I.J. Mathematical Sciences and Computing, **2017, 4, 52-66**
Published Online November 2017 in MECS (http://www.mecs-press.net)
DOI: 10.5815/ijmsc.2017.04.05

Available online at ***http://www.mecs-press.net/ijmsc***

Augmented Apriori by Simulating Map-Reduce

R.Akila[a], Dr.K.Mani[b]

[a]*Assistant Professor, Nehru Memorial College, Puthanampatti, Tiruchirappalli, Tamilnadu, India- 621 007.*
[b]*AssociateProfessor, Nehru Memorial College, Puthanampatti, Tiruchirappalli, Tamilnadu, India- 621 007.*

Received: 13 July 2016; Accepted: 08 June 2017; Published: 08 November 2017

Abstract

Association rule mining is a data mining technique which is used to identify decision-making patterns by analyzing datasets. Many association rule mining techniques exist to find various relationships among itemsets. The techniques proposed in the literature are processed using non-distributed platform in which the entire dataset is sustained till all transactions are processed and the transactions are scanned sequentially. They require more space and are time consuming techniques when large amounts of data are considered. An efficient technique is needed to find association rules from big data set to minimize the space as well as time. Thus, this paper aims to enhance the efficiency of association rule mining of big transaction database both in terms of memory and speed by processing the big transaction database as distributed file system in Map-Reduce framework. The proposed method organizes the transactions into clusters and the clusters are distributed among many parallel processors in a distributed platform. This distribution makes the clusters to be processed simultaneously to find itemsets which enhances the performance both in memory and speed. Then, frequent itemsets are discovered using minimum support threshold. Associations are generated from frequent itemsets and finally interesting rules are found using minimum confidence threshold. The efficiency of the proposed method is enhanced in a noticeably higher level both in terms of memory and speed.

Index Terms: Map-Reduce, Distributed File System, Association Rule Mining, Cluster, Apriori, Minimum Support, Minimum Confidence.

1. Introduction

Data Mining discovers interesting patterns from large amounts of data where the data can be stored in databases, data warehouses and data repositories [1]. These patterns can be used to predict future happenings

* Corresponding author. Tel.:
E-mail address: grr.akila@gmail.com

and to make important decisions. Association rule mining is one of the data mining tasks to discover decision-making patterns. The decision-making patterns are generated in the form of if-then rules which are human understandable. The associations can be used for prediction [16]. Though Apriori is one most popular and easiest association rule mining technique, its efficiency is degraded due to many reasons such as performing too many scans on the database, requiring too much processing time, memory and generating too many candidate sets. These issues are exaggerated much more when big data is dealt.

Data repositories are long-term storage of big data and data are collected from multiple sources which is organized so as to facilitate management decision-making. The data are stored under a unified schema and are summarized. Data repository systems provide data analysis capabilities, collectively referred to as On Line Analytical Processing (OLAP). OLAP operations include drill-down, roll-up, and pivot [2]. Data repositories may be big dataset. Big data is huge in its volume which is structured. Most of the structured data in scientific domain are voluminous. Processing of such big data requires state of the art computing machines. Setting up such an infrastructure is expensive.. A distributed platform is employed for tackling such scenarios using Map-Reduce [15]. Distributed platform employs distribution among multiprocessor systems wherein many tasks can be processed simultaneously. It requires tasks to be partitioned and to be distributed among the processors. The interaction among multiple processors can take place using message passing. It provides responses at great speed and utilizes the system efficiently. Similar kind of characteristics can also be accomplished in a single processor system using multithreading. Only one difference is that partitioned tasks are designed as threads, instead of processes. Map-Reduce framework is to handle big data in a distributed platform. It needs clustering of documents. Document clustering is an important part of mining [17].

Map-Reduce are used in Artificial Intelligence as functional programming [6]. It has received the highlight since it is reintroduced by Google to solve the problems by analyzing Big Data. It is defined as multiple bytes of data in distributed computing. It is inspired by Google's Map-Reduce and Google File Systems (GFS) [2]. This approach is especially for big data set and the same can be applied for association rule mining. Apriori algorithm is a powerful and important algorithm of association rule mining to mine frequent itemsets for generating Boolean association rules [10]. It is expensive because of frequent scans on the database [12], computational complexity and costly comparisons for the generation of candidate itemsets [3]. Map-Reduce programming model can be used [11-13] for the implementation of scalable Apriori algorithm and it is a parallel data processing system. The input and output data have key-value pairs in a specific format. The users express an algorithm using two functions which are map and reduce functions [6-9]. Map function generates a set of intermediate key-value pairs. Reduce function combines all intermediate values associated with the same key [4] [14] [15]. Cloud computing and Grid computing are distributed environment which may be used for parallelism [5] [8] [9]. Apache Hadoop distribution is one of the cluster frameworks in distributed platform that helps to distribute large amounts of data across a number of nodes in the framework [7] [14].

It is observed that Map-Reduce programming model is used to analyze large amounts of data with the consumption of less memory, less processing time and the efficiencies of existing association rule mining techniques lag behind while voluminous data are analyzed. In addition, 1, 2, 3,…,n-candidate itemsets are generated for the generation of frequent itemsets and all the transactions are scanned for the total number of itemsets of 1, 2, ,3,…,n candidate itemsets. They lead to too many scans, need of more memory space and need more consumption of time. So it is worthwhile to incorporate Map-Reduce with the association rule mining to enhance the efficiency of association rule mining. In the proposed work, the transactional database (TDB) is distributed among multiple processors to deal with big data. The TDB is not distributed entirely. It is partitioned into clusters and the number of clusters is determined by the number of processors. These clusters consisting of the transactions are distributed among parallel processors. After the clusters are assigned to the processors, they are stored and processed by parallel processors. Parallel processes are begun with map process. It scans each transaction to find only single itemsets with their occurrences. It is further continued by finding combinations for 2, 3, …, n-itemsets with their occurrences from single itemsets. Then, reduce process performs its task by summing up the number of occurrences of the generated itemsets within each processor and the itemsets generated by all clusters are accumulated together to construct itemsets for the database. One

more reduce process follows to sum the occurrences of the accumulated itemsets. Map and Reduce produce key-value pairs where the itemsets are keys and their occurrences are values. Then, frequent itemsets are generated using minimum support and appropriate interesting rules are generated using the minimum confidence.

