Optimization of Distributed Association Rule Mining Approach Based On Vertical Partitioning

MONIKA
Banasthali University
Rajasthan, India
Rao_031@yahoo.co.in

Dr. Harish Rohil
Department of Computer Science and Applications
Chaudhary DeviLal University (CDLU)
Sirsa,Haryana

Abstract— Association rule mining is a one of the most important technique in data mining. It extracts significant patterns from transaction databases and generates rules used in many decision support application. Modern organizations are geographically distributed. Using the traditional centralized association rule mining to discover useful patterns in such distributed system is not always feasible because merging data sets from different sites into a centralized site incurs huge network communication and time costs. This paper present an optimized Distributed Association Rule Mining (D-ARM) based on vertical partitioning. The existing D-ARM algorithms have lots of communication overhead, which is a major issue for concerning. The proposed approach minimizes this communication overhead and it is based on total count. The papers then discuss the Total Count on Vertical Dataset (TCDV) use of this structure which offers significant advantages with respect to existing D-ARM techniques.

Keywords- Data Mining, Distributed Association Rule Mining, Vertical Partitioning

I. INTRODUCTION

Data mining technology aim to find useful patterns from large amount of data. Data mining is the process of analyzing data from different angles & getting useful information about data. The data mining can help in predicting a trend or values, classifying, categorizing the data & in finding correlations, patterns from the dataset. The overall goal of data mining process is to extract information from a dataset & transform it into an understandable structure for future use.

Consider I = {i1… in} be a set of items. Let D be a set of transactions or database. Each transaction $t \in D$ is an item set such that t is a proper subset of I. A transaction t supports X, a set of items in I, if X is a proper subset of t. An association rule is an implication of the form X→Y, where X and Y are subsets of I and $X \cap Y = \emptyset$. The support of rule X→Y can be computed by the following equation: Support (X→Y) = |X→Y| / |D|, where |X→Y| denotes the number of transactions in the database that contains the itemset XY, and |D| denotes the number of the transactions in the database D. The confidence of rule is calculated by following equation: Confidence (X→Y) = |X→Y|/|X|, where |X| is number of transactions in database D that contains item set X. A rule XY is strong if support(X→Y) ≥ min_support and confidence(X→Y) ≥ min_confidence, where min_support and min_confidence are two given minimum thresholds [1].

Association rule mining algorithms scan the database of transactions and calculate the support and confidence of the rules and retrieve only those rules having support and confidence higher than the user specified minimum support and confidence threshold [1].

Association rule mining consists of two stages:

1. The discovery of frequent itemsets.
2. The generation of association rules.

It follows, that in the vast majority of cases, the discovery of the frequent set dominates the performance of the whole process. Therefore, we explicitly focus the paper on the discovery of such set [2].

Need for development of Distributed system for mining of association rules because of its unique properties:

1. Databases or data warehouses may store a huge amount of data. Mining association rules in such databases may require substantial processing power, and distributed system is a possible solution.
2. Many large databases are distributed in nature. For example, the huge numbers of transaction records of hundreds of Sears's department stores are likely to be stored at different sites.

This observation motivates authors to study efficient distributed algorithms for mining association rules in databases.

This study may also shed new light on parallel data mining. Furthermore, a distributed mining algorithm can also be used to mine association rules in a single large database by partitioning the database among a set of sites and processing the task in a distributed manner. The high flexibility, scalability, low cost performance ratio, and easy connectivity of a distributed system make it an ideal platform for mining association rules [3].

Two types of database layouts are employed in association rules mining: horizontal and vertical. In the traditional horizontal database layout, each transaction consists of a set of items and the database contains a set of transactions. Most Apriori-like algorithms use this type of layout. For vertical database layout, each item maintains a set of transaction ids (denoted by tidset) where this item is contained. Eclat uses vertical data layout. It has been shown that vertical layout performs generally better than the horizontal format. Table 1 & Table 2 show examples for different types of layouts [4].

