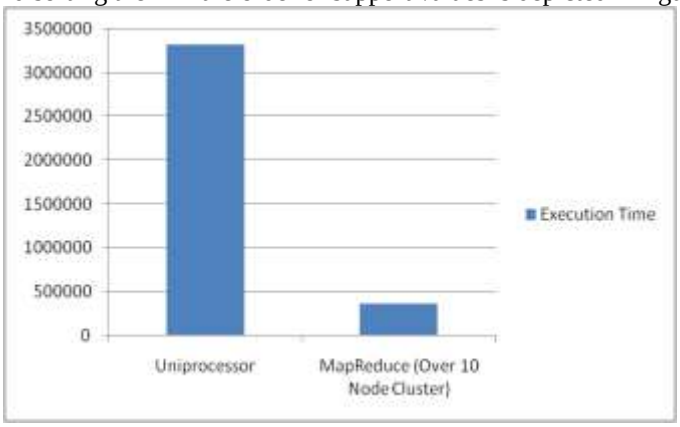

Fig. 4.7: Comparison of Execution time over Uni-Processor and Cluster Environment. (Computation of Item Support Values and Item Sorting thereafter)

It is evident that MapReduce framework provides an improvement of 94 percent as compared to the standalone solution.

At the end of this iteration, the support values of all the elements are computed and the elements are sorted in descending order of the support values. The time complexity of setting up of MFI matrix can be measured as shown in table 4.10.

Table - 4.10
Setting Up Of MFI

Node /Time			*Binary Vectors*		
Node 1	*Transactions*	*1-1000*	*<key1, value1>*		
	Items	*All*			
Node 2	*Transactions*	*1001-2000*	*<key2, value2>*		
	Items	*All*			
Node 3	*Transactions*	*2001-3000*	*<key3, value3>*		
	Items	*All*			
Node 4	*Transactions*	*3001-4000*	*<key4, value4>*		
	Items	*All*			
Node 5	*Transactions*	*4001-5000*	*<key5, value5>*	*REDUCE*	*<key, value>*
	Items	*All*			
Node 6	*Transactions*	*5001-6000*	*<key6, value6>*		
	Items	*All*			
Node 7	*Transactions*	*6001-7000*	*<key7, value7>*		
	Items	*All*			
Node 8	*Transactions*	*7001-8000*	*<key8, value8>*		
	Items	*All*			
Node 9	*Transactions*	*8001-9000*	*<key9, value9>*		
	Items	*All*			
Node 10	*Transactions*	*9001-9835*	*<key10, value10>*		
	Items	*All*			

Each node processes entire sorted list of items, in the order of support values, and represent its 1000 transactions in terms of binary vector of length 169 (item count), in which a 1 indicates the presence of the item and a zero indicates the absence of the item. For each transaction, the binary representation is done in 169*n time units where n is the maximum number of items in any transactions. Considering the average value of items in any transactions to be 10, the time unit operations in one vector representation is 1690. Thus, for a total of 1000 transactions, the time required is 1690*1000 = 1690000. Also, each of the vector is compared if it has already been existed or not, which takes 1000(1000+1)/2 comparisons for the entire set of 1000 records. Each of this comparison has a time complexity of 169 as there are 169 elements in vector matching. Thus the computation of comparison needs a time of 84584500 time units. These records are tabulated in STE matrix requiring 1000 time units. This gives a total computation time of 86275500.

After this matching, the individual blocks are matched again to remove the redundancy among the records. There are $^{10}C_2 = 45$ pairs of blocks, each requiring 10000 matching operations, thus requiring 450000 matching of vectors to remove redundancy. Each such comparison matching requires 169 time units operation, thus giving a total of 76050000 time units operation. The removal of the corresponding rows from STE and the increase in the count of redundant rows can be ignored. Thus, MapReduce Architecture requires a total of 162325500 time unit operations to create MFI.

The time required by a uni-processor system for the same can be computed as follows:

Table - 4.11
Time Complexity Analysis of Uniprocessor System Architecture

Operation	*Time Complexity*
One Vector Representation	*1690 (Assuming 10 items average per transaction)*
9835 Transactions	*16621150*
Comparison Operation	*8450845000*
STE Updating	*10000*
Total	*8467477840*

The comparison of the time complexity of Uniprocessor and MapReduce Architecture is shown in figure 4.8.