The rest of the paper is organized as follows. The related work based on Map-Reduce and Apriori algorithm is presented in section 2. Section 3 focuses on the proposed method. An example for proposed method is described in section 4. Section 5 discusses the results. Finally, section 6 ends with conclusion.

2. Related Work

Apriori discovers interesting relationships among various attributes of a dataset using prior knowledge. It has gained its popularity by its efficiency. However, it lacks in handling big dataset by requiring more space and also spending more time to generate candidate and frequent itemsets. Map-Reduce framework resolves many issues related with big data. This paper integrates Map-Reduce in a distributed platform with association rule mining to enhance the efficiency of association rule mining by minimizing the memory requirement and by minimizing the time spent for the generation of candidate itemsets. Thus, this section describes various work related to Map-Reduce and Association Rule Mining.

Jiawei Han and Michelin Kamber [1] have described many association rule mining techniques such as Apriori, FP-growth, closed frequent itemset mining and mining frequent itemset using vertical data format. In [2], Jongwook Woo stated that Map/Reduce algorithm has received highlights as cloud computing services with Hadoop frameworks and many approaches are there to convert many sequential algorithms to the corresponding Map-Reduce algorithms. They presented Map-Reduce algorithm of the legacy Apriori algorithm that has been popular to collect the itemsets frequently occurred to compose Association Rule in Data Mining and further stated that theoretically, their proposed algorithm provides high performance computation depending on the number of Map and Reduce nodes. Sanjay Rathee et al. [3] proposed in-memory distributed dataflow platform called Spark which overcomes the disk I/O bottlenecks in Map-Reduce. Spark presents a platform for distributed Apriori which dramatically reduces this computational complexity by eliminating the candidate generation and avoiding costly comparisons.

Jeffrey Dean and Sanjay Ghemawat [4] executed Map-Reduce on large clusters of computers and many Map-Reduce programs have been implemented and more than one thousand Map-Reduce jobs are executed on Google's clusters every day. In [5], R. Sumithra et al. mined distributed data in a cloud environment using a hybrid apriori association rule algorithm which combines the benefits of both hash-t and weighted apriori algorithms and they compared the performance with the existing implementation of weighted and hash-t algorithms. Shafali Agarwal [6] emphasized about improving the performance of various applications using recent Map-Reduce models and discussed the usefulness of processing large scale dataset. Also they performed a comparative study of given models those correspond to Apache Hadoop and Phoenix based on execution time and fault tolerance.

In [7], Sonali Satija and Dr. Rajender Nath stated that association rules are if/then statements that help to discover relationships among unrelated data. In order to find the frequent itemsets, there is a need to scan the database many times. The main limitation of Apriori algorithm wastes more time to hold a vast number of candidate sets. In addition, they stated that single processor memory and CPU resources are very limited, which make the algorithm performance inefficient. Because of growth of information, enterprises deal with growing amount of data. So, the solution to this problem is parallel and distributed computing. They said that this can be achieved by Hadoop Map-Reduce model. They have implemented an efficient Map-Reduce Apriori algorithm based on Hadoop Map-Reduce model. In paper [8],Vijay Swaroop discussed the usage of Frequent Itemset Mining (FIM) in cloud computing to analyze data and aimed at extracting frequent itemsets using generation of candidates with many nodes on Map-Reduce programming model and its development platform-Hadoop. Zhang Danping et al. [9] optimized apriori algorithm according to the characteristic achieved by Map-Reduce model running parallel. MR-Apriori algorithm which is improved by parallelism reduces time consumption

significantly and its strong extending ability is suitable for large data analysis, processing and mining. The high extension ability of MR-Apriori algorithm has been realized based on Hadoop platform in cloud computing environments.

In [10], Sudhakar Singh et al. presented an overview of parallel Apriori algorithm implemented on Map-Reduce framework. They categorized on the basis of Map and Reduce functions used to implement them like 1-phase vs k-phase, I/O of Mapper, Combiner and Reducer, using functionality of Combiner inside Mapper etc. This survey discussed and analyzed the various implementations of Apriori on Map-Reduce framework by the basis of their distinguishing characteristics. Moreover, it also includes the advantages and limitations of Map-Reduce framework. Y. Venkata Raghavarao [11] et al. stated that Distributed Big Data Mining (DBDM) is an architecture to rectify the mismatch between the available centralized old data mining systems and distributed systems. One feature of DBDM is its plug-in concept to carry out different data mining algorithms on big data by the help of Map-Reduce functionality and also used to find frequent stream data patterns using the improved apriori algorithm. In [12], Uday K. Kakkad and Prof. Rajanikanth Aluvalu provided a survey to implement Apriori algorithm for huge raw data and also overcome Hadoop limitation using enhanced scheduling algorithm. They further described that implementation of data mining techniques in cloud will allow users to retrieve meaningful information from virtually integrated data repository that reduces the costs of resources. In [13], A.Pradeepa and Dr.Antony selvadoss Thanamani stated that an integrating classification and association rule mining produces more efficient and accurate classifiers than traditional techniques. They developed an associative rule mining that inherits the Map-Reduce scalability to huge datasets and to thousands of processing nodes easily and quickly. It comprehensively evaluates an accurate and effective classification technique, highly competitive and scalable if compared with other traditional associative classification approaches.

In [14], A. Ezhilvathani and Dr. K. Raja extracted frequent patterns among itemsets in the transaction databases and other repositories and mentioned that Apriori algorithms have a great influence for finding frequent itemsets using candidate generation. Apache Hadoop software framework is based on Map-Reduce programming model to improve the processing of large scale data on high performance cluster to process vast amount of data in parallel on large clusters of computer nodes resulting in reliable, scalable and distributed computing. In [15], Anjan K Koundinya1 et al. described that most of the structured data in scientific domain are voluminous. Processing those data requires state of the art computing machines. Setting up such an infrastructure is expensive. A distributed environment is employed for tackling such scenarios. Apache Hadoop distribution is one of the clusters frameworks in distributed environment that helps by distributing voluminous data across a number of nodes in the framework and focused on Map-Reduce design and implementation of Apriori algorithm for structured data analysis. G.Vamsi Krishna [16] proposed a method based on Gaussian Mixture model together with K- means clustering to predict rainfall with many associated factors. Deepa B. Patila and Yashwant V. Dongre [17] performed a comparative study on fuzzy c- means and k-means clustering and stated that fuzzy clustering are more appropriate for document clustering.