Table: 1
Horizontal Layout

Transaction Id	Items
1	2,1,5,3
2	2,3
3	1,4
4	3,1,5
5	2,1,3
6	2,4

Table: 2
Vertical Layout

Item	Transaction Id Set
1	1,3,4,5
2	1,2,5,6
3	1,2,4,5
4	3
5	1,4

II. Related Work

Finding of interesting association rules in databases may disclose some useful patterns for decision support, selective marketing, financial forecast, medical diagnosis, and many other applications, it has attracted a lot of attention in recent data mining research. Mining association rules may require iterative scanning of large transaction or relational databases which is quite costly in processing. Therefore, efficient mining of association rules in transaction or relational databases has been studied substantially [3].

Since its introduction in 1993, the Association Rule Mining (ARM) has been studied intensively. Many algorithms, representing several different approaches, were suggested. Some algorithms, such as Apriori[5], DHP [14], FP growth[15] are bottom up & others, like pincer-search use a hybrid approach, trying to guess large itemsets at an early stage. Algorithms for D-ARM problem usually can be seen as parallelization of sequential ARM algorithm. The CD, FDM & DDM algorithms parallelize Apriori & PDM [13] parallelize DHP [14].

Two Basic parallel algorithms, Count Distribution (CD) and Data Distribution (DD) were proposed in [4]. The CD algorithm scales linearly and has excellent speedup and sizeup behavior with respect to number of transactions. Hence, the CD algorithm, like its sequential counterpart Apriori, is unscalable with respect to the increasing size of candidate set. The DD algorithm addresses the memory problem of the CD algorithm by partitioning the candidate set assigning a partition to each processor in the system [10].

FDM [3] was the further improvement on the CD algorithm. It gives better performance as compare to CD algorithm. In FDM the number of candidate sets generated can be substantially reduced to about 10-25% of that generated in CD [3].

In most of the above algorithms, the database is divided horizontally, called segmentation between nodes. There are also many algorithms that use vertical database.

Apriori [5] based & inspired algorithms are good with sparse datasets, where frequent patterns are very short. For dense datasets such as telecommunications and census data, which have many, long frequent patterns, the performance of these algorithms degrades incredibly. TO overcome these problems, a number of vertical mining algorithms has been proposed. I.e. Eclat, Dclat.

Eclat [8] algorithm is better than previous algorithms, but it still need a lot of communication. Dclat [7] is an improvement on Eclat, that uses concept of Diffset for generating frequent-itemset.

There are also many D-ARM algorithms that follow the structure of tree. The FP-growth algorithm is a new generation of frequent pattern mining that uses a compressed FP tree structure for mining a complete set of frequent itemsets without candidate itemsets generation. It works well if size of FP-tree is typically smaller and if all items are ordered from highest to lowest support count. However, for very large DB, a lot of time is required to first sort the support of 1-itemsets.

To avoid this overhead, the frequent item tree FI-growth also was proposed. This algorithm constructs an FI-tree represented by ordering the items by sequence in transactions.

III. Proposed Work

Our proposed algorithm is based on a central P-tree structure. In this method, a single pass of database is done to perform a partial summation of the support counts. These partial counts are stored in a tree structure that we call the P-tree which enumerates item sets counted in lexicographic order. The P-Tree contains all the sets of items present as distinct records in the database. Plus some additional sets that are leading subsets of these.

The Distributed version of P-Tree, PP-Tree was also proposed. It was based on vertical partitioning of item sets. This method divides the ordered set of items into subsequences & then for each subsequence it defines a PP-Tree. The **drawback** of existing approach is that the later trees in the sequences are of increasing size. For construction of complete tree it read the PP trees from all sites into a single site, and applies the Apriori-TFP algorithm to build a *T*-tree that finds the final frequent sets in the partition. As a consequence there is a lot of communication overhead because N numbers of PP-trees are passed after each pass. We try to generate the frequent item sets by expanding central P-Tree using vertical partitioning of datasets and item sets and having less communication.

This paper presents a approach which have greater efficiency in terms of communication overhead. This approach is based on (vertical) partitioning and has different way to partition the database.