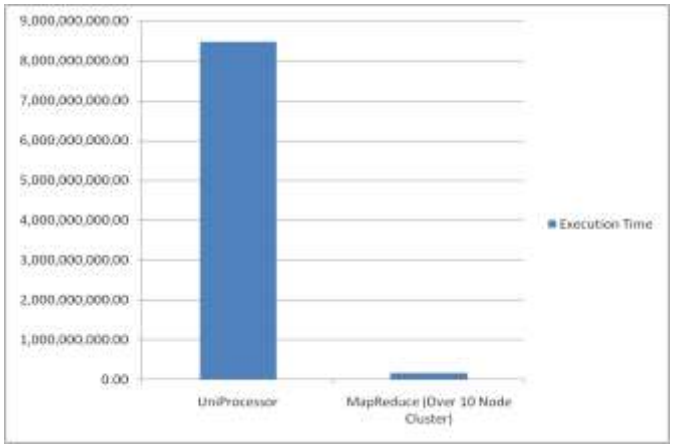

Fig. 4.8: Comparison of Execution time over Uni-Processor and Cluster Environment. (Computation of Matrix of Frequent Items, MFI)

Table 4.11 suggests a relative improvement of 94 percent using 10 node cluster framework in the computation of MFI. The mining of association rules can be made in a straightforward way after the creation of MFI using the Apriori Property. Also, all the elements are considered for frequent itemset mining in MsApriori. The combined support of a set of items is the minimum support of any of the item belonging to the set and using this approach, the confidence values of rules are checked for association rule mining. New items can be added in the matrix in batch mode ion which the batch size can be fixed depending upon the relative accuracy desired in the results and without causing unnecessary reprocessing in the computation of support values and MFI matrix. Section 5 analyses the results and concludes the paper.

V. CONCLUSION AND FUTURE SCOPE

Association rule mining aims to discover interesting patterns in a database. There are two steps in this data mining technique. The first step is finding all frequent itemsets and the second step is generating association rules from these frequent itemsets. Association rule mining algorithms generally focus on the first step since the second one is direct to implement. Although there are a variety of frequent itemset mining algorithms, each one requires some pre-specified value of the minimum support count. This results in inefficient mining of information as some of the infrequent rules get skipped away although these are of particular interest. Moreover, classical Apriori algorithm is inefficient in the sense that complete database scan has to be performed for generating k-frequent itemsets from K-1 frequent itemsets.

The contribution of this paper is many fold. It first proposes Matrix Apriori for Multiple Minimum Support Values of the items. The existing algorithm for implementing multiple minimum support is called MsApriori. Thus, a technique of implementing MsApriori using Matrices, called Matrix MsApriori is presented. Moreover, a technique for solution of the problem of mining association rules is presented using MapReduce Framework is presented. The MapReduce framework employs clustering techniques implementing multiprocessing to solve large computational problems using divide and conquer technique. The comparison of the analytical results over uni-processor and cluster is presented and the results are tabulated which shows an improvement of about 94 percent using 10 node cluster over a transaction database consisting of 9835 transactions.

As future scope, the same algorithm is to be implemented utilizing the domain specific heuristic knowledge to filter out uninteresting rules from the interesting ones, to provide a complete solution as a whole, which relates to the issues of efficiency, infrequent rule mining and dynamic database problem. Moreover, machine learning can also be implemented using the categorical description of the item sets to give an insight into which of the rules may be investigated for being potentially profitable. This can be done using the training sets based on the already discovered profitable association rules and thereby, checking for the profitability of the new ones.

REFERENCES

[1] Agrawal, R., T. Imielinski, and A. Swami (1993). Mining association rules between sets of items in large databases. In Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data, SIGMOD '93, New York, NY, USA, pp. 207–216. ACM.
[2] Agrawal, R. and R. Srikant (1994). Fast algorithms for mining association rules in large databases. In Proceedings of the 20th International Conference on Very Large Data Bases, VLDB '94, San Francisco, CA, USA, pp. 487–499. Morgan Kaufmann Publishers Inc.
[3] R.Agrawal, H.Mannila, R.Srikant, H.Toivonen, and A.I.Verkamo, Fast Discovery of Association Rules. In U. Fayyad et al. (eds), Advances in Knowledge Discovery and Data Mining (Menlo Park, CA: AAAI Press, 1996, 307-328).
[4] Hong-Zhen Zheng, Dian-Hui Chu, De-Chen Zhan : Association Rule Algorithm Based on Bitmap and Granular Computing. AIML Journal, Volume (5), Issue (3), September, 2005, pp. 51-54.
[5] Han, J. and M. Kamber (2006). Data Mining. Concepts and Techniques (2nd ed. ed.). Morgan Kaufmann.
[6] R. Agrawal, Heikki Mannila Fast Discovery for Mining Association Rules, International journal of computer applications, 2000, pp. 86-91.
[7] Pav´on, J., S. Viana, and S. G´omez (2006). Matrix apriori: Speeding up the search for frequent patterns. In Proceedings of the 24th IASTED International Conference on Database and Applications, DBA'06, Anaheim, CA, USA, pp. 75–82. ACTA Press.
[8] J. Dean and S. Ghemawat. MapReduce: Simplified Data Processing on Large Clusters. In OSDI'04, 6th Symposium on Operating Systems Design and Implementation, Sponsored by USENIX, in cooperation with ACM SIGOPS, pages 137–150, 2004.