3. Proposed Work

The proposed Augmented Apriori by Simulating Map-Reduce method simulates Map-Reduce of big data for mining association rules. It is a method of analyzing large amounts of data simultaneously. The properties of simultaneous process are directed towards quick response and the efficient utilization of memory and processor. It is basically distributed file system in which the document/file is divided into clusters. The number of clusters to be constructed depends on the number of parallel processors which are working parallel and the contents of the clusters are transactions which are processed sequentially. The number of transactions to be allotted to each cluster is determined by the total number of transactions and the total number of parallel processors. The transactions to be assigned to each cluster are determined by the order in which they present in the database. As existing clustering techniques of using distance or link measures cannot be used for clustering transactions, transactions are assigned to clusters based on their existence order. Since parallelism takes place on clusters, even voluminous data can be processed with minimum time which leads to quick response. Map process finds

pairs of keys and their values. Itemsets of 1, 2, 3,…, n-itemsets are the keys and the number of occurrences of them are their values. Reduce process accumulates the occurrences of the itemsets. It is accomplished by examining single or multiple occurrences of the itemset among the transactions. If there is a single occurrence of the key, then the value of the corresponding key that is found by map is retained. Otherwise, the values those are found by map are added together.

When parallelism begins on each cluster within each processor as soon as the cluster is received, initially single items are extracted from each transaction. Map is invoked to find the occurrences of single items before 2, 3, 4,…, n-itemsets are constructed and Reduce process is then performed to compute the sum of the occurrences of the single itemsets. It is followed by the construction of combinations (2, 3, …, n-itemsets) of itemsets from the extracted single itemsets and map is again invoked to map the occurrences of the combinations of 2, 3, …, n-itemsets. Reduce process is then performed to compute the sum of the occurrences of the combinations of 2, 3, …, n-itemsets. The processes of map, combination and reduce are repeated for the remaining transactions of the cluster and they are performed parallel by all clusters. Then, the itemsets those found by all parallel processors are integrated together. Reduce on communicating processor is performed on the itemsets collected together from all parallel processors. It sums the occurrences of them. It is the total occurrences of the itemsets of the database. It is noted that as map and reduce processes are performed only on clusters and the size of the each cluster is much lesser than the size of the database, the memory requirement for each cluster is always lesser than the same for the database. Also the processing time is lesser as well as the number of transactions those are processed for each cluster is lesser than the number of transactions those are processed for the database. Moreover, there is no I/O bottlenecks and network traffic, as the communication takes place only between parallel processors and the communicating processor and the communication never takes place among parallel processors. Besides, as the existence of the itemsets are the building blocks for generating itemsets, the generation of candidate set after the generation of every frequent itemset and the examination of their existence in transactions are discarded. This characteristic also enhances the speed of processing.

The tasks of the proposed work are as follows.

i. Divide transactions into Clusters.
ii. Allocate one cluster to one processor.
iii. Start Parallelism on clusters.

a. Incorporate parallelism, if multiprocessor system or multiple systems exists.
b. Incorporate Multithreading, if a single processor exists.

iv. Map single items of transactions.

a. Find key-value pairs for single items.

v. Reduce the values of the single itemsets found in step (iv).

a. Sum the values of keys if they are indistinguishable.
b. Retain the values of keys if they are distinguishable.

vi. Repeat steps (vii) to (x) for 2, 3, 4, …, n-itemsets.
vii. Find Combinations from single itemsets.
viii. Map the itemsets found in step (vii).

a. Find key-value pairs for itemsets of 2, 3, 4, ..., n-itemsets.

ix. Reduce the values of the itemsets.

a. Sum the values of keys if they are indistinguishable.
b. Retain the values of keys if they are distinguishable.

x. Repeat steps from (iv) to (vi) for all other transactions.
xi. Merge the itemsets generated by parallel processors.
xii. Reduce further on the merged itemset.

a. Sum the values of keys if they are identical.
b. Retain the values of keys if they are not identical.

xiii. Determine Min_Support to discover frequent itemsets.
xiv. Generate association rules based on Min_Confidence.

The proposed method is shown in Fig. 1.

