An Investigation into the Central Data Warehouse based Association Rule Mining

Gurpreet Singh Bhamra
DCSE, M. M. University,
Mullana-133203,Haryana,India

Anil Kumar Verma
DCSE, Thapar University,
Patiala-147004, Punjab, India

Ram Bahadur Patel
DCSE, G. B. Pant Engineering College,
Ghurdauri-246194, Uttarakhand, India

ABSTRACT

Data Mining(DM) technique is used to mine interesting hidden knowledge from large databases using various computational techniques/tools. Association Rule Mining(ARM) today is one of the most important aspects of DM tasks. In ARM all the strong association rules are generated from the Frequent Itemsets. In this study a central Data Warehouse based client-server model for ARM is designed, implemented and tested. The Outcome of this investigation and the advantages of software agents forms the base and motivation of using software agent technology in Distributed Data Mining.

General Terms:

Data Mining,Distributed Data Mining

Keywords:

Data Warehouse, Frequent Itemsets, Association Rule Mining

1. INTRODUCTION

The exponential growth in the stored data generated an urgent need for new technique that can intelligently transform this enormous amount of data into useful knowledge. Consequently, DM has become a powerful technology with a great potential to help companies focus on the most important information in the huge data they have collected about the behavior of their customers and potential customers. DM involves the use of computational techniques/tools such as classification, association rules, clustering, etc. to automatically extract some interesting and valid data patterns or trends representing knowledge, implicitly stored in large databases [9, 10]. These tools can include statistical models,algorithms, and machine learning methods. In a classical knowledge discovery technique in distributed environment, a single centrally integrated data repository called Data Warehouse (DW) is created and then DM techniques are used to mine the data and extract the knowledge [15].

The central DW based approach, however, is ineffective or infeasible because of heavy storage and computational costs involved in managing data form the ever increasing and updated distributed resources where data is produced continuously in streams. Network communication cost is also involved while transferring huge data over the wired or wireless network in a limited network bandwidth scenario. It is also not desirable to centrally collect the privacy-sensitive raw distributed data of the business organizations like banking, and telecommunication as they want only knowledge to be exchanged globally. Data from modern business organizations are not only geographically distributed but also horizontally or vertically fragmented making it difficult if not possible to combine them in a central location. Performance and scalability of a DM application can be increased by distributing the workload among sites [16]. Resource constraints of distributed and mobile environment are not considered in algorithms for central DW based DM as certain data sets are immovable in practice. This paper is an attempt to practically investigate the classical central DW based client-server model for association rule mining.

Rest of the sections are organized as follows. Section 2 describes the ARM concepts. Important algorithms for ARM are discussed in section 3. Section 4 completely describes the Central DW based ARM approach with subsections 4.1 for preliminary notations, 4.2 for synthetic data sets used and 4.3 for architectural layout or working modal along with the various algorithms involved in the system. Experimentation and result evaluation is discussed in section 5 and finally the article is concluded in section 6.

2. ASSOCIATION RULE MINING (ARM)

Let $DB = \{T_j, j = 1 \ldots D\}$ be a transactional dataset of size D where each transaction T is assigned an identifier (TID) and $I = \{d_i, i = 1 \ldots m\}$, total m data items in DB. A set of items in a particular transaction T is called itemset or pattern.An itemset $P = \{d_i, i = 1 \ldots k\}$, which is a set of k data items in a particular transaction T and $P \subseteq I$, is called k-itemset.Support of an itemset, $s(P) = \frac{No\_of\_T\_containing\_P}{D}\%$, is the frequency of occurrence of itemset P in DB, where $No\_of\_T\_containing\_P$ is the support count (sup_count) of itemset P. Frequent itemsets (FIs) are the itemsets that appear in DB frequently, i.e., if $P \geq min\_th\_sup$ (given minimum threshold support), then P is a Frequent k-Itemset.Finding such FIs plays an essential role in mining the interesting relationships among itemsets. Frequent Itemsets Mining (FIM) is the task to find the set of all subsets of FIs in a transactional database. It is CPU and input/output intensive task, mainly because of the large amount of itemsets generated and large size of the datasets involved in progress [10].

Association Rules (ARs) first introduced in [1], are used to discover the associations (or co-occurrences) among items in a database. ARs can be used to find the patterns of customers purchase such as how the transaction of buying some goods will impact on the transactions of buying others. Such rules can be implemented to design the goods shelves, to manage the stock and to classify the customers according to the purchase patterns.AR is an implication expression of the forms $P \Rightarrow Q[support, confidence]$ where , $P \subset I$,$Q \subset I$ and P and Q are disjoint itemsets, i.e., $P \cap Q = \phi$. An AR is measured in terms of its support and confidence factor where:

—Support $s(P \Rightarrow Q) = p(P \cup Q) = \frac{No\_of\_T\_containing\_both\_P\_and\_Q}{D}\%$: the probability of both P and Q appearing in T,we can say that s % of the transactions support the rule$P \Rightarrow Q$, $0 \leq s \leq 1.0$ or $0\% \leq s \leq 100\%$

—Confidence $c(P \Rightarrow Q) = p(Q \mid P) = \frac{s(P \Rightarrow Q)}{s(P)} = \frac{sup\_count(P \Rightarrow Q)}{sup\_count(P)}\%$:the conditional probability of Q given P, we can say that when itemset P occurs in a transaction there are $c\%$ chances that itemset Q will occur in that transaction, $0 \leq c \leq 1.0$ or $0\% \leq c \leq 100\%$

An Association rule $P \Rightarrow Q$ is said to be strong if

(1) $s(P \Rightarrow Q) \geq min\_th\_sup$,i.e., support of the rule is greater than or equal to the given minimum threshold support and
(2) $c(P \Rightarrow Q) \geq min\_th\_conf$, i.e., confidence of the rule is greater than or equal to the given minimum threshold confidence

Association Rule Mining(ARM) today is one of the most important aspects of DM tasks. In ARM all the strong association rules are generated from the FIs. The ARM can be viewed as a two-step process [2, 17].

(1) Find all frequent k-itemsets (L_k)
(2) Generate Strong Association Rules from (L_k)
    (a) For each frequent itemset $l \in L$, generate all non empty subsets of l.
    (b) For every non empty subset s of l , output the rule "$s \Rightarrow (l - s)$", if $\frac{sup\_count(l)}{sup\_count(s)} \geq min\_th\_conf$

3. ALGORITHMS FOR MINING FREQUENT ITEMSETS

FIM, being the first step in ARM task, has lots of proposed algorithms both sequential and parallel. Since ARM is dedicated to handle the huge amount of data so time complexity and resource complexity in such algorithms are carefully considered. These algorithms can be classified into two categories:

—Candidate-generation-and-subset-test approach
—Pattern-growth approach

The first category, the candidate-generation-and-subset-test approach, such as Apriori algorithm [3], is directly based on the downward closure property of the FIs, i.e., if a set cannot pass a test, all of its supersets will fail the same test as well or all nonempty subsets of a FI must also be frequent or any (k-1)-itemset that is not frequent cannot be a subset of a frequent k-itemset. This property is used in subset testing of FIs. Apriori algorithm [3] has emerged as one of the best FIM and subsequently ARM algorithms since ARs can be straightforwardly generated from a set of FIs. It has been the basis of many subsequent ARM and/or ARM-related algorithms. Apriori algotirhm iteratively identifies FIs in the data by making use of the downward closure property of the itemsets in the generation of candidate (possibly frequent) itemset where a candidate itemset is confirmed as frequent only when all its subsets are identified as frequent in the previous pass.

The Apriori algorithm performs repeated scan of the database, successively computing support-counts for sets of single items, pairs, triplets and so on. At the end of each pass, sets that fails to reach the required threshold support are eliminated and candidates for the next pass are constructed as supersets of the remaining (frequent) sets. Since no set can be frequent which has an infrequent subset, this procedure guarantees that all frequent sets will be found. Since the number of database passes of the Apriori algorithm equals the size of the maximal frequent itemset, it scans the darabase k times even when only one k-frequent itemset exists. The drawback of this method is that if the dataset is very large, the required multiple database scans can be one of the limiting factors. Working model and implementation of Apriori algorithm can also be found in [5]. Many algorithms have been proposed, directed at improving the performance of the Apriori algorithm using different types of approaches. An analysis of the best known algorithms can be found in [14, 13].

A second category of methods, pattern-growth methods has been proposed such as FP-growth [12] and Apriori-TFP[8] have been proposed. These algorithms typically operate by recursively processing a tree structure into which the input data has been encoded. In the study shown by [11], a novel FI tree structure, FP-tree was proposed. FP-tree is an extended prefix-tree structure for strong compressed information about FIs. It consists of one root labeled as NULL and a set of item prefix sub trees as the descendents of the root. A frequent-item header table is also kept to link all the transactions containing that item. Each node in the item prefix sub tree consists of three fields: item-name, count and node link where item-name registers which item this node represents, count registers the number of transactions represented by the path reaching this node and node-link links to the next node in the FP-tree carrying the same item name or NULL if there is none.