A. Total Count on Vertical Dataset (TCVD) Approach

Authors have assumed that the items in a transaction or in an item set are sorted in lexicographic order. Firstly given Horizontal database as in Table 3 are converted in vertical data layout seen in Table 4. Then we allocate the distinct item sets with their Tid set to distinct nodes. During partitioning of database we also calculate L_1 and send it to each node. Now each node calculate only those candidate item sets C_2 that's according to item set assigned to that nodes.

```
Input: Database D
Output: $L_k$  //K=1to n
1) Take set of transaction and convert it into vertical layout.
2) Assign one or more item and their corresponding tid set to distinct nodes.
3) Now at each node we calculate the candidate item set $C_k$ from $L_1$.
Generate only those candidates set that start with item assign to that particular node.
                    $C_k = L_{k-1} \cup L_{k-1}$
4) We now calculate the frequent K-item set at individual nodes from their corresponding $C_k$.
5) The steps 2 & 3 are repeated until $C_k$ is empty.
// End of Algorithm for Total Count on Vertical Dataset
```

Figure 1: Algorithm for Total Count on Vertical Dataset

Example: We are given a horizontal dataset in Table 3. Each transaction t has item set in lexicographic order. These items have distinct values in real world.

Table 3: Horizontal Form of Dataset

Tid	1	2	3	4	5	6	7	8	9
Item set	abcde	Abce	abde	abe	acde	ade	ace	b	bcde
Tid	10	11	12	13	14	15	16	17	18
Itemset	bce	bd	bde	be	cd	cde	ce	d	de

Table 4: Vertical Form of Dataset

Item	Tidset
a	1,2,3,4,5,6,7
b	1,2,3,4,8,9,10,11,12,13
c	1,2,5,7,9,10,14,15,16
d	1,3,5,6,9,11,12,14,15,17,18
e	1,2,3,5,6,7,9,12,13,15,16,18

We can assign one or more items to a single node if their Tidset is around to similar, for example we assign item d & e to a single node because their Tidset is similar. Item a is assign to node 1,item b is assign to node 2,item c is assign to node 3 and item d,e are to assign to node 4. Here min-sup=2 at all nodes.

At node 1:

L_1= {a,b,c,d,e}

C_2= { ab,ac,ad,ae}

We calculate support of each itemset in C_2,and discard those itemsets whose support count is less than min-sup.

Here all items of C_2 have support more then min-sup.

Support: ab-4, ac-4, ad-4, ae-6

L_2= {ab,ac,ad,ae}

C_3= {abc,abd,abe,acd,ace,ade}

Support: abc-2, abd-2, abe-2, acd-2, ace-4,

ade-4

L_3= {abc,abd,abe,acd,ace,ade}

C_4= {abcd,abce,abde,acde}

Support: abcd-1, abce-2, abde-2, acde-2

L_4= {abce,abde,acde}

C_5= {abcde}

Support: abcde-1

So **L_5= {}.**

Now, no more candidate sets can be generated.

At node 2:

L_1= {a,b,c,d,e}

C_2= {bc,bd,be}

Support: bc-4, bd-5, be-7

L_2= {bc,bd,be}

C_3= {bcd,bce,bde}

Support:bcd-2, bce-4,bde-4

L_3= {bcd,bce,bde}

C_4= {bcde}

Support: bcde-2

L_4= {bcde}

Now, no more candidate itemsets can be generated.

Figure 2: Computation of Itemsets

At node 3 and 4 same calculations are performed as shown in figure 2.

After calculating local itemsets at each node we can exchange with each other if required. So, here communication overhead is minimum or zero. Our main aim is to mining the overhead of communication. But in this approach, most of the transactions are redundant on every node. Sometime in worst case, size of database at an node can be same as central otherwise this algorithm gives good results. This approach is based on total support count of items.

IV. Experimental Evaluation

A simulator with GUI was designed & developed with Microsoft Visual Basis in C# language. Simulator accepts text files as input and experiment is performed on test.txt file which contains 18 transactions and each transaction contains different numbers and combination of items.

Assessment of proposed approach was made on the basis of:

1. Support
2. Confidence

The **support** is defined as the percentages of transactions containing both X and Y in D.