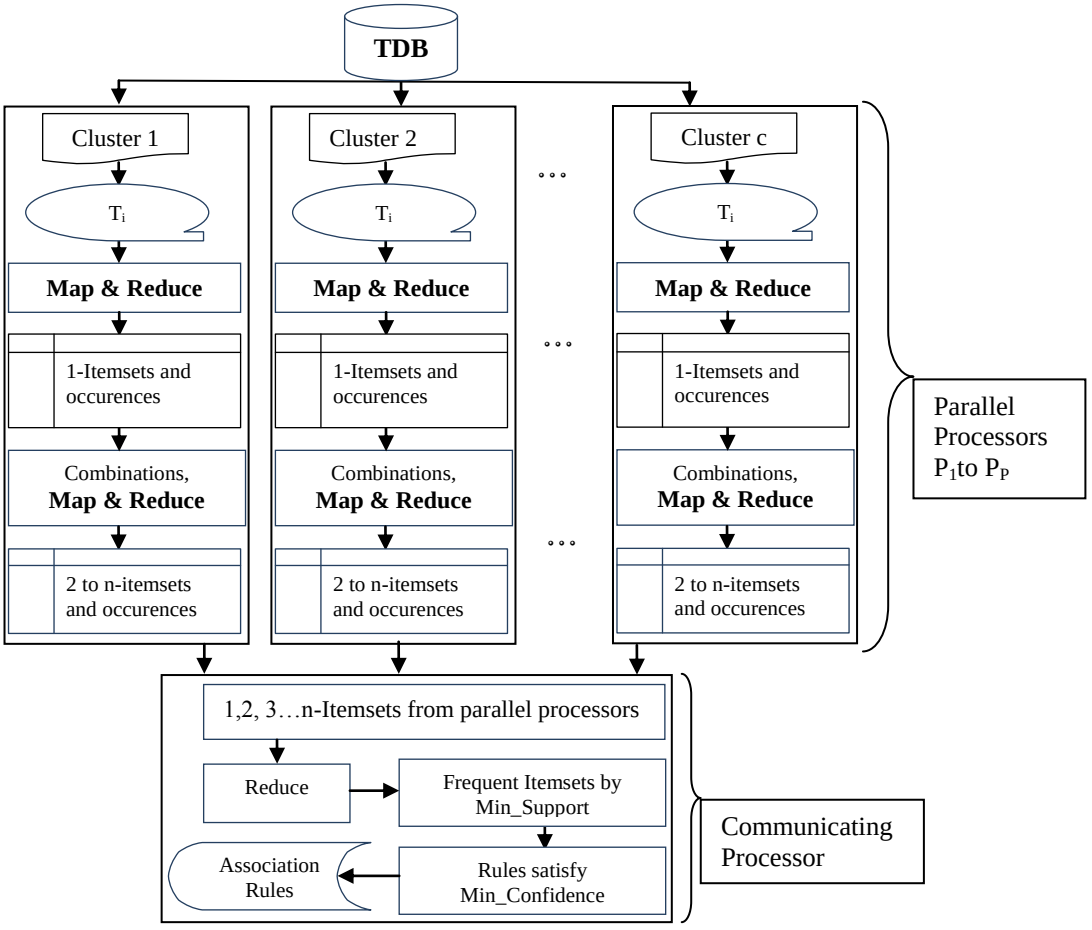

Fig.1. Map-Reduce to generate Association Rules

In order to understand the proposed method, let n be the total number of items, t be the total number of transactions, p be the number of processors to be working in parallel, c be the number of clusters which is equivalent to the number of processors or systems. Further, Let C_1, C_2 and C_c are the clusters allotted to the processors P_1, P_2 and P_p respectively. Approximately equal numbers of transactions are assigned to all clusters and they are processed sequentially. Thus, $\{T_i\}$, $\{T_j\},\ldots,\{T_l\}$ are the sets of transactions where $1\leq i\leq t/p$, $t/p+1\leq j\leq 2t/p,\ldots,(c-1)t/p\leq l\leq ct/p$ which are assigned to clusters $C_1, C_2,\ldots,C_c$ respectively and they are processed in parallel.

Let TDB be Transactional DataBase. It is assumed that an equal number of transactions are assigned to all clusters $C_1, C_2,\ldots, C_c$, in the first phase and it is computed as

$$t = qp + r \tag{1}$$

where, q is the number of transactions assigned equally to all clusters and r is the residual transactions which are outliers. If r = 0, there is no residual transactions which indicates that the transactions are equally assigned to the clusters. Otherwise, r is assigned to C_i, $1\leq i\leq p\text{-}1$ in round robin fashion. Further, $C_i \subseteq TDB$. Thus, C_i consists of $\{T_x, T_y, \ldots, T_z\}$, where the number of items in $T_x \neq T_y \neq \ldots \neq T_z$. The size of a cluster is the number of transactions in the cluster i.e. t/p and the total number possible combinations of k-itemsets where $k \geq 2$ from n-itemsets is calculated using (2)

$$C(n,k) = \sum_{k=2}^{n} \frac{P(n,k)}{k!} \tag{2}$$

where,

$$P(n,k) = \frac{n!}{(n-k)!}$$

Further the Map process outputs a key-value pair denoted by (I_{ik}, O_{Iik}) where I_{ik} denotes i^{th} item of k^{th} itemset, $1\leq i,k\leq n$ and O_{Iik} denotes the occurrences of the itemset I_{ik}. Let M_{ikj} be Map on i^{th} item of k^{th} itemset of j^{th} transaction, $1\leq i\leq n$, $1\leq j\leq t$, $1\leq k\leq n$. Thus, M_{ikj} (I_{ik}, O_{Iik}) represents mapping of i^{th} item I_{ik} of k^{th} itemset with its occurrence O_{Iik}. The item I_{ik} is the key and O_{Iik} is the value. These are key-value pairs of map. If the itemset I_{ik} occurs multiple times, a separate new entry is made for each occurrence with the value 1. Let reduce on parallel processors denoted as $RDP_{ikj}(I_i, O_{Ii})$ represents reduce process on i^{th} item of k^{th} itemset in j^{th} transaction with its occurrences O_{Ii} whereas reduce on communicating processor is denoted as $RDC_{ik}(I_i, CO_{Ii})$ represents reduce process for the cumulative occurrences of the itemset I_i of all clusters which are received from 1 to p processors. Thus,

$$M_{ikj}\left(I_{ik}, O_{I_{ik}}\right) = \left(\{I_{ik}\}, 1\right) \tag{3}$$

$$RDP_{ikj}\left(I_i, O_{I_i}\right) = \left\{ \begin{matrix} 1, \exists! I_i \in RDP_{ikj} \\ \sum_{i=1}^{n} O(I_i), Otherwise \end{matrix} \right\} \tag{4}$$

$$RDC_{ik}\left(I_i, CO_{I_i}\right) = \left\{ \begin{matrix} 1, \exists! I_i \in RDC_{ik} \\ \sum_{i=1}^{n} CO(I_i), Otherwise \end{matrix} \right\} \tag{5}$$

The proposed method is shown in Algorithm 1.

Algorithm 1: Augmented_Apriori() //Parallel Processes of Clusters

```
//T_j: j^th transaction //SIS:  Store single itemsets
//C_b:Cluster // RDP_ik: Reduce on i^th item of k^th itemset on Parallel Processors
//I_ik: i^th item of k^th itemset; // CIS_jk: CombinedItemSet of i^th item of k^th itemset
Augmented_Apriori _Parallel_Processor_Processes (C_b);
Begin
//Single Itemset Extraction;
for each transaction T_j in C_b of database D do
begin
for each item I_i∈T_j  do
begin
   SIS_jk←{ I_i }; // Single ItemSet
end
for each itemset I_i ∈ SIS_jk do
begin
MAP_ikj(I_i ,O_Ii) ←  MAP(I_i ,O_Ii);    //MAP of items I_i of  Single ItemSets  SIS_ik
end
for each itemset I_i∈MAP_ikj(I_i ,O_Ii) do
begin
RDP_ikj(I_i ,O_Ii) ←REDUCE(I_i ,O_Ii, MAP_ikj);  //Reduce of Items I_i of  Mapped Single ItemSets
end
//Combinations of itemsets
for each itemset  I_i∈SIS_jk do
begin
   for e←2 to powerset(SIS_jk) step 1 do
   begin
         CIS_jk= SIS_ik  || SIS_ik ;      //Combinations of  ItemSets
         for each itemset  I_i ∈ CIS_jk
         begin
            MAP_ikj(I_i ,O_Ii) ←  MAP(I_i ,O_Ii); //MAP of items I_i of  Combined ItemSets  CIS_ik
         end
         for each itemset I_i∈MAP_jkj(I_i ,O_Ii) do
          begin
          RDP_ikj(I_i ,O_i) ←REDUCE(I_i ,O_Ii, MAP_ikj); //Reduce of Items I_i of  Mapped Combined ItemSets
         end
    endfor //e
end // transactions
End //Augmented_Apriori_Parallel_Processor_Processes

// Map Function
MAP(key, value)
begin
      return MAP_ikj(key,1);
end

// Reduce Function
```