Based on the FP-tree [11], an itemset fragment growth algorithm, FP-growth [12] was designed to avoid the costly generation of large number of candidate sets. The FP-growth is a partition-based, divide and conquer method used to decompose the mining task into set of smaller tasks for mining confined itemsets in conditional databases , thereby dramatically reducing the search space. The size of the FP-tree is usually small and will not grow exponentially. FP-growth method is efficient and scalable for mining both long and short FIs and faster than Apriori algorithm but has some disadvantages. Its first principal drawback is that because FP-trees are repeatedly generated, FP-growth can have significant storage requirements. Secondly, larger number of links makes it difficult to distribute the tree. These drawbacks are particularly significant with respect to the dense datasets.

In general, no single FIM/ARM algorithm has been identified to fit all types of data [4]. Real datasets can be sparse and/or dense according to their applications. For example, for telecommunication data analysis, calling patterns for home users vs. business users can be very different: some are frequent and dense (e.g. to family members and close friends) while others are large and sparse. Similar situations arise for market basket analysis, etc. It is hard to select an appropriate ARM method. Large applications need more scalability. Many existing methods are efficient when the data set is not very large. Otherwise, their core data structure(such as FP-tree) or the intermediate results(e.g. the set of candidates in Apriori or the recursively generated sub-trees in FP-growth) may not fit in main memory and may cause thrashing.

4. CENTRAL DW BASED ARM

4.1 Preliminaries and Definitions

A client-server based framework for mining the strong association rules from central DW can be formally described as:

—$S = \{S_i, i = 1 \ldots n\}$, n distributed sites
—$S_{CENTRAL}$,a Central site
—$DB_i = \{T_j, j = 1 \ldots D_i\}$, Transactional Data Set of size D_i at the local site S_i, where each transaction T_j is assigned an identifier (TID)
—$I = \{d_l, l = 1 \ldots m\}$, total m data items in each DB_i
—$DB = \bigcup_{i=1}^{n} DB_i$, the Central Data Warehouse at $S_{CENTRAL}$ of size $D = \bigcup_{i=1}^{n} D_i$
—δ_1 , start time of transferring DB_i over the network at site S_i
—δ_2 , end time of receiving DB_i at $S_{CENTRAL}$
—$\delta = \delta_2 - \delta_1$, total network time taken in dispatching DB_i at S_i
—min_th_sup, minimum threshold support
—min_th_conf, minimum threshold confidence
—L^{FI}, the list of frequent k-itemsets at $S_{CENTRAL}$
—L^{FISC}, the list of support count of every frequent k-itemsets in L^{FI}
—L^{SAR}, the list of strong association rules at$S_{CENTRAL}$

4.2 Data Sets

Data set consists of total 11250 transactions of 10 items which are horizontally partitioned and these horizontally partitioned synthetic transactional datasets (DB_i) are created and stored across three distributed sites S_1,S_2 and S_3, with 3500, 3850 and 3900 transactions respectively using a Synthetic Transactional Dataset Generator(TDSGenerator v1.0) tool [6].These data sets use Boolean values to show whether an item exists in a transaction. Table 1 shows size and density of each DB_i and snapshot of each DB_i at distributed sites are shown in Figure 1.

Table 1. Size and density of each DB_i.

Site Name	No. of Items	No. of Transactions	Approximate Density	Size of Data Set
S1	10	3500	60%	172KB
S2	10	3850	50%	192KB
S3	10	3900	65%	192KB

4.3 Framework for Central DW based ARM

Figure 2 shows the client-server model for central DW based ARM approach. Various components involved in the framework are:

(1) *Dataset Dispatcher*: It is a client application running at each distributed site S_i. It dispatches DB_i over the network to the *Central DW Manager* Server application running at $S_{CENTRAL}$ and keeps track of the δ. Algorithm 1 describes the working of *Dataset Dispatcher*.
(2) *Central DW Manager*: It is a server application running at $S_{CENTRAL}$. It handles all the incoming requests from the Dataset Dispatcher. It receives, stores each DB_i and waits continuously till all the clients have dispatched their DB_i. Algorithm 2 describes the working of *Central DW Manager*. Various other assistant components of *Central DW Manager* application are:-
  (a) *Dataset Merger*: It merges each DB_i into a single DB. See Algorithm 3.
  (b) *FI and SC Generator*: It scans DB and generates L^{FI} and L^{FISC} to be further processed by *SAR Harvester*. Apriori [3] algorithm is used to create L^{FI} with the given min_th_sup . See Algorithm 4 to Algorithm 6.
  (c) *SAR Harvester*: It generates the strong association rules from L^{FI} and L^{FISC} with the constraint of given min_th_conf. See Algorithm 7.

Algorithm 1 DATASETDISPATCHER

Input: $DB_i, Transactional\ Data\ Set\ of\ size\ D_i\ at\ S_i$
Output: $\delta, Total\ network\ time\ taken\ in\ dispatching\ DB_i$

```
1: procedure DATASETDISPATCHER($DB_i$)
2:     $\alpha \leftarrow open\ a\ new\ socket\ with\ CentralDWManager$
3:     $\beta \leftarrow bind\ ObjectInputStream\ at\ \alpha$
4:     $\chi \leftarrow bind\ ObjectOutputStream\ at\ \alpha$
5:     $\xi \leftarrow bind\ ObjectInputStream\ with\ local\ DB_i$
6:     $A \leftarrow read\ the\ entire\ DB_i\ object\ at\ \xi\ stream$
7:     $\delta_1 \leftarrow get\ start\ time\ for\ dataset\ transfer$
8:     $write\ A\ on\ \chi\ stream$
9:     $\delta_2 \leftarrow read\ end\ time\ received\ on\ \beta\ stream$
10:    $\delta \leftarrow \delta_2 - \delta_1$
11:    $Close\ all\ the\ opened\ channels$
12:    return $\delta$
13: end procedure
```

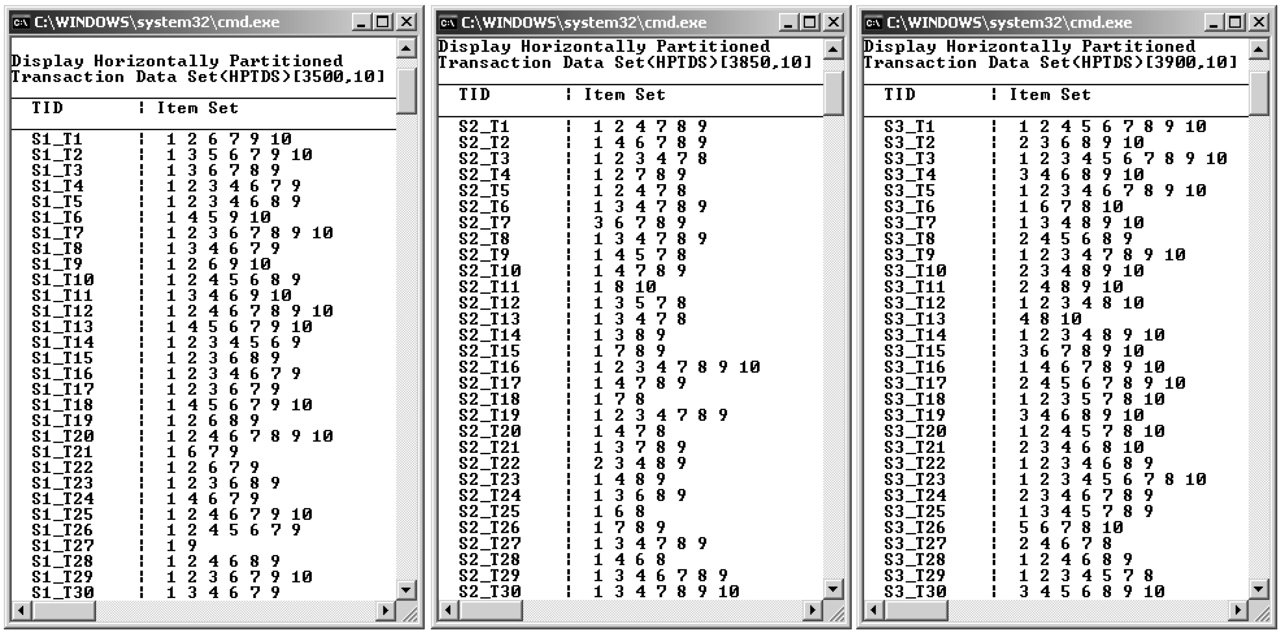
Algorithm 3 DATASETMERGER

Input: $\{DB_i, i = 1 \cdots n\}$
Output: $DB, AggregatedDataset$

```
1: procedure DATASETMERGER($\{DB_i, i = 1 \cdots n\}$)
2:     for all $DS \in \{DB_i, i = 1 \cdots n\}$ do
3:         $\alpha \leftarrow bind\ ObjectInputStream\ with\ DS$
4:         $A \leftarrow read\ the\ entire\ DS\ object\ at\ \alpha\ stream$
5:         $add\ A\ into\ a\ single\ unit\ DB$
6:     end for
7:     $Close\ all\ the\ opened\ channels$
8:     return $DB$
9: end procedure
```

