Available Online at www.ijcsmc.com

International Journal of Computer Science and Mobile Computing

A Monthly Journal of Computer Science and Information Technology

ISSN 2320–088X

IJCSMC, Vol. 3, Issue. 5, May 2014, pg.1254 – 1261

RESEARCH ARTICLE

Optimization of Distributed Association Rule Mining Based Partial Vertical Partitioning

MONIKA

monika.choudhary56@gmail.com

Abstract: ***Association rule mining is a one of the most important technique in data mining. Data mining is the process of analyzing data from different angles & getting useful information about data. 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 partial count. The papers then discuss the Partial Count on Vertical Dataset (TCDV) use of this structure which offers significant advantages with respect to existing DARM techniques.***

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

---

I. Introduction

Data mining (sometimes called data or knowledge discovery) is the process of analyzing data from different perspectives and summarizing it into useful information - information that can be used to increase revenue, cuts costs, or both. Data mining software is one of a number of analytical tools for analyzing data. It allows users to analyze data from many different dimensions or angles, categorize it, and summarize the relationships identified. Technically, data mining is the process of finding correlations or patterns among dozens of fields in large relational databases.

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∈D is an item set such that t is a proper subset of I. A transaction t supports A, a set of items in I, if A is a proper subset of t. An association rule is an implication of the form A→B, where A and B are subsets of I and A∩B= Ø.

The support of rule A→B can be computed by the following equation:
Support (A→B) = |A→B| / |D|, where |A→B| denotes the number of transactions in the database that contains the itemset AB, and |D| denotes the number of the transactions in the database D.

The confidence of rule is calculated by following equation:
Confidence (A→B) = |A→B|/|A|, where |A| is number of transactions in database D that contains item set A.

Rule AB is strong if support(A→B) ≥ min_support and confidence(A→B) ≥ 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 [2].

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 [3].
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 [4].

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 [5].

Table: 1
Horizontal Layout

tid	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	tidset
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 [4]. 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[16], DHP [15], FP growth[6] 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 [14] parallelize DHP [15]. Two Basic parallel algorithms, Count Distribution (CD) and Data Distribution (DD) were proposed in [5]. 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 [11].

FDM [4] 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 [4].

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 [16] 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 [9] algorithm is better than previous algorithms, but it still need a lot of communication. Dclat [8] 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.

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.

Partial Count Vertical Database (PCVD) proposed removes the problem of redundancy of transactions sometimes exists in Total count on Vertical Database (TCVD)[1] by using different way to partition the database. In this algorithm transaction are distributed according to the 1st item of transactions. This approach based on partial support count but gives the approximate result as TCVD approach with minimum overhead of communication. Algorithm for Partial Count shown in fig 1.

Input: Database D

Output: L_k //K=1to n

1) Convert a given dataset into vertical dataset by allocating the transaction according to 1st item of each transaction.

2) Distribute these items with their 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.

// End of Algorithm for Partial Count on Vertical Dataset

Fig.1: Algorithm for Partial Count

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 4 shows the Vertical data set according to algorithm for partial count.

Table 3: Horizontal Form of Dataset

Tid	1	2	3	4	5	6	7	8	9
Itemset	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	8,9,10,11,12,13
C	14,15,16
D	17,18

Here we can see that no transaction is repeated. Now, item a is assign to node 1, item b is assign to node 2, item c is assign to node 3, item d is assign to node.

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. **L_2= {ac,ae}** C_3= {ace} **L_3= {}** Now, no more candidate sets can be generated.	**At node 2:** **L_1= {a, b, c, d, e}** C_2= {bc,bd,be} **L_2= {bd}** **At Node 3** **L_1= {a,b,c,d,e}** C_2= {cd,ce} **L_2= {ce}** **At node 4** **L_1= {a,b,c,d,e}** C_2= {de} **L_2= {}.**

All items in L_k are frequent items. After calculation of all size frequent items, sites communicate with each other so that final results will available at all sites.

IV. RESULT

A simulator with GUI was designed & developed with Microsoft Visual Basic 4.0 in C# language. Simulator accepts text files as input and experiment is performed on dataset as specified in table 1. Simulator produces L_k as result and contains items whose support is greater than given threshold value.

In Tid set, for each item only those Tid are selected that contains specified item as 1st item in transactions. There is a row for each unique item available in dataset. In computation of support for an item, simulator checks that item into its corresponding Tids present in Tid set. In Tid set each item contains distinct Tids.