REDUCE(key, value, AIS_{jk})
begin
for each occurrence of *key* $\in AIS_{ik}$ do
AIS_{ik}*(value)* ← AIS_{jk}*(value*)+1;
endfor
return AIS_{ik}*(key, value);*
end

// Processes of Communicating Processors
Program Augmented_Apriori_Communicating_Processor_Processes(*t,p,Min_Supp,Min_Conf*)
//t: Number of Transactions*; p:* Number Parallel Processors*;Min_Supp*: Minimum Support;
Min_Conf: Minimum Confidence; *IAR*: Interesting Association Rule;
//CC:Compute Confidence of generated itemsets // MIS_{ik}: Merged ItemSets of i^{th} item of k^{th} itemset
//RDC_{ik}: Reduce on i^{th} item of k^{th} itemset Communicating processor
Input: t,p, *Min_Supp, Min_Conf*
Output: *IAR*
// Cluster Generation
ntc←*t*/*p*; *r*←*t*%*p*;
if $r \neq 0$ *tt*←t-*r*; endif
if *r* =0 *tt*←*t*; endif
v=1; *nntc* ← *ntc;*
for *i*←1;*i*≤*tt*;*i*←*i*+*ntc*)
begin
for j←i to *nntc* do
begin
C_v ← {T_j}; *j*←*j*+1;
end
v←*v*+1; *nntc*←*nntc*+ *ntc;*
end
// assignment of residual transactions only if $r \neq 0$
v ←1;
for *i*←*tt*+1 to *t* do
begin
C_v ← ^{T_i}; // Append Transactions with existing in clusters
v←*v*+1;
end
// Assignment of Clusters to Processors
for *i* ← 1 to *p*
P_i ← C_i
//Parallelism Begins On Cluster
Augmented_Apriori_ Parallel_Processor_Processes (C_b);// Invoking Paralle l Processes within Clusters
//Collect itemsets from parallel processors
MIS_{ik} (I_i ,O_{Ii}) ←^{ RDP_{ikj}(I_i ,O_{Ii}))}; //Merging of itemsets
for each itemset $I_i \in MIS_{ik}$ do
RDC_{ik}(I_i ,O_{Ii}) ←*REDUCE*(I_i ,O_{Ii} MIS_{ik}); *//REDUCE on Merged Itemsets*
// Constructing Rules
for each itemset I_i in RDC_{ik} do
Generate Antecedent and Consequent
AR ← Antecedent ⟶ Consequent

CC←*Min_Supp*(*AR*)/*Min_Supp*(Antecedent) //Calculating Confidence of the rules
if *CC* ≥ *Min_Conf* // *Examining with Min_Conf*
then *IAR*← *AR* // Assigning Association Rules as Interesting Association Rules
else reject
endif
end
end //Augmented_Apriori_Communicating_Processor_Processes

4. Proposed Work

To show the relevance of the proposed method, let there are two processors p=2 and the number of transactions t=10. As there are two processors, the transactions are divided into two clusters in which the first 5 transactions are assigned to the first cluster and the next 5 transactions are assigned to the second cluster using (1). As there is no residual transaction i.e. r=0, exactly equal numbers of transactions are assigned to each cluster. These two clusters with their corresponding transactions are shown in Table 1.

Table 1. Cluster 1 and Cluster 2

Cluster 1		Cluster 2	
Transactions	Itemsets	Transactions	Itemsets
T_1	I_1,I_2,I_3,I_4	T_6	I_1,I_2,I_3,I_4,I_5
T_2	I_2,I_4	T_7	I_2,I_3,I_5
T_3	I_1,I_4	T_8	I_4
T_4	I_3	T_9	I_1,I_3,I_4
T_5	I_1,I_2,I_4,I_5	T_{10}	I_2,I_3,I_4,I_5

Map-Reduce process is performed in parallel by the two processors but the transactions in each cluster are processed sequentially. Before map is invoked, the single itemsets which occur in the transactions are extracted and map is invoked to find key-value pairs with itemsets as keys and their corresponding occurrences as their values. Map assigns the value 1 to each key using (3). Thus, Map process on T_1 of cluster 1 that runs on processor 1 has generated the key-value pairs ($\{I_1\}$,1), ($\{I_2\}$,1), ($\{I_3\}$,1), ($\{I_4\}$,1). The number of the combinations is computed using (2) and the computed number of different combinations are also found from the generated single itemsets and the combinations are $\{I_1,I_2\}$, $\{I_1,I_3\}$, $\{I_1,I_4\}$ $\{I_2,I_3\}$, $\{I_2,I_4\}$, $\{I_3,I_4\}$, $\{I_1,I_2,I_3\}$, $\{I_1,I_2,I_4\}$,$\{I_1,I_3,I_4\}$, $\{I_2,I_3,I_4\}$ and $\{I_1,I_2,I_3,I_4\}$. Mapping on combinations finds ($\{I_1,I_2\}$,1), ($\{I_1,I_3\}$,1), ($\{I_1,I_4\}$,1), ($\{I_2,I_3\}$,1), ($\{I_2,I_4\}$,1), ($\{I_3,I_4\}$,1), ($\{I_1,I_2,I_3\}$,1), ($\{I_1,I_2,I_4\}$,1), ($\{I_1,I_3,I_4\}$,1), ($\{I_2,I_3,I_4\}$,1), ($\{I_1,I_2,I_3,I_4\}$,1) as key-value pairs using (3). Reduce on these key-value pairs retains the same value for the key which has only one occurrence and sums the values of the key which already exists using (4).