5. EXPERIMENT AND RESULTS

All the components of the framework are implemented in java language. Table 2 describes the required configuration for the framework. Figure 3 shows the snapshots of the running state of the Dataset Dispatcher at each distributed site.Time taken by each Dataset Dispatcher in transferring DB_i over the network is shown in Figure 4. Figure 5 shows the snapshot of the running state of the CentralDWManager at $S_{CENTRAL}$. L^{FI} and L^{FISC} lists generated by FIandSCGenerator module with 20% min_th_sup is shown in Figure 6.
Strong Association Rules generated by SAR_Harvestor module with 50% min_th_conf are shown in Figure 7. Due to space constraints strong rules for 3-itemsets, and 2-itemsets are not shown



```
Display Horizontally Partitioned
Transaction Data Set(HPTDS)[3500,10]

   TID        ¦ Item Set

   S1_T1      ¦  1 2 6 7 9 10
   S1_T2      ¦  1 3 5 6 7 9 10
   S1_T3      ¦  1 3 6 7 8 9
   S1_T4      ¦  1 2 3 4 6 7 9
   S1_T5      ¦  1 2 3 4 6 8 9
   S1_T6      ¦  1 4 5 9 10
   S1_T7      ¦  1 2 3 6 7 8 9 10
   S1_T8      ¦  1 3 4 6 7 9
   S1_T9      ¦  1 2 6 9 10
   S1_T10     ¦  1 2 4 5 6 8 9
   S1_T11     ¦  1 3 4 6 9 10
   S1_T12     ¦  1 2 4 6 7 8 9 10
   S1_T13     ¦  1 4 5 6 7 9 10
   S1_T14     ¦  1 2 3 4 5 6 9
   S1_T15     ¦  1 2 3 6 8 9
   S1_T16     ¦  1 2 3 4 6 7 9
   S1_T17     ¦  1 2 3 6 7 9
   S1_T18     ¦  1 4 5 6 7 9 10
   S1_T19     ¦  1 2 6 8 9
   S1_T20     ¦  1 2 4 6 7 8 9 10
   S1_T21     ¦  1 6 7 9
   S1_T22     ¦  1 2 6 7 9
   S1_T23     ¦  1 2 3 6 8 9
   S1_T24     ¦  1 4 6 7 9
   S1_T25     ¦  1 2 4 6 7 9 10
   S1_T26     ¦  1 2 4 5 6 7 9
   S1_T27     ¦  1 9
   S1_T28     ¦  1 2 4 6 8 9
   S1_T29     ¦  1 2 3 6 7 9 10
   S1_T30     ¦  1 3 4 6 7 9
```

```
Display Horizontally Partitioned
Transaction Data Set(HPTDS)[3850,10]

   TID        ¦ Item Set

   S2_T1      ¦  1 2 4 7 8 9
   S2_T2      ¦  1 4 6 7 8 9
   S2_T3      ¦  1 2 3 4 7 8
   S2_T4      ¦  1 2 7 8 9
   S2_T5      ¦  1 2 4 7 8
   S2_T6      ¦  1 3 4 7 8 9
   S2_T7      ¦  3 6 7 8 9
   S2_T8      ¦  1 3 4 7 8 9
   S2_T9      ¦  1 4 5 7 8
   S2_T10     ¦  1 4 7 8 9
   S2_T11     ¦  1 8 10
   S2_T12     ¦  1 3 5 7 8
   S2_T13     ¦  1 3 4 7 8
   S2_T14     ¦  1 3 8 9
   S2_T15     ¦  1 7 8 9
   S2_T16     ¦  1 2 3 4 7 8 9 10
   S2_T17     ¦  1 4 7 8 9
   S2_T18     ¦  1 7 8
   S2_T19     ¦  1 2 3 4 7 8 9
   S2_T20     ¦  1 4 7 8
   S2_T21     ¦  1 3 7 8 9
   S2_T22     ¦  2 3 4 8 9
   S2_T23     ¦  1 4 8 9
   S2_T24     ¦  1 3 6 8 9
   S2_T25     ¦  1 6 8
   S2_T26     ¦  1 7 8 9
   S2_T27     ¦  1 3 4 7 8 9
   S2_T28     ¦  1 4 6 8
   S2_T29     ¦  1 3 4 6 7 8 9
   S2_T30     ¦  1 3 4 7 8 9 10
```

```
Display Horizontally Partitioned
Transaction Data Set(HPTDS)[3900,10]

   TID        ¦ Item Set

   S3_T1      ¦  1 2 4 5 6 7 8 9 10
   S3_T2      ¦  2 3 6 8 9 10
   S3_T3      ¦  1 2 3 4 5 6 7 8 9 10
   S3_T4      ¦  3 4 6 8 9 10
   S3_T5      ¦  1 2 3 4 6 7 8 9 10
   S3_T6      ¦  1 6 7 8 10
   S3_T7      ¦  1 3 4 8 9 10
   S3_T8      ¦  2 4 5 6 8 9
   S3_T9      ¦  1 2 3 4 7 8 9 10
   S3_T10     ¦  2 3 4 8 9 10
   S3_T11     ¦  2 4 8 9 10
   S3_T12     ¦  1 2 3 4 8 10
   S3_T13     ¦  4 8 10
   S3_T14     ¦  1 2 3 4 8 9 10
   S3_T15     ¦  3 6 7 8 9 10
   S3_T16     ¦  1 4 6 7 8 9 10
   S3_T17     ¦  2 4 5 6 7 8 9 10
   S3_T18     ¦  1 2 3 5 7 8 10
   S3_T19     ¦  3 4 6 8 9 10
   S3_T20     ¦  1 2 4 5 7 8 10
   S3_T21     ¦  2 3 4 6 8 10
   S3_T22     ¦  1 2 3 4 6 8 9
   S3_T23     ¦  1 2 3 4 5 6 7 8 10
   S3_T24     ¦  2 3 4 6 7 8 9
   S3_T25     ¦  1 3 4 5 7 8 9
   S3_T26     ¦  5 6 7 8 10
   S3_T27     ¦  2 4 6 7 8
   S3_T28     ¦  1 2 4 6 8 9
   S3_T29     ¦  1 2 3 4 5 7 8
   S3_T30     ¦  3 4 5 6 8 9 10
```

Fig. 1. Snapshot of Transactions of DB_1 ,DB_2 and DB_3

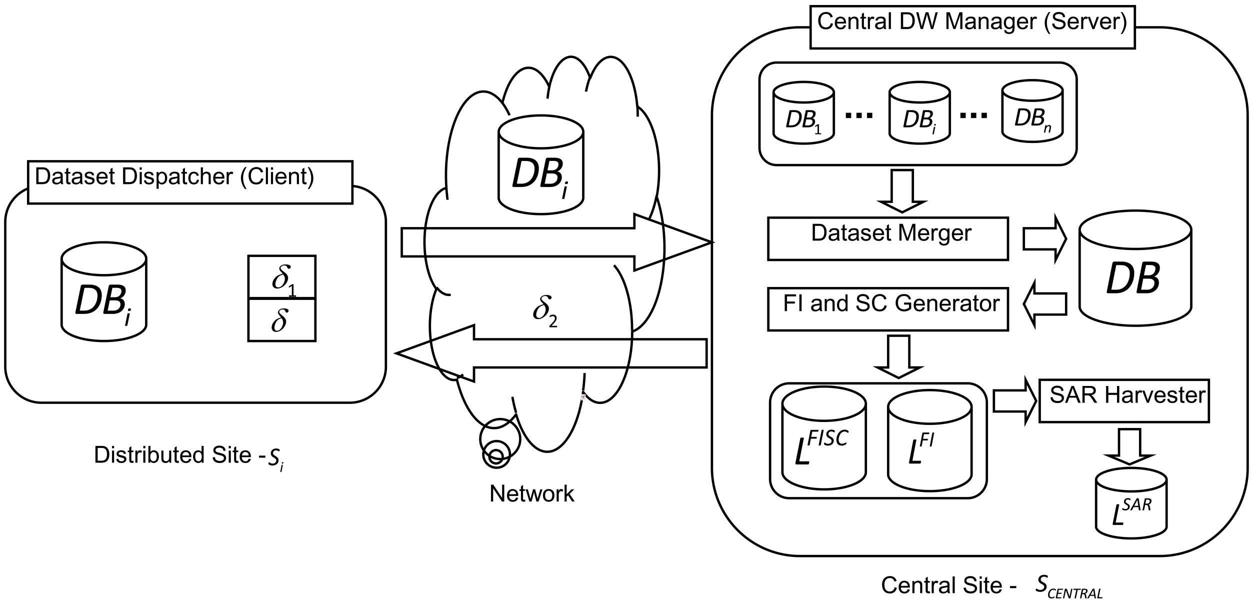


Fig. 2. Framework for Central DW based ARM

here. CPU time taken by Dataset Merger module, FIandSCGenerater modeule and SAR Harvester modeule are 203789677 nano seconds, 291567951788 nano seconds and 218382486678 nano seconds respectively. Number of strong ARs for frequent 2-itemsets, 3-itemsets and 4-itemsets are 54, 174 and 51 respectively. The result analysis is performed in Table 3.