Comparison of these results with TCVP approach, shows that the items above min-sup value 2 is almost same in both approaches.

Algorithm for partial support count is proposed to remove the redundancy problem of transactions in Tidset assign to each item. It assigns distinct Tid to each item with the effect of less number to items as compare to total support count algorithm[1]. So authors compare both of these algorithm outcomes based on number of items with different values of threshold support and shown in fig 2.

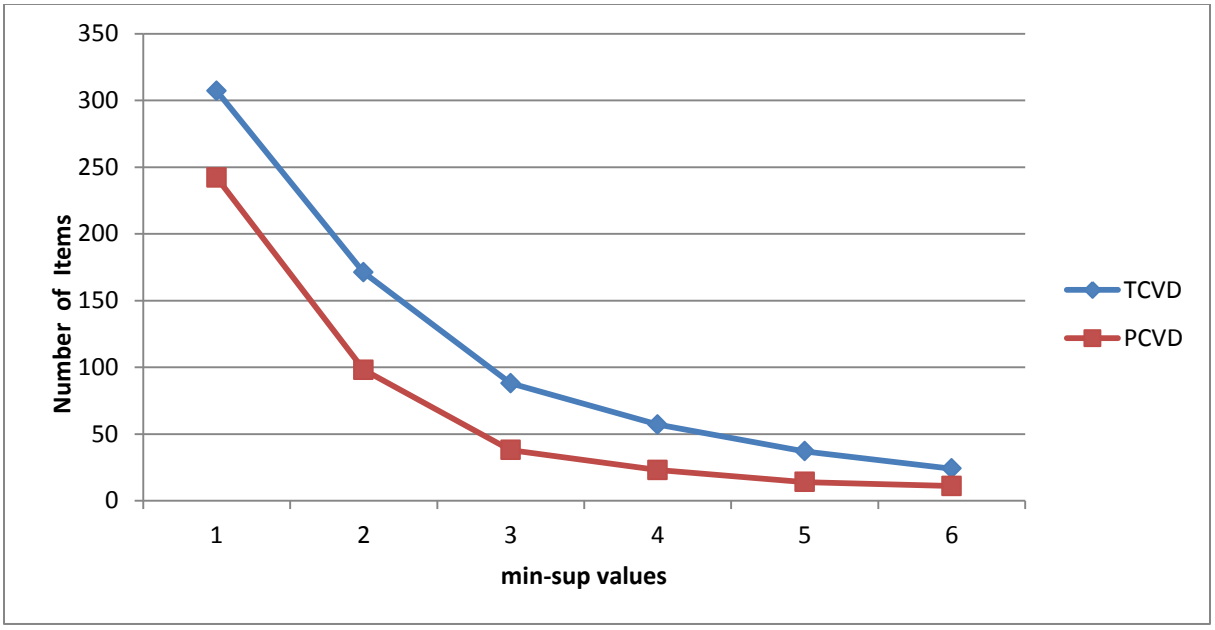

Fig. 2: Relationship between Number of Items and Support Values for TCVD and PCVD

In this paper, partial count on vertical dataset is designed to overcome the communication overhead. This approach is evaluated using support and communication overhead. To perform the experiment simulator with GUI have been designed and developed in Microsoft Visual Studio 6.0.

In this simulator first take dataset as input and use different way to design of tidset from dataset, after that simulator assign different items to nodes and produced item sets of different size at all nodes. Those items have high support value considered as frequent items. Efficiency of proposed approach is better than the existing approach in term of communication overhead as already proved by authors.

This approach removes the problem of redundancy of transactions by using different way to partition the database. This approach based on partial support count but gives the same result as TCVD approach with minimum overhead of communication.

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 work presented in the research can be extended for multi-level and multi-dimensional association rules. These rules can choose the approach for frequent item sets mining according to the properties of the dataset to be mined.

References

[1] Monika, Dr. Harish Rohil "Optimization of Distributed Association Rule Mining Approach Based On Vertical Partitioning", International Journal of Computer Science Engineering (IJCSE) ISSN: 2319-7323, Vol. 2, No.04,Pages: 137-142, July 2013

[2] 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

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

[4] 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.

[5] 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.

[6] 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.

[7] 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.

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

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

[10] 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.

[11] 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

[12] 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.

[13] 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.

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

[15] 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.

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

[17] 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