When T_2 of cluster 1 of processor 1 is processed by map, it finds ($\{I_2\}$,1), ($\{I_4\}$,1) and the map that follows combination discovers the key-value pair ($\{I_2,I_4\}$,1) for the combination $\{I_2,I_4\}$ of single items I_2 and I_4. Reduce on T_1 and T_2 of processor 1 produces ($\{I_1\}$,1), ($\{I_2\}$,2), ($\{I_3\}$,1), ($\{I_4\}$,2), ($\{I_1,I_2\}$,1), ($\{I_1,I_3\}$,1), ($\{I_1,I_4\}$,1), ($\{I_2,I_3\}$,1) , ($\{I_2,I_4\}$,2), ($\{I_3,I_4\}$,1), ($\{I_1,I_2,I_3\}$,1), ($\{I_1,I_2,I_4\}$,1), ($\{I_1,I_3,I_4\}$,1), ($\{I_2,I_3,I_4\}$,1) and ($\{I_1,I_2,I_3,I_4\}$,1) using (3). Since both the keys $\{I_2\}$ and $\{I_4\}$ occur twice, the values of the keys are added. As the remaining keys occur only once, their values are retained with 1 using (4). Similar map, combination and reduce are performed on all other remaining transactions of the cluster 1. Table 2 shows the results obtained by the first processor. All the transactions of the second cluster are processed in the same way using (2) (3) and (4). Table 3 shows the results obtained by the second processor. Then, the key-value pairs generated by the two processors are integrated together. Then, reduce process of communicating processor sums the values of the keys of merged key-value pairs of all clusters using (5). It is shown in Table 4 (a).

Table 2. Key-Value Pairs generated by Processor 1

Keys	Values	Keys	Values	Keys	Values	Keys	Values
$\{I_1\}$	3	$\{I_4\}$	4	$\{I_5\}$	1	$\{I_1,I_2,I_3\}$	1
$\{I_2\}$	3	$\{I_1,I_4\}$	3	$\{I_1,I_5\}$	1	$\{I_1,I_2,I_3,I_4\}$	1
$\{I_1,I_2\}$	2	$\{I_2,I_4\}$	3	$\{I_2,I_5\}$	1	$\{I_2,I_3,I_4\}$	1
$\{I_3\}$	2	$\{I_3,I_4\}$	1	$\{I_4,I_5\}$	1	$\{I_1,I_2,I_4,I_5\}$	1
$\{I_1\ I_3\}$	1	$\{I_1,I_2,I_4\}$	2	$\{I_1,I_2,I_5\}$	1	$\{I_2,I_4,I_5\}$	1
$\{I_2\ I_3\}$	1	$\{I_1,I_3,I_4\}$	1	$\{I_1,I_4,I_5\}$	1		

Table 3. Key-Value Pairs generated by Processor 2

Keys	Values	Keys	Values	Keys	Values	Keys	Values	Keys	Values
$\{I_1\}$	2	$\{I_4\}$	4	$\{I_1,I_3,I_4\}$	2	$\{I_1,I_2,I_3,I_4\}$	1	$\{I_3,I_5\}$	3
$\{I_2\}$	3	$\{I_1,I_4\}$	2	$\{I_2,I_3,I_4\}$	2	$\{I_1,I_3,I_4,I_5\}$	1	$\{I_1,I_3,I_5\}$	1
$\{I_1,I_2\}$	1	$\{I_2,I_4\}$	2	$\{I_1,I_2,I_3\}$	1	$\{I_1,I_2,I_4,I_5\}$	1	$\{I_1,I_5\}$	1
$\{I_3\}$	4	$\{I_3,I_4\}$	3	$\{I_1,I_2,I_4\}$	1	$\{I_1,I_2,I_3,I_4,I_5\}$	1	$\{I_2,I_5\}$	3
$\{I_1\ I_3\}$	2	$\{I_4,I_5\}$	2	$\{I_1,I_2,I_5\}$	1	$\{I_1,I_2,I_4,I_5\}$	1	$\{I_2,I_4,I_5\}$	2
$\{I_2\ I_3\}$	3	$\{I_5\}$	3	$\{I_1,I_4,I_5\}$	1	$\{I_2,I_3,I_4,I_5\}$	2	$\{I_2,I_3,I_5\}$	3
$\{I_1,I_2,I_3,I_5\}$	1								

Table 4 (b) shows some of the interesting associations with *Min_Support* ≥ 3 and *Min_Confidence* $\geq 60\%$.

Table 4. (a) Key-Value Pairs generated by both the Processors 1 and 2; (b) Interesting Associations with their Confidence

Keys	Values	Keys	Values	Keys	Values
$\{I_1\}$	5	$\{I_1,I_2,I_3,I_4\}$	2	$\{I_1,I_2,I_3\}$	2
$\{I_2\}$	6	$\{I_5\}$	4	$\{I_1,I_2,I_4\}$	3
$\{I_3\}$	6	$\{I_1,I_5\}$	2	$\{I_1,I_3,I_4\}$	3
$\{I_4\}$	8	$\{I_2,I_5\}$	4	$\{I_2,I_3,I_4\}$	3
$\{I_1,I_2\}$	3	$\{I_4,I_5\}$	3	$\{I_1,I_4,I_5\}$	2
$\{I_1,I_3\}$	3	$\{I_1,I_2,I_5\}$	2	$\{I_2,I_3,I_5\}$	3
$\{I_1,I_4\}$	5	$\{I_2,I_4,I_5\}$	3	$\{I_1,I_2,I_3,I_5\}$	1
$\{I_2,I_3\}$	4	$\{I_1,I_2,I_4,I_5\}$	2	$\{I_2,I_3,I_4,I_5\}$	2
$\{I_2,I_4\}$	5	$\{I_3,I_5\}$	3	$\{I_1,I_2,I_3,I_4,I_5\}$	1
$\{I_3,I_4\}$	4	$\{I_1,I_3,I_5\}$	1	$\{I_1,I_3,I_4,I_5\}$	1

(a)

Rules	Confidence	Rules	Confidence
$I_1 \rightarrow I_4$	100%	$I_5 \rightarrow I_4$	80%
$I_4 \rightarrow I_1$	62.5%	$I_5 \rightarrow I_2I_3$	60%
$I_2 \rightarrow I_3$	66.7%	$I_2I_3 \rightarrow I_5$	75%
$I_3 \rightarrow I_2$	66.7%	$I_3I_5 \rightarrow I_2$	100%
$I_2 \rightarrow I_4$	83.3%	$I_2I_5 \rightarrow I_3$	75%
$I_4 \rightarrow I_2$	62.5%	$I_5 \rightarrow I_2I_4$	60%
$I_2 \rightarrow I_5$	66.7%	$I_2I_4 \rightarrow I_5$	60%
$I_5 \rightarrow I_2$	80%	$I_2 I_5 \rightarrow I_4$	75%
$I_5 \rightarrow I_3$	60%	$I_4I_5 \rightarrow I_2$	75%