6. CONCLUSION

The practical investigation and result analysis in this study indicates that this central DW based DM approach is ineffective because of various costs involved in managing data. This approach is also not privacy-preserving and scalable.Performance and scalability of a DM application can be increased by distributing the workload among sites. This is possible when the DM is performed locally and

Algorithm 2 CENTRALDWMANAGER

```
Input:
    —$DB = \{DB_i, i = 1 \cdots n\}, Transactional\ Data\ Set\ from\ n\ disttibuted\ sites$
    —$I = \{d_l, l = 1 \cdots m\}, total\ m\ items\ in\ DB_i$
    —$minthrsup, the\ given\ minimum\ threshold\ support$
    —$minthrconf, the\ given\ minimum\ threshold\ confidence$
Output: $L^{SAR}, the\ list\ of\ strong\ association\ rules$

 1: procedure CENTRALDWMANAGER($DB, I, minthrsup, minthrconf$)
 2:     $P \leftarrow open\ a\ new\ server\ port\ at\ \#\ 9900$
 3:     while $all\ DB_i\ from\ registered\ clients\ not\ received$ do
 4:         $\beta \leftarrow accept\ the\ incoming\ request\ at\ P\ and\ open\ a\ new\ socket\ with\ the\ client$
 5:         $start\ a\ new\ Thread\ of$ CENTRALDWMANAGERIMPL($\beta$)
 6:     end while
 7:     $DB \leftarrow Call$ DATASETMERGER($\{DB_i, i = 1 \cdots n\}$)
 8:     $L^{FI\&SC} \leftarrow Call$ FIANDSCGENERATOR($DB, I, minthrsup$)
 9:     $L^{SAR} \leftarrow Call$ SARHARVESTER($L^{FI\&SC}, minthrconf$)
10:     $Close\ all\ the\ opened\ channels$
11:     return $L^{SAR}$
12: end procedure

13: procedure CENTRALDWMANAGERIMPL($\alpha : socket\ with\ the\ client$)
14:     $\beta \leftarrow bind\ ObjectInputStream\ at\ \alpha$
15:     $\gamma \leftarrow bind\ ObjectOutputStream\ at\ \alpha$
16:     $A \leftarrow read\ the\ entire\ DB_i\ object\ at\ \beta\ stream$
17:     $\epsilon \leftarrow get\ end\ time\ for\ dataset\ receiving$
18:     $write\ \epsilon\ on\ \gamma\ stream\ to\ send\ to\ the\ client$
19:     $\xi \leftarrow bind\ ObjectOutputStream\ with\ local\ file\ system\ on\ server$
20:     $write\ A\ on\ \xi\ stream\ to\ save\ DB_i\ on\ the\ server$
21:     $Close\ all\ the\ opened\ channels$
22: end procedure
```

Table 3. Result Analysis.

Parameter	Analysis
Storage Cost(cost incurred in storing and managing the data)	Size of central DB = 565248 bytes (552KB) As all data from distributed sites are centrally colleted so storage cost is high at central location.
Communication Cost(cost incurred in terms of time taken in transferring the data over the network)	Maximum δ in transferring DB_i = 2684481005 ns. As entire DB_i is to be transferred to central location for mining so communication cost is also high particularly in case of limited network bandwidth.
Computational Cost(cost incurred in terms of CPU time taken in executing the task)	CPU time taken by FI and SC Generator module = 291567951788 ns, CPU time taken by SAR Harvester = 218382486678 ns. Total CPU time taken by two major modules in Central DW based ARM is 509950438466 nano sec as huge dataset has to be processed at central location.
Global Knowledge	Central DW based ARM does not reflect the global knowledge as it also contains the strong rules for frequent itemsets which may not be locally frequent.
Security	No security in transferring DB_i from each site over the network.
Scalability	Adding more sites would affect the performance of the system by increasing all the costs involved. Hence it is not scalable system.

Table 2. Framework configuration.

Site Name	Processor	OS	LAN Configuration	
			IP^1	Network
$S_{CENTRAL}$	$Intel^2$	MS^3	192.168.46.5	NW^4
S_1	$Intel^2$	MS^3	192.168.46.212	NW^4
S_2	$Intel^2$	MS^3	192.168.46.189	NW^4
S_3	$Intel^2$	MS^3	192.168.46.213	NW^4

1. IP address with Mask:255.255.255.0 and Gateway:192.168.46.1
2. Intel Pentium Dual Core(3.40 GHz,3.40 GHz)with 512MB RAM
3. Microsoft Windows XP Professional ver. 2002
4. Network Speed:100 Mbps and Network Adaptor: Intell 82566DM-2 Gigabit NIC

only results are carried at central site for mining global knowledge. Intelligent software agent technology and its advantages discussed in [7] makes this technology best suited for mining the distributed data. This paper is an attempt to practically investigate the classical central DW based client-server model for association rule mining and paving the way for moving ahead in further research of designing a framework for agent based distributed data mining.

7. REFERENCES

[1] R. Agrawal, T. Imielinski, and A. Swami. Mining association rules between sets of items in large databases. In *Proceedings*

Algorithm 4 FIandSCGenerater: Part-1

Input:
—$DB = \bigcup_{i=1}^{n} DB_i, Central\ Datawarehouse$
—$I = \{d_l, l = 1 \cdots m\}, total\ m\ items\ in\ DB_i$
—$minthrsup, the\ given\ minimum\ threshold\ support$
Output: $L^{FI\&SC}, the\ list\ of\ frequent\ itemsets\ their\ support\ counts$

```
procedure FIandSCGenerater($DB, Itemset, minthrsup$)
    $T \leftarrow DB.size$                                              ▷ No. of records in DB
    $I \leftarrow Itemset.size$                                         ▷ No. of items in DB
    $minsupcount \leftarrow (T \times minthrsup)/100$
    ▷ generate frequent-1 itemset list ($FIL_1$) and support count list ($FISC_1$ )
    $CFIL_1 \leftarrow \{1, 2, 3 \cdots I\}$                               ▷ candidate frequent-1 itemset
    for $i \leftarrow 1, I$ do                 ▷ initialize the support count array $SCFIL_1$ to zero
        $SCFIL_1[i] \leftarrow 0$
    end for
    $k \leftarrow 1$
    for all $candidate\ c \in CFIL_1$ do                  ▷ find support count for every candidate
        for all $transaction\ t \in DB$ do                                        ▷ scan DB
            if $c \subset t$ then
                $SCFIL_1[k] \leftarrow SCFIL_1[k] + 1$
            end if
        end for
        $k \leftarrow k + 1$
    end for
    ▷ prune $CFIL_1$ to generate $FIL_1$ and $FISC_1$
    for $k \leftarrow 1, I$ do
        if $SCFIL_1[k] \geq minsupcount$ then
            $add\ c_k \in CFIL_1\ to\ FIL_1$
            $add\ SCFIL_1[k]\ to\ FISC_1$
        end if
    end for
    if $FIL_1 \neq \emptyset$ then
        $add\ FIL_1\ to\ L^{FI}$
        $add\ FISC_1\ to\ L^{FISC}$
    end if
    $k \leftarrow 2$
    while $FIL_{k-1} \neq \emptyset$ do
        $CFIL_k \leftarrow Call$ GenerateCFIL($FIL_{k-1}$)                 ▷ candidate frequent-k itemset
        for $i \leftarrow 1, CFIL_k.length$ do              ▷ initialize the array $SCFIL_k$ to zero
            $SCFIL_k[i] \leftarrow 0$
        end for
        $i \leftarrow 1$
        for all $candidate\ c \in CFIL_k$ do              ▷ find support count for every candidate
            for all $transaction\ t \in DB$ do                                    ▷ scan DB
                if $c \subset t$ then
                    $SCFIL_k[i] \leftarrow SCFIL_k[i] + 1$
                end if
            end for
            $i \leftarrow i + 1$
        end for
        ▷ prune $CFIL_k$ to generate $FIL_k$ and $FISC_k$
        for $i \leftarrow 1, SCFIL_k.length$ do
            if $SCFIL_k[i] \geq minsupcount$ then
                $add\ c_i \in CFIL_k\ to\ FIL_k$
                $add\ SCFIL_k[i]\ to\ FISC_k$
            end if
        end for
        if $FIL_k \neq \emptyset$ then
            $add\ FIL_k\ to\ L^{FI}$
            $add\ FISC_k\ to\ L^{FISC}$
        end if
        $k \leftarrow k + 1$
    end while
    $add\ T\ to\ L^{FI\&SC}$
    $add\ L^{FI}\ to\ L^{FI\&SC}$
    $add\ L^{FISC}\ to\ L^{FI\&SC}$
    return $L^{FI\&SC}$
end procedure
```