Support (X=>Y) = P (X∪Y)

The **confidence** is defined as the percentage of transactions containing X that also contain Y in D.

Confidence(X=>Y) = P(Y/X) = Support (X∪Y)/Support(X)

Two experiments were performed with different values of support i.e. 2, 4. For the experiments, 30 transactions were taken and specification is shown in table 5.

Table 5: Specification of Dataset

Specification	**Dataset**
No of Records	30
No of Items	10

Table 6: Summary of Result

Support threshold value	**Total Item to be Selected**
1	307
2	171
3	88
4	57
5	37
6	24

For threshold value 2, number of items to be selected are 171 whereas, for threshold value 5, only 37 items are selected as shown in Table 6. This concludes that as the value of threshold is increases, number of selected item sets to be decreased. Different organization can mine their database at different values of support threshold for marketing decisions.

To test the efficiency of TCVD approach, authors conducted experiment comparing authors' approach with Tree based partitioning approach. The experiment is implemented Microsoft Visual Basic 4.0 using C# and run on a 2.10 GHz machine with a relatively RAM of 2GB based on an important factor i.e. Communication Overhead.

Algorithm for total count on vertical dataset is better than the existing approach due to following reasons:

1. In existing approach, there is a need to generate local PP tree for counting of support of each item, which is complex process to generate and have more memory requirement. In proposed approach we can directly calculate support from database.
2. In existing approach for each partition, read the PP trees form all segments into memory, and apply the Apriori-TFP algorithm to build a *T*-tree that finds the final frequent sets in the partition. This stage requires the PP trees for each segment of data to be read once only. Here for each pass to be completed, passing of tree is performed which creates a lots of communication overhead in network.

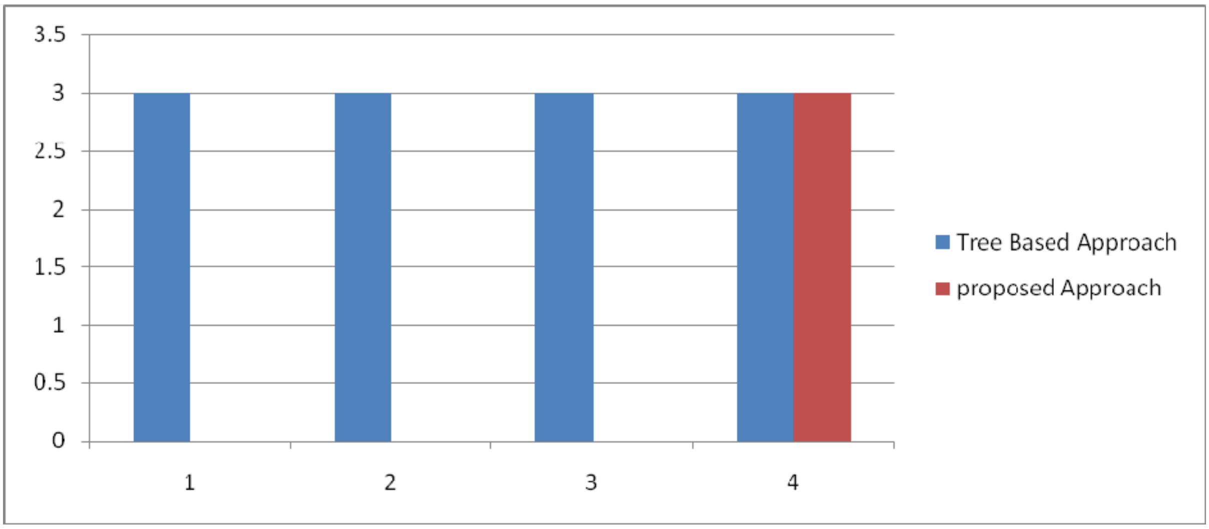

Figure 3: Communication Overhead of Tree Based Partitioning and Proposed Approach

Figure 3 clearly indicates that the proposed TCVD algorithm outperforms in terms of communication overhead as compare to tree based partitioning algorithm. In existing approach, after each phase, all nodes communicate to transfers PP Tree, where in proposed approach, nodes communicate with each other only after final phase.