(b)

5. Results and Discussion

The Proposed Augmented Apriori enhances the performance of the generation of association rules. It increases the processing speed and decreases the requirement of memory space. Apart, it requires only one scan on the database which also improves the performance in terms of processing speed. Thus, quick response is provided. In the classical Apriori, the transactions are scanned for each candidate set and the number of candidate sets generated relies on the total number of items in the database. But in the proposed Augmented Apriori, the transactions are scanned only once for extracting single itemsets. Fig. 2 (a) and (b) describe that the Classical Apriori scans each transaction for the total number of itemsets in the database whereas the proposed Augmented Apriori scans each transaction for the total number of single items in the transaction.

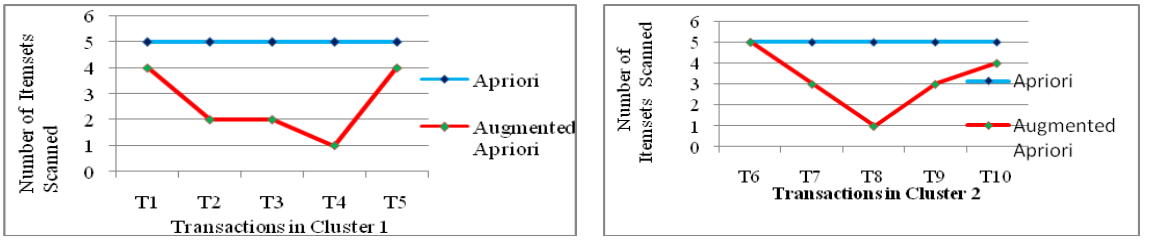

Fig.2. (a) Comparison of scans on the itemsets of Cluster 1; (b) Comparison of scans on the itemsets of Cluster 2

Also, classical Apriori performs sequential process on all the transactions of the entire database and sequential process needs more time to produce associations. But Augmented Apriori performs parallel process on the clusters of the database and it needs lesser time to produce associations.

Fig. 3 shows that in the classical Apriori, the number of scans on the total number of items of the entire database determines the processing time whereas in the proposed Augmented Apriori, the number of scans on the total number of items of the transactions of each cluster determines the processing time. It is noted that the number of the transactions in each cluster is always lesser than that of the database, so the total number of items in each cluster is also lesser than that of the database. It also reveals that when scalability of items increases, the Augmented Apriori outperforms well than Classical Apriori.

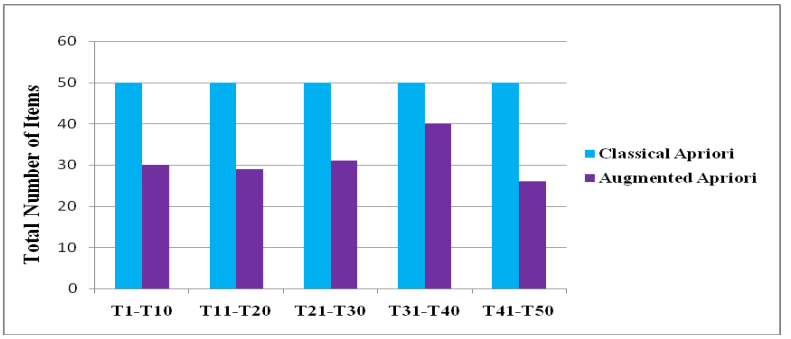

Fig.3. Comparison of Scalability issues Based on the Number of Items in the Transactions

Fig. 4 reveals that the performance of the classical Apriori suffers when the scalability increases in terms of transactions whereas the proposed Augmented Apriori scales up well. It is noted that the increase in the number of transactions decreases the performance of the classical Apriori whereas in the proposed Augmented Apriori, the increase in the number of transactions does not degrade the performance and it is caused by the distribution of tasks which balances the load in parallel. The proposed Augmented Apriori scales up well with the number of transactions. It is also observed that Augmented Apriori generates specific patterns with high speed because itemsets are generated using parallelism on clusters.

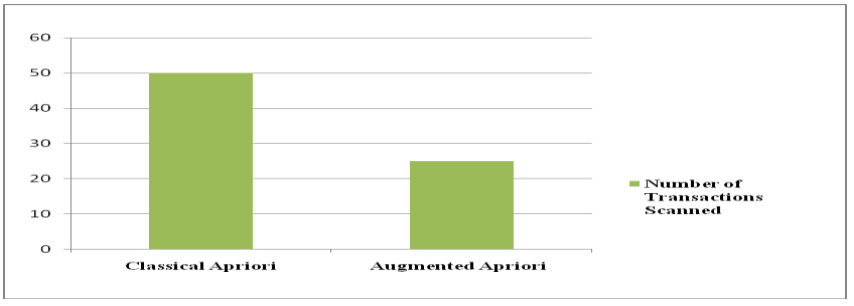

Fig.4. Comparison of Scalability issues based on the number of transactions

From tables 2 and 3, it is revealed that only lesser number transactions are processed by each processor in the proposed method. So the speed of the generation of association rules is increased. Also it minimizes the time for retrieval, as database is accessed only once for clustering and also each transaction is accessed only once for retrieving single itemsets and it is not accessed to find every frequent itemset of 2, 3, ..., n-itemsets. It is also evident from Fig. 1, the proposed augmented apriori requires less memory space, as only clusters are stored and maintained by each processor. Moreover, the requirement of memory space is further minimized by storing the output of Map and Reduce processes of parallel processors only in buffers and by permanently storing only the output of Reduce process of communicating processor.