Algorithm 5 GENERATECFIL

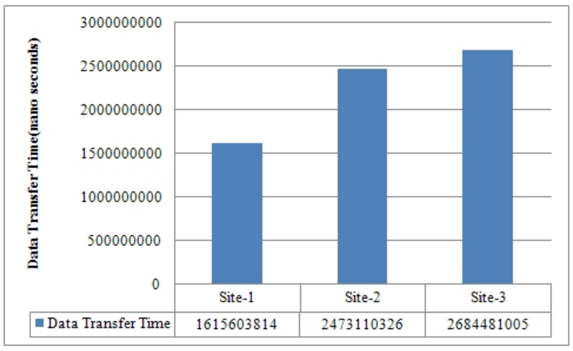
Input: $L_{k-1}, Frequent\ k-1\ itemsets$
Output: $C_k, Candidate\ Frequent\ k\ itemsets$

```
 1: procedure GENERATECFIL($L_{k-1}$)
 2:     for all $itemset\ l_1 \in L_{k-1}$ do
 3:         for all $itemset\ l_2 \in L_{k-1}$ do
 4:             if $(l_1[1] = l_2[1]) \wedge (l_1[2] = l_2[2]) \wedge \cdots \wedge (l_1[k-1] = l_2[k-1])$ then
 5:                 $c \leftarrow l_1 \otimes l_2$                      ▷ join step: generate candidates
 6:             end if
 7:             if HASINFREQUENTSUBSET($c, L_{k-1}$) then
 8:                 $delete\ c$                                ▷ prune step: remove unfruitful candidate
 9:             else
10:                 $add\ c\ to\ C_k$
11:             end if
12:         end for
13:     end for
14:     return $C_k$
15: end procedure
```

Algorithm 6 HASINFREQUENTSUBSET

Input: $c, Candidate\ k-itemset$
Output: $L_{k-1}, Frequent\ k-1\ itemsets$

```
1: procedure HASINFREQUENTSUBSET($c, L_{k-1}$)
2:     for all $(k-1)\ subset\ s \in c$ do
3:         if $s \notin L_{k-1}$ then
4:             return TRUE
5:         else
6:             return FALSE
7:         end if
8:     end for
9: end procedure
```



	Site-1	Site-2	Site-3
■ Data Transfer Time	1615603814	2473110326	2684481005

Fig. 4. Dataset Transfer Time (δ) at S_1, S_2 and S_3

of the ACM-SIGMOD International Conference of Management of Data, pages 207–216, 1993.

[2] R. Agrawal and J. C. Shafer. Parallel mining of association rules. *IEEE Transaction on Knowledge and Data Engineering*, 8(6):962–969, 1996.

[3] Rakesh Agrawal and Ramakrishnan Srikant. Fast Algorithms for Mining Association Rules in Large Databases. In *Proceedings of the 20th International Conference on Very Large Data Bases(VLDB'94)*, pages 487–499. Morgan Kaufmann Publishers Inc., Sept. 12-15 1994.

[4] Kamal Ali Albashiri, Frans Coenen, and Paul Leng. EMADS: An extendible multi-agent data miner. *Knowledge-Based Systems*, 22(7):523–528, October 2009b.

[5] Gurpreet Singh Bhamra, Ram Bahadur Patel, and Anil Kumar Verma. An Encounter with Strong Association Rules. In *Proceedings of IEEE International Advanced Computing Conference (IACC-2010)*, pages 342–346. Thapar University, Patiala, Punjab, India, IEEE, Feb 19-20 2010.

[6] Gurpreet Singh Bhamra, Ram Bahadur Patel, and Anil Kumar Verma. TDSGenerator: A Tool for generating synthetic Transactional Datasets for Association Rules Mining. *International Journal of Computer Science Issues (IJCSI)*, 8(2):184–188, March 2011.

[7] Gurpreet Singh Bhamra, Ram Bahadur Patel, and Anil Kumar Verma. Intelligent Software Agent Technology: An Overview. *International Journal of Computer Applications(IJCA)*, 89(2):19–31, March 2014.

[8] Frans Coenen, Graham Goulbourne, and Paul Leng. Tree Structures for Mining Association Rules. *Data Mining and Knowledge Discovery*, 8(1):25–51, January 2004.

[9] Usama M. Fayyad, Gregory Piatetsky-Shapiro, Padhraic Smyth, and Ramasamy Uthurusamy. *Advances in Knowledge Discovery and Data Mining*. AAAI/MIT Press, 1996.

[10] J. Han and M. Kamber. *Data Mining: Concepts and Techniques*. Morgan Kaufmann, 2nd edition, 2006.

Algorithm 7 SARHARVESTOR

Input:
—$L^{FI\&SC}, Collection\ of\ number\ of\ records\ ,\ frequent\ k\ itemset\ and\ support\ count$
—$minthrconf, the\ given\ minimum\ threshold\ confidence$
Output: $L^{SAR}, List\ of\ Strong\ Association\ Rules$

```
1: procedure SARHARVESTOR($L^{FI\&SC}, minthrconf$)
2:     $T \leftarrow L^{FI\&SC}.get(0)$                                    ▷ No. of records
3:     $L^{FI} \leftarrow L^{FI\&SC}.get(1)$                               ▷ frequent k-itemset list
4:     $L^{FISC} \leftarrow L^{FI\&SC}.get(2)$                             ▷ support count list
5:     for $k \leftarrow 2, L^{FI}.size$ do
6:         $L_k \leftarrow L^{FI}.get(k)$                                  ▷ get frequent k-itemset list
7:         for all $l \in L_k$ do
8:             $l_{subsets} \leftarrow generate\ all\ non-empty\ subsets\ of\ l$
9:             $l_{spcount} \leftarrow get\ support\ count\ of\ l\ from\ L^{FISC}$
10:            $AR_{support} \leftarrow (l_{spcount}/T) \times 100$        ▷ support of the association rule
11:            for all $non-empty\ subset\ s \in l_{subsets}$ do
12:                $s_{spcount} \leftarrow get\ support\ count\ of\ s\ from\ L^{FISC}$
13:                $AR_{conf} \leftarrow (l_{spcount}/s_{spcount}) \times 100$   ▷ confidence of the association rule
14:                if $AR_{conf} \geq minthrconf$ then
15:                    $AR_{strong} \leftarrow "s \Rightarrow l - s[AR_{support}\%, AR_{conf}\%]"$
16:                    print $AR_{strong}$
17:                    $add\ AR_{strong}\ to\ L^{SAR}$
18:                end if
19:            end for
20:        end for
21:    end for
22:    return $L^{SAR}$
23: end procedure
```

[11] Jiawei Han, Jian Pei, and Yiwen Yin. Mining frequent patterns without candidate generation. In *Proceedings of the 2000 ACM SIGMOD international conference on Management of data*, pages 1–12. ACM, 2000.

[12] Jiawei Han, Jian Pei, Yiwen Yin, and Runying Mao. Mining Frequent Patterns without Candidate Generation: A Frequent-Pattern Tree Approach. *Data Mining and Knowledge Discovery*, 8(1):53–87, January 2004.

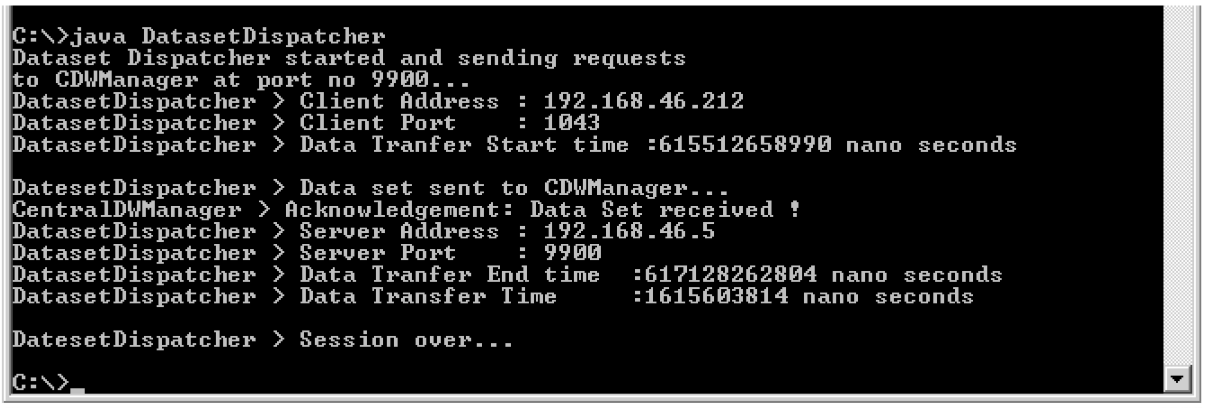
[13] Jochen Hipp, Ulrich Guntzer, and Gholamreza Nakhaeizadeh. Algorithms for association rule mining a general survey and comparison. *ACM SIGKDD Explorations Newsletter*, 2(1):58–64, June 2000.

[14] Renata Ivancsy, Ferenc Kovacs, and Istvan Vajk. An analysis of association rule mining algorithms. In *Proceedings of the 4th International ICSC Symposium on Engineering of Intelligent Systems*, pages 774–778, Feb. 29 - March 2 2004.