V. CONCLUSION

The aim of this research is to achieve efficient methods for association rules mining in distributed environment which have less communication overhead in comparison with the previous algorithms. Most of D-ARM algorithms aim is to minimize communication overhead which is a major issue in distributed system. The approach proposed in this paper use a different method for partitioning of dataset which minimize communication overhead. Experimental result shows the efficiency of proposed approach as compared with existing approach. The mining results can provide security for customers' transaction behavior and also provide a reference for the formulation of marketing strategy. The work presented in the research can be extended for multi-level and multi-dimensional association rules.

REFERENCES

[1] Komal Shah, Amit Thakkar, Amit Ganatra "A Study on Association Rule Hiding Approaches", International Journal of Engineering and Advanced Technology (IJEAT) ISSN: 2249 – 8958, Volume-1, Issue-3, February 2012

[2] Dao-I Lin, Zvi M Kedem "Pincer Search: A New Algorithm for Discovering the Maximum Frequent Item Set", New York University, Sep 11, 1997.

[3] Cheung DWL, Han J, Ng VT, Fu AW, Fu Y. "A Fast Distributed Algorithm for Mining Association Rules", International Conference on Parallel and Distributed Systems Proceedings, pages 31-42, 1996.

[4] Mingjun Song, Sanguthevar Rajasekaran "A Transaction Mapping Algorithm for Frequent Item Sets Mining", IEEE Transactions on Knowledge and Data Engineering, Vol. 18, No. 4, Pages 472-481, April 2006.

[5] "Department of Computer Science", University of Liverpool Liverpool L69 3BX, UK

[6] C. Agarwal Ramesh, C. Aggarwal Charu, V.V.V. Prasad. "A Tree Projection Algorithm for Generation of Frequent Itemsets", IBM T. J. Watson Research Center, Yorktown Heights, NY 10598.

[7] Mohammed J. Zaki and Karam Gouda "Fast Vertical Mining Using Diffsets", SIGKDD '03, August,P ages 24-27, 2003, Washington, DC, USA.

[8] M. J. Zaki "Scalable Algorithms for Association Mining", IEEE Transactions on Knowledge and Data Engineering, Pages 372-390, May-June 2000.

[9] Frans Coenen,Paul Leng and Shakil Ahmed. "T-Trees,Vertical Partitioning and Distributed Association Rule Mining". Third IEEE International Conference on Data Mining (ICDM'03), April,2003.

[10] Eui-Hong (Sam) Han, George Karypis and Vipin Kumar. "Scalable Parallel Data Mining for Association Rules", Cray Research Inc., and NSF grant CDA, University of Minnesota, Minneapolis, USA, July 15, 1997

[11] Shakil Ahmed, Frans Coenen and Paul Leng "Tree-based Partitioning of Data for Association Rule Mining", Department of Computer Science, The University of Liverpool, Uk.

[12] Agrawal R., Mannila H., Srikant R., Toivonen H., Verkamo A. I. "Fast Discovery of Association Rules", Advances in Knowledge Discovery and Data Mining, pages 307-328, Proceedings of the 20th International Conference on Very Large Data Bases, Pages 478-499, 1994.

[13] "Efficient parallel data mining for association rules" In Proc. of ACM Conference on Information and Knowledge Management, Baltimore, MD, pages 31– 36, November 1995.

[14] J. S. Park, M.S. Chen and P. S. Yu "An Effective Hash-based Algorithm for Mining Association Rules", ACM-SIGMOD International Conference Management of Data , Pages 175-186, San Jose, CA, May 1995.

[15] Dehao Chen, Chunrong Lai, Wei Hu, WenGuang Chen, Yimin Zhang and Weimin Zhen "Tree Partition Based Parallel Frequent Pattern Mining on Shared Memory Systems", IEEE,2006.

[16] Bundit Manaskasemsak, Nunnapus Benjamas, Arnon Rungsawang, Athasit Surarerks Putchong Uthayopas, "Parallel Association Rule Mining based on FI-Growth Algorithm", 978-1-4244-1890-9/07 ©2007 IEEE