6. Conclusion

A novel augmented Apriori algorithm has been proposed in this paper and it simulates Map-Reduce of Big Data with parallelism in a distributed platform. The problems of discovering association rules from big data and inefficiencies of association rule mining techniques have been resolved. Multiple scans on the database is minimized to single scan. The database is scanned only once to cluster the transactions in order to distribute

them among multiple processors. As only the clustering process requires database scan and the database is not scanned for each candidate itemset to find frequent itemsets, single scan is sufficient. Moreover, since the transactions are clustered based on the order in which they appear in the database, processing time need not be spent for finding closeness among transactions as it is done for clustering data points. As the database is scanned only once for clustering and all other processes related to itemset generation take place in parallel only in the clusters, less processing time is accomplished.

Quick response is also achieved by the proposed Augmented Apriori due to the distribution of the transactions into clusters which work simultaneously among multiple processors. As there is no communication among the multiple processors, this kind of distribution resolves I/O bottlenecks, network traffic and data collision in map-reduce. Besides, as all 1, 2, 3,…, n-itemsets are constructed only from the existing items, there is no need to process the transactions unnecessarily and each transaction is processed only once to extract single items. The memory space requirement for each cluster is always lesser than the same of the database, since the size of each cluster is lesser than the size of the database.

The result clearly shows that the proposed Augmented Apriori performs well in terms of memory and speed than the classical apriori. This unique and innovative idea enhances the performance of the association rule mining.

References

[1] Jiawei Han and Michelin Kamber, "Data Mining Concepts and Techniques", 2nd Ed, Morgan Kaufmann Publisher, 2006.

[2] Jongwook Woo, "Apriori-Map/Reduce Algorithm", In the Proceeding of International Conference on Parallel and Distributed Processing Techniques and Applications, 2012.

[3] Sanjay Rathee, Manohar Kaul and Arti Kashyap, "R-Apriori: An Efficient Apriori based Algorithm on Spark", In the proceedings of Information and Knowledge Mangaement, ACM Digital Library, Oct 2015.

[4] Jeffrey Dean and Sanjay Ghemawat,"Map Reduce: Simplified Data Processing on Large Clusters", Google Inc, Google Research Publication.

[5] R. Sumithra, Sujni Paul and D. Ponmary Pushpa Latha, "A hybrid algorithm combining weighted and hash T apriori algorithms in Map Reduce model using Eucalyptus cloud platform", WSEAS Transaction Computers,Vol. 14, 2015.

[6] Shafali Agarwal and Zeba Khanam, "Map Reduce: A Survey Paper on Recent Expansion", International Journal of Advanced Computer Science and Applications, Vol. 6, 2015.

[7] Sonali Satija and Dr. Rajender Nath, "Performance Improvement of Apriori Algorithm Using Hadoop", International Journal of Advanced Research in Computer Science and Software Engineering, June 2015.

[8] Vijay Swaroop, "A Line of Attack to Accentuate FIM in Cloud Computing", International Journal of Advanced Research in Computer Science and Software Engineering, Vol. 5, Jan 2015.

[9] Zhang Danping, Yu Haoran and Zheng Linyu, "Apriori Algorithm Research Based on Map-Reduce in Cloud Computing Environments", The Open Automation and Control Systems Journal, Vol. 6, 2014.

[10] Sudhakar Singh, Rakhi Garg and P. K. Mishra, "Review of Apriori Based Algorithms on Map Reduce Framework", ICC, 2014.

[11] Y. Venkata Raghavarao, L. S. S Reddy and A. Govardhan, "Map Reducing Stream Based Apriori in Distributed Big Data Mining", International Journal of Advanced Research in Computer Science and Software Engineering, Vol. 4, July 2014.

[12] Mr. Uday K. Kakkad and Prof. Rajanikanth Aluvalu, "Association Rule", Journal of Computer Engineering (IOSR-JCE), Vol.16, Mar-Apr 2014.

[13] A.Pradeepa and Dr.Antonyselvadoss Thanamani, "Parallelized Comprising For Apriori Algorithm Using Map reduce Framework", International Journal of Advanced Research in Computer and Communication Engineering, Vol. 2, Nov 2013.

[14] A.Ezhilvathani and Dr. K. Raja,"Implementation Of Parallel Apriori Algorithm On Hadoop Cluster", International Journal of Computer Science and Mobile Computing, Vol. 2, April 2013.
[15] Anjan K Koundinya, Srinath N K, K A K Sharma, Kiran Kumar, Madhu M N and Kiran U Shanbag, "Map/Reduce Design and Implementation of Apriori algorithm For Handling Voluminous Data-Sets", Advanced Computing: An International Journal, Vol. 3, Nov 2012.
[16] G.Vamsi Krishna," Prediction of Rainfall Using Unsupervised Model based Approach Using K-Means Algorithm", International Journal of Mathematical Sciences and Computing, PP 11-20, July 2015.
[17] Deepa B. Patila , Yashwant V. Dongre, "A Fuzzy Approach for Text Mining", International Journal of Mathematical Sciences and Computing, 4, PP 34-43, Nov 2015.

Authors' Profiles

R.Akila received her B.Sc and M.Sc degrees in Computer Science from Seethalakshmi Ramaswami College, affiliated to Bharathidasan University, Tiruchirappalli, Tamilnadu, India. She received M.Phil degree in Computer Science from Mother Teresa Women's University, Kodaikanal, Tamilnadu, India. She has worked in E.M.G.Yadava Women's College, Madurai and in Cauvery College for Women, Trichy. She is presently working as an Assistant Professor in the P.G. and Research Department of Computer Science, Nehru Memorial College, Puthanampatti, Tiruchirappalli. She is pursuing PhD degree in Computer Science at Bharathidasan University. She has published and presented around 7 research papers at international/nationals journals and conferences. Her research interests include Data Mining techniques, Algorithms, Big Data and fuzzy systems.

K. Mani received his MCA and M.Tech from the Bharathidasan University, Trichy, India in Computer Applications and Advanced Information Technology respectively. Since 1989, he has been with the Department of Computer Science at the Nehru Memorial College, affiliated to Bharathidasan University where he is currently working as an Associate Professor. He completed his PhD in Cryptography with primary emphasis on evolution of unified framework for enhancing the security and optimizing the run time in cryptographic algorithms. He published and presented around 35 research papers at international/nationals journals and conferences. His research interests include Cryptography, Data Mining and Coding Theory.

How to cite this paper: R.Akila, K.Mani,"Augmented Apriori by Simulating Map-Reduce", International Journal of Mathematical Sciences and Computing(IJMSC), Vol.3, No.4, pp.52-66, 2017.DOI: 10.5815/ijmsc.2017.04.05