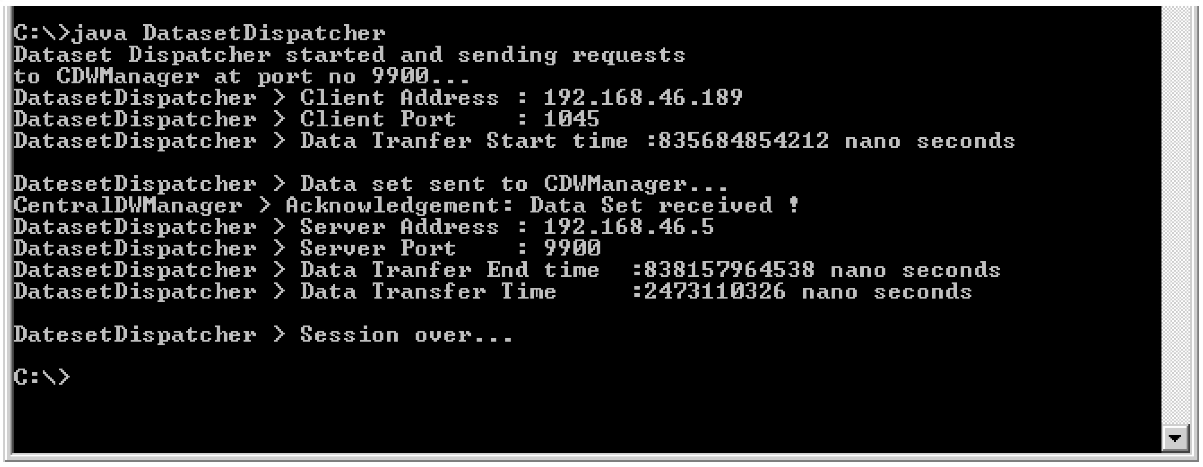
[15] Byung-Hoon Park and Hillol Kargupta. Distributed Data Mining: Algorithms, Systems, and Applications. Department of Computer Science and Electrical Engineering, University of Maryland Baltimore County, 1000 Hilltop Circle Baltimore, MD 21250, 2002.

[16] Grigorios Tsoumakas and Ioannis Vlahavas. Distributed Data Mining. Department of Informatics, Aristotle University of Thessaloniki, Thessaloniki, Greece, 2009.

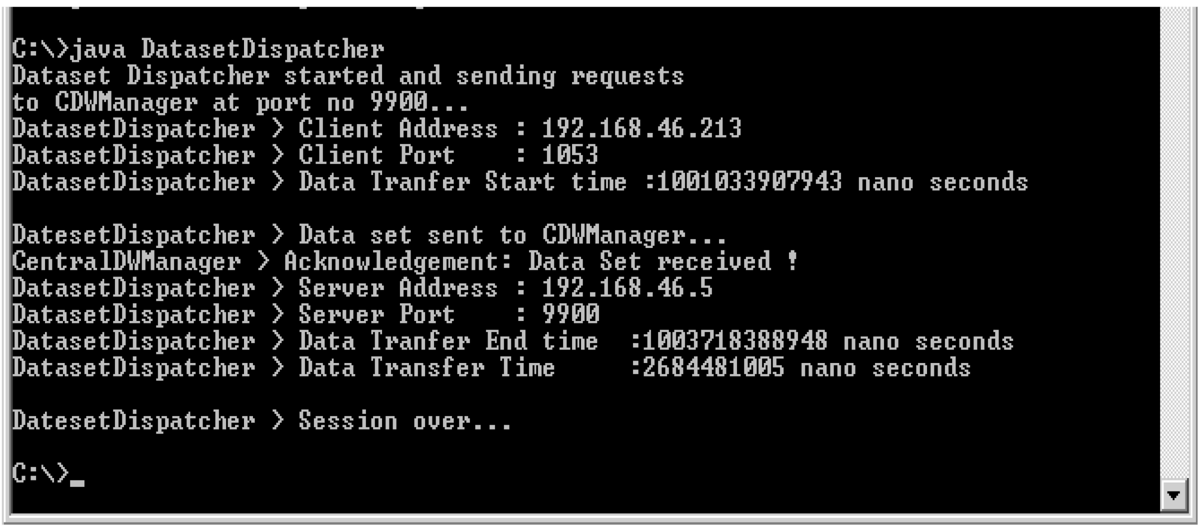
[17] M. J. Zaki. Parallel and distributed association mining: a survey. *IEEE Concurrency*, 7(4):14–25, 1999.



```
C:\>java DatasetDispatcher
Dataset Dispatcher started and sending requests
to CDWManager at port no 9900...
DatasetDispatcher > Client Address : 192.168.46.212
DatasetDispatcher > Client Port    : 1043
DatasetDispatcher > Data Tranfer Start time :615512658990 nano seconds

DatesetDispatcher > Data set sent to CDWManager...
CentralDWManager > Acknowledgement: Data Set received !
DatasetDispatcher > Server Address : 192.168.46.5
DatasetDispatcher > Server Port    : 9900
DatasetDispatcher > Data Tranfer End time   :617128262804 nano seconds
DatasetDispatcher > Data Transfer Time      :1615603814 nano seconds

DatesetDispatcher > Session over...

C:\>_
```

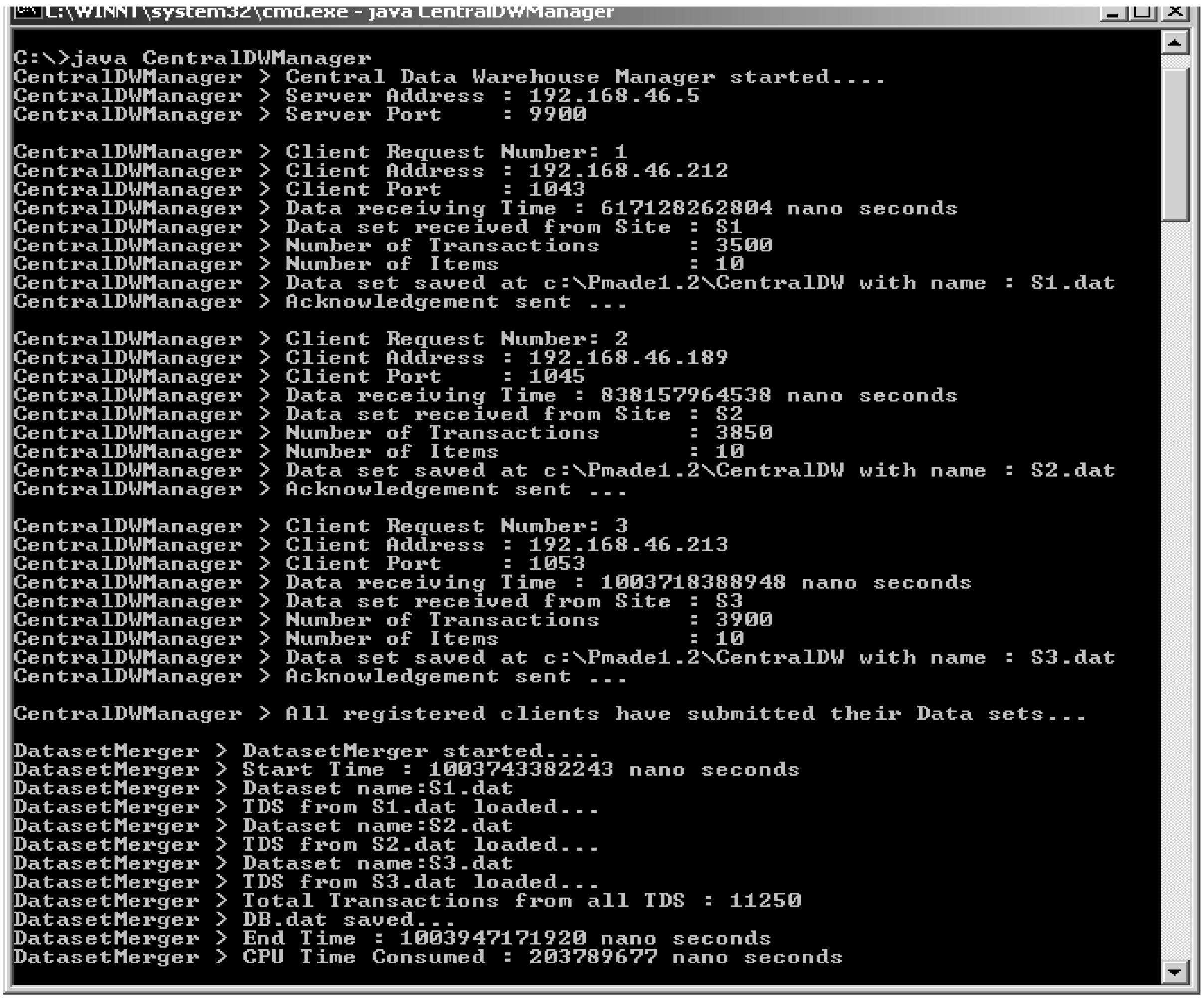


```
C:\>java DatasetDispatcher
Dataset Dispatcher started and sending requests
to CDWManager at port no 9900...
DatasetDispatcher > Client Address : 192.168.46.189
DatasetDispatcher > Client Port    : 1045
DatasetDispatcher > Data Tranfer Start time :835684854212 nano seconds

DatesetDispatcher > Data set sent to CDWManager...
CentralDWManager > Acknowledgement: Data Set received !
DatasetDispatcher > Server Address : 192.168.46.5
DatasetDispatcher > Server Port    : 9900
DatasetDispatcher > Data Tranfer End time   :838157964538 nano seconds
DatasetDispatcher > Data Transfer Time      :2473110326 nano seconds

DatesetDispatcher > Session over...

C:\>
```