[9] J.Han, J.Pei, Y.Yin, Mining Frequent Patterns without Candidate Generation. In: Proceedings of ACM-SIGMOD International Conference on Management of Data. Vol 29, No. 2, 2012, 1-12.
[10] M. Dimitrijevic, and Z. Bosnjak "Discovering interesting association rules in the web log usage data". Interdisciplinary Journal of Information, Knowledge, and Management, 5, 2010, pp.191-207.
[11] Rameshkumar, K.; Sambath, M.; Ravi, S., "Relevant association rule mining from medical dataset using new irrelevant rule elimination technique," in Information Communication and Embedded Systems (ICICES), 2013 International Conference on , vol., no., pp.300-304, 21-22 Feb. 2013 doi: 10.1109/ICICES.2013.6508351.
[12] K. Yun Sing "Mining Non-coincidental Rules without a User Defined Support Threshold". 2009.
[13] Sourav Mukherji, A framework for managing customer knowledge in retail industry, IIMB Management Review, Volume 24, Issue 2, June 2012, Pages 95-103, ISSN 0970-3896, http://dx.doi.org/10.1016/j.iimb.2012.02.003.
[14] Jiao Yabing, "Research of an Improved Apriori Algorithm in Data Mining Association Rules," International Journal of Computer and Communication Engineering vol. 2, no. 1, pp. 25-27 , 2013.
[15] C. Wang, R. Li, and M. Fan, "Mining Positively Correlated Frequent Itemsets," Computer Applications, vol. 27, pp. 108-109, 2007.
[16] Kimmo Hatonen, Mika Klemettinen, Heikki Mannila, Pirjo Ronkainen, and Hannu Toivonen. Knowledge discovery from telecommunication network alarm databases. In Stanley Y. W. Su, editor, Proceedings of the 12th International Conference on Data Engineering (ICDE'96), pages 115 – 122, New Orleans, Louisiana, USA, February 1996. IEEE Computer Society Press.
[17] Mika Klemettinen. Rule Discovery from Telecommunication Network Alarm Databases. PhD thesis, Department of Computer Science, P.O. Box 26, FIN-00014 University of Helsinki, Finland, January 1999.
[18] Rakesh Agrawal, Tomasz Imielinski, and Arun N. Swami. Mining association rules between sets of items in large databases. In Peter Buneman and Sushil Jajodia, editors, Proceedings of ACM SIGMOD International Conference on Management of Data (SIGMOD'93), pages 207 – 216, Washington, D.C., USA, May 1993. ACM.
[19] Rakesh Agrawal, Heikki Mannila, Ramakrishnan Srikant, Hannu Toivonen, and A. Inkeri Verkamo. Fast discovery of association rules. In Usama M. Fayyad, Gregory Piatetsky-Shapiro, Padhraic Smyth, and Ramasamy Uthurusamy, editors, Advances in Knowledge Discovery and Data Mining, pages 307 – 328. AAAI Press, Menlo Park, California, USA, 1996.
[20] Awadalla, Medhat H. A.; El-Far, Sara G., " Aggregate Function Based Enhanced Apriori Algorithm for Mining Association Rules", International Journal of Computer Science Issues (IJCSI);May2012, Vol. 9 Issue 3, p277, May 2012.
[21] Dehay, D.; Leskow, J.; Napolitano, A., "Central Limit Theorem in the Functional Approach," in Signal Processing, IEEE Transactions on , vol.61, no.16, pp.4025-4037, Aug.15, 2013 doi: 10.1109/TSP.2013.2266324
[22] Vipul Mangla, Chandni Sarda, SarthakMadra), "Improving the efficiency of Apriori Algorithm in Data Mining", International Journel of Engineering and Innovative technology, Volume 3, Issue 3 September 2013.
[23] Sumithra, R.; Paul, S., "Using distributed apriori association rule and classical apriori mining algorithms for grid based knowledge discovery," in Computing Communication and Networking Technologies (ICCCNT), 2010 International Conference on , vol., no., pp.1-5, 29-31 July 2010 doi: 10.1109/ICCCNT.2010.5591577.
[24] Yew-Kwong Woon; Wee-Keong Ng; Das, A., "Fast online dynamic association rule mining," in Web Information Systems Engineering, 2001. Proceedings of the Second International Conference on , vol.1, no., pp.278-287 vol.1, 3-6 Dec. 2001 doi: 10.1109/WISE.2001.996489.