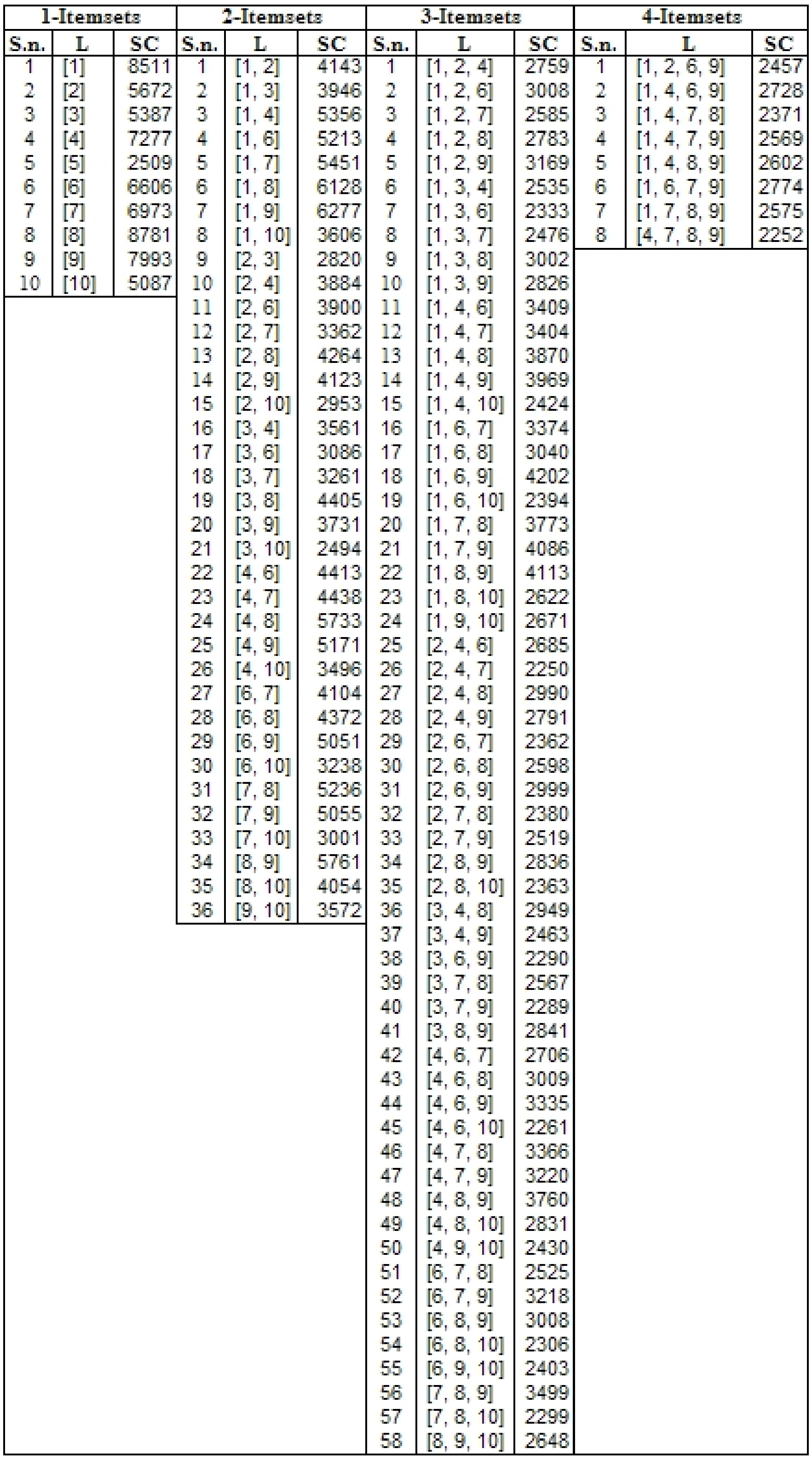
```
C:\>java DatasetDispatcher
Dataset Dispatcher started and sending requests
to CDWManager at port no 9900...
DatasetDispatcher > Client Address : 192.168.46.213
DatasetDispatcher > Client Port    : 1053
DatasetDispatcher > Data Tranfer Start time :1001033907943 nano seconds

DatesetDispatcher > Data set sent to CDWManager...
CentralDWManager > Acknowledgement: Data Set received !
DatasetDispatcher > Server Address : 192.168.46.5
DatasetDispatcher > Server Port    : 9900
DatasetDispatcher > Data Tranfer End time   :1003718388948 nano seconds
DatasetDispatcher > Data Transfer Time      :2684481005 nano seconds

DatesetDispatcher > Session over...

C:\>_
```

Fig. 3. Dataset Dispatcher running at S_1, S_2 and S_3



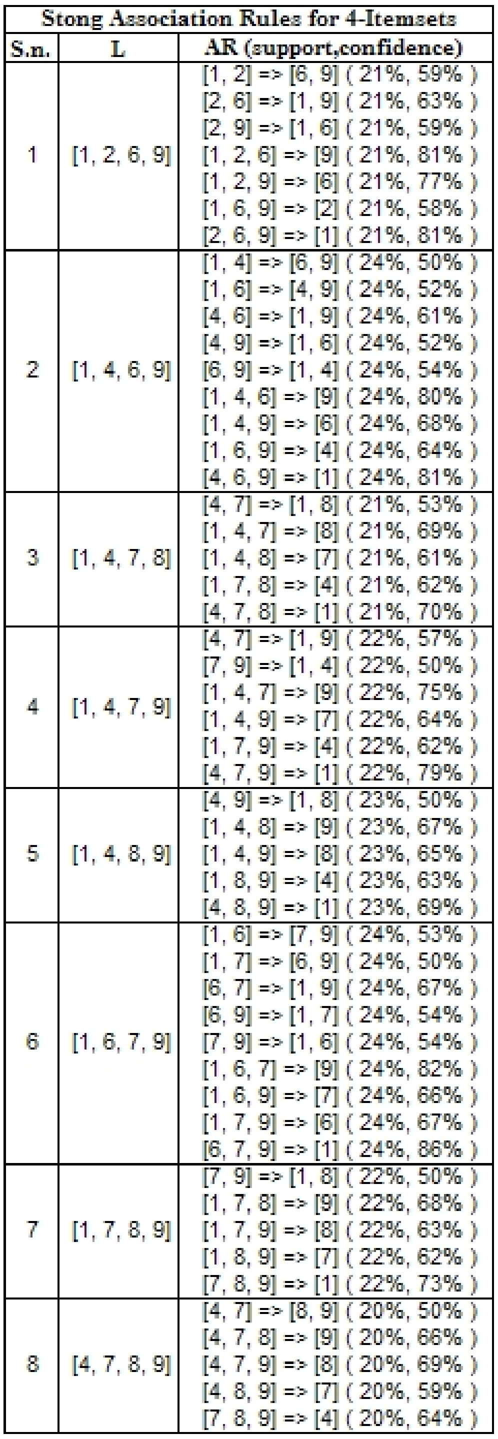
```
C:\WINNT\system32\cmd.exe - java CentralDWManager

C:\>java CentralDWManager
CentralDWManager > Central Data Warehouse Manager started....
CentralDWManager > Server Address : 192.168.46.5
CentralDWManager > Server Port    : 9900

CentralDWManager > Client Request Number: 1
CentralDWManager > Client Address : 192.168.46.212
CentralDWManager > Client Port    : 1043
CentralDWManager > Data receiving Time : 617128262804 nano seconds
CentralDWManager > Data set received from Site : S1
CentralDWManager > Number of Transactions      : 3500
CentralDWManager > Number of Items             : 10
CentralDWManager > Data set saved at c:\Pmade1.2\CentralDW with name : S1.dat
CentralDWManager > Acknowledgement sent ...

CentralDWManager > Client Request Number: 2
CentralDWManager > Client Address : 192.168.46.189
CentralDWManager > Client Port    : 1045
CentralDWManager > Data receiving Time : 838157964538 nano seconds
CentralDWManager > Data set received from Site : S2
CentralDWManager > Number of Transactions      : 3850
CentralDWManager > Number of Items             : 10
CentralDWManager > Data set saved at c:\Pmade1.2\CentralDW with name : S2.dat
CentralDWManager > Acknowledgement sent ...

CentralDWManager > Client Request Number: 3
CentralDWManager > Client Address : 192.168.46.213
CentralDWManager > Client Port    : 1053
CentralDWManager > Data receiving Time : 1003718388948 nano seconds
CentralDWManager > Data set received from Site : S3
CentralDWManager > Number of Transactions      : 3900
CentralDWManager > Number of Items             : 10
CentralDWManager > Data set saved at c:\Pmade1.2\CentralDW with name : S3.dat
CentralDWManager > Acknowledgement sent ...

CentralDWManager > All registered clients have submitted their Data sets...

DatasetMerger > DatasetMerger started....
DatasetMerger > Start Time : 1003743382243 nano seconds
DatasetMerger > Dataset name:S1.dat
DatasetMerger > TDS from S1.dat loaded...
DatasetMerger > Dataset name:S2.dat
DatasetMerger > TDS from S2.dat loaded...
DatasetMerger > Dataset name:S3.dat
DatasetMerger > TDS from S3.dat loaded...
DatasetMerger > Total Transactions from all TDS : 11250
DatasetMerger > DB.dat saved...
DatasetMerger > End Time : 1003947171920 nano seconds
DatasetMerger > CPU Time Consumed : 203789677 nano seconds
```

Fig. 5. Central DW Manager running at $S_{CENTRAL}$



1-Itemsets			2-Itemsets			3-Itemsets			4-Itemsets		
S.n.	L	SC	S.n.	L	SC	S.n.	L	SC	S.n.	L	SC
1	[1]	8511	1	[1, 2]	4143	1	[1, 2, 4]	2759	1	[1, 2, 6, 9]	2457
2	[2]	5672	2	[1, 3]	3946	2	[1, 2, 6]	3008	2	[1, 4, 6, 9]	2728
3	[3]	5387	3	[1, 4]	5356	3	[1, 2, 7]	2585	3	[1, 4, 7, 8]	2371
4	[4]	7277	4	[1, 6]	5213	4	[1, 2, 8]	2783	4	[1, 4, 7, 9]	2569
5	[5]	2509	5	[1, 7]	5451	5	[1, 2, 9]	3169	5	[1, 4, 8, 9]	2602
6	[6]	6606	6	[1, 8]	6128	6	[1, 3, 4]	2535	6	[1, 6, 7, 9]	2774
7	[7]	6973	7	[1, 9]	6277	7	[1, 3, 6]	2333	7	[1, 7, 8, 9]	2575
8	[8]	8781	8	[1, 10]	3606	8	[1, 3, 7]	2476	8	[4, 7, 8, 9]	2252
9	[9]	7993	9	[2, 3]	2820	9	[1, 3, 8]	3002			
10	[10]	5087	10	[2, 4]	3884	10	[1, 3, 9]	2826			
			11	[2, 6]	3900	11	[1, 4, 6]	3409			
			12	[2, 7]	3362	12	[1, 4, 7]	3404			
			13	[2, 8]	4264	13	[1, 4, 8]	3870			
			14	[2, 9]	4123	14	[1, 4, 9]	3969			
			15	[2, 10]	2953	15	[1, 4, 10]	2424			
			16	[3, 4]	3561	16	[1, 6, 7]	3374			
			17	[3, 6]	3086	17	[1, 6, 8]	3040			
			18	[3, 7]	3261	18	[1, 6, 9]	4202			
			19	[3, 8]	4405	19	[1, 6, 10]	2394			
			20	[3, 9]	3731	20	[1, 7, 8]	3773			
			21	[3, 10]	2494	21	[1, 7, 9]	4086			
			22	[4, 6]	4413	22	[1, 8, 9]	4113			
			23	[4, 7]	4438	23	[1, 8, 10]	2622			
			24	[4, 8]	5733	24	[1, 9, 10]	2671			
			25	[4, 9]	5171	25	[2, 4, 6]	2685			
			26	[4, 10]	3496	26	[2, 4, 7]	2250			
			27	[6, 7]	4104	27	[2, 4, 8]	2990			
			28	[6, 8]	4372	28	[2, 4, 9]	2791			
			29	[6, 9]	5051	29	[2, 6, 7]	2362			
			30	[6, 10]	3238	30	[2, 6, 8]	2598			
			31	[7, 8]	5236	31	[2, 6, 9]	2999			
			32	[7, 9]	5055	32	[2, 7, 8]	2380			
			33	[7, 10]	3001	33	[2, 7, 9]	2519			
			34	[8, 9]	5761	34	[2, 8, 9]	2836			
			35	[8, 10]	4054	35	[2, 8, 10]	2363			
			36	[9, 10]	3572	36	[3, 4, 8]	2949			
						37	[3, 4, 9]	2463			
						38	[3, 6, 9]	2290			
						39	[3, 7, 8]	2567			
						40	[3, 7, 9]	2289			
						41	[3, 8, 9]	2841			
						42	[4, 6, 7]	2706			
						43	[4, 6, 8]	3009			
						44	[4, 6, 9]	3335			
						45	[4, 6, 10]	2261			
						46	[4, 7, 8]	3366			
						47	[4, 7, 9]	3220			
						48	[4, 8, 9]	3760			
						49	[4, 8, 10]	2831			
						50	[4, 9, 10]	2430			
						51	[6, 7, 8]	2525			
						52	[6, 7, 9]	3218			
						53	[6, 8, 9]	3008			
						54	[6, 8, 10]	2306			
						55	[6, 9, 10]	2403			
						56	[7, 8, 9]	3499			
						57	[7, 8, 10]	2299			
						58	[8, 9, 10]	2648			

Fig. 6. Lists of Frequent Itemsets and their support count with 20% min_th_sup at $S_{CENTRAL}$



Stong Association Rules for 4-Itemsets		
S.n.	L	AR (support,confidence)
1	[1, 2, 6, 9]	[1, 2] => [6, 9] (21%, 59%)
		[2, 6] => [1, 9] (21%, 63%)
		[2, 9] => [1, 6] (21%, 59%)
		[1, 2, 6] => [9] (21%, 81%)
		[1, 2, 9] => [6] (21%, 77%)
		[1, 6, 9] => [2] (21%, 58%)
		[2, 6, 9] => [1] (21%, 81%)
2	[1, 4, 6, 9]	[1, 4] => [6, 9] (24%, 50%)
		[1, 6] => [4, 9] (24%, 52%)
		[4, 6] => [1, 9] (24%, 61%)
		[4, 9] => [1, 6] (24%, 52%)
		[6, 9] => [1, 4] (24%, 54%)
		[1, 4, 6] => [9] (24%, 80%)
		[1, 4, 9] => [6] (24%, 68%)
		[1, 6, 9] => [4] (24%, 64%)
		[4, 6, 9] => [1] (24%, 81%)
3	[1, 4, 7, 8]	[4, 7] => [1, 8] (21%, 53%)
		[1, 4, 7] => [8] (21%, 69%)
		[1, 4, 8] => [7] (21%, 61%)
		[1, 7, 8] => [4] (21%, 62%)
		[4, 7, 8] => [1] (21%, 70%)
4	[1, 4, 7, 9]	[4, 7] => [1, 9] (22%, 57%)
		[7, 9] => [1, 4] (22%, 50%)
		[1, 4, 7] => [9] (22%, 75%)
		[1, 4, 9] => [7] (22%, 64%)
		[1, 7, 9] => [4] (22%, 62%)
		[4, 7, 9] => [1] (22%, 79%)
5	[1, 4, 8, 9]	[4, 9] => [1, 8] (23%, 50%)
		[1, 4, 8] => [9] (23%, 67%)
		[1, 4, 9] => [8] (23%, 65%)
		[1, 8, 9] => [4] (23%, 63%)
		[4, 8, 9] => [1] (23%, 69%)
6	[1, 6, 7, 9]	[1, 6] => [7, 9] (24%, 53%)
		[1, 7] => [6, 9] (24%, 50%)
		[6, 7] => [1, 9] (24%, 67%)
		[6, 9] => [1, 7] (24%, 54%)
		[7, 9] => [1, 6] (24%, 54%)
		[1, 6, 7] => [9] (24%, 82%)
		[1, 6, 9] => [7] (24%, 66%)
		[1, 7, 9] => [6] (24%, 67%)
		[6, 7, 9] => [1] (24%, 86%)
7	[1, 7, 8, 9]	[7, 9] => [1, 8] (22%, 50%)
		[1, 7, 8] => [9] (22%, 68%)
		[1, 7, 9] => [8] (22%, 63%)
		[1, 8, 9] => [7] (22%, 62%)
		[7, 8, 9] => [1] (22%, 73%)
8	[4, 7, 8, 9]	[4, 7] => [8, 9] (20%, 50%)
		[4, 7, 8] => [9] (20%, 66%)
		[4, 7, 9] => [8] (20%, 69%)
		[4, 8, 9] => [7] (20%, 59%)
		[7, 8, 9] => [4] (20%, 64%)

Fig. 7. Strong Association Rules for Frequent 4-Itemsets at $S_{CENTRAL}$