International Journal of Latest Research in Engineering and Technology (IJLRET)
ISSN: 2454-5031
www.ijlret.com‖ Volume 2 Issue 3‖ March 2016 ‖ PP 52-56

A Comprehensive Survey of Non- Apriori Parallel Association Rule Mining Algorithms

Prachi Dorle[1], Roshan Gangurde[2]
[1]*Department of MCA, K K Wagh Institute of Engineering Education & Research, Nashik, SP Pune University, India*
[2]*Department of MCA, K K Wagh Institute of Engineering Education & Research, Nashik, SP Pune University, India*

Abstract- There is three main parallel association rule mining algorithms:-Count Distribution algorithm, Data Distribution algorithm and Candidate Distribution algorithm. Existing parallel association rule mining algorithms suffer from many problems when mining huge transactional datasets. One major problem is that most of the parallel algorithms for a shared nothing environment are Apriori based algorithms. Apriori-based algorithms are proven to be non scalable due to many reasons, mainly: (1) the repetitive I/O disk scans, (2) the huge computation and communication 3) great deal of redundancy calculation involved during the candidacy generation. Since the databases to be mined are often very large, and the association rule mining is computationally and I/O intensive, we must rely on high-performance parallel mining method. This paper presents different parallel algorithms given by various researches to generate association rules by various methods. We have done comparative analysis of different algorithms for association rules on various parameters.

Index Terms—Non-Apriori, COFI-Tree, PBFI miner, IFP Tree

I. INTRODUCTION

Association rules mining is one of the important research concepts in data mining which is used to identify the dependencies and correlation among attributes in the database. Most parallel data mining algorithms were developed from the Apriori-based method. Three parallelization approaches for mining association rules are count distribution, data distribution and candidate distribution. The first approach is simple parallelization.

1) Count Distribution - All processors generates the entire list of candidates; thus, each processor independently counts partial supports and then exchanges them with the others. 2) Data Distribution - Each processor generates the disjoint candidate sets and broadcasts them to all others. 3) Candidate Distribution - It determines a heuristic based on support for partitioning both data and candidates.

All the three parallel algorithms suffer from high communication and a great deal of redundancy calculations.

Parallel association rule mining algorithms currently proposed in the literature are not sufficient for extremely large datasets but new solutions that do not heavily depend on repeated I/O scan, less reliant on memory size and do not require a lot of communication costs between nodes are listed over here.

The aim of this paper is to present a review of different Non-Ariori Parallel Association rule mining algorithms.

The paper is organized as follows. Section 2 surveys some Non-Apriori based parallel association rule mining algorithms. Section 3. Conclusion 4. References

II. COMPREHENSIVE SURVEY

This section presents a comprehensive survey, mainly focused on Parallel Association Rule mining algorithms those are not Aprori based. Most of the existing work paid attention to working and its performance.

1. Inverted Matrix

A new parallel association rule mining disk-based algorithm that is based on the Co-occurrence Frequent Item tree (COFI trees) concept is divided into two main phases. [1]

The first one, considered pre-processing, requires only two full I/O scans of the dataset and generates a special disk-based data structure called Inverted Matrix. In the second phase, the Inverted Matrix is replicated among nodes and mined in parallel using different support levels to generate association rules using Inverted Matrix.

The Inverted Matrix layout combines the two layouts Transactional Layout & Horizontal vs Vertical Layout. The idea of this approach is to associate each item with all transactions in which it occurs (i.e. an inverted index), and to associate each transaction with all its items using pointers.

Building this Inverted Matrix is done in two phases, in which phase one scans the database once to find the frequency of each item and orders them into ascending order. The second phase scans the database again once to sort each transaction into ascending order according to the frequency of each item, and then fills in the matrix appropriately.

Each parallel node mines separately each one of these trees as soon as they are built, with minimizing the candidacy generation and without recursively building conditional sub-trees. The trees are discarded as soon as mined. Finally, all globally frequent patterns generated at each parallel node are gathered into the master node to produce the full set of frequent patterns.

The small trees we build (Co-Occurrence Frequent Item-tree, or COFI-tree) are similar to the conditional FP-Tree in general in the sense that they have a header with ordered frequent items and horizontal pointers pointing to a succession of nodes containing the same frequent item, a counter that computes the occurrences of this item in the tree and the prefix tree with paths representing sub-transactions. However, the COFI-trees have bidirectional links in the tree allowing bottom-up scanning as well, and the nodes contain not only the item label and a frequency counter, but also a participation counter. Another difference, is that a COFI-tree for a given frequent item x contains only nodes labeled with items that are more frequent, or as frequent as x.

This algorithm is a disk-based-algorithm that can store the massive size and allow random access, and small memory structures that can be independently created and mined based on the available absolute memory size.

2. Parallel FI-growth association rule mining algorithm

This algorithm is used for rapid extraction of frequent item sets from large dense databases [2]**.** It can efficiently be parallelized in a cluster computing environment. This algorithm need at most two database scans; but need to fit all data structures in main memory.

Similar to the FP-growth algorithm, FI-growth represents the data set as a prefix sharing tree, called an "FI-tree". The FI-growth algorithm does not only avoid time consuming operation of sorting the item sets as in FP-growth, but also provides an efficient operation for FI-tree manipulation. It commonly consists of two phases: FI-tree construction and mining.

In practice, the FI-tree structure consists of two structures in memory: a "header table" that has linkages to each item in an "items-tree". Constructing an FI-tree requires scanning the database only twice: the first scan creates the header table, and the second scan creates the items tree.

In Mining process, there are three steps in this phase. **Branching step**: The first step is to sum the count of each item. **Subset finding step**: The main goal of this step is to find all possible remaining paths. **Pruning step**: The last step is to remove nodes (and also paths) for item sets that fall below the minimum support.

Parallel FI-Growth -We first partition the transaction database into several portions, and distribute them to different processors for computation. Each processor independently constructs its own local FI-tree structure and discovers corresponding frequent item sets. However, all processors need to perform a one-time synchronization to exchange their sub-trees before the last two steps in the mining phase.

The partitioning has an effect on the local counts, namely, a decrease of support values of items residing in each partition. Consequently, some items may be pruned out if local pruning is performed, although their global support counts would exceed the required min_sup value. The more partitions created, the more items may be pruned out locally. This pruning directly affects the correctness of final results.

The synchronizing step begins with all processors exchanging their local header tables. The combined header table is used to determine which local sub-trees should be sent to which target processors.

Exchanging of local header table:

Sending of local sub-tree:

A parallel FI-growth algorithm is used to accelerate association rule mining. This demonstrates that the FI-tree structure is suitable for independent parallel processing, though some synchronization overhead is incurred. To improve parallelization and reduce synchronization overhead, we employ a hierarchical minimum support heuristic for distributed pruning of discovered items. The experiments show that this hierarchical approach yields results very close to pruning based on min_sup setting on single processor configuration.

3. PBFI miner – Parallel Bit Frequent Item set Miner

This algorithm uses bit object to express data and IFP-tree structure to compass the database sufficiently [3]. Secondly, it adopts intersection operation of bit object to find the frequent item sets and their support count, and it doesn't need to generate a great number of condition pattern trees. Thirdly, the algorithm uses parting strategy to achieve near optimal balancing between processors. At last, in order to reduce telecommunication traffic, the processors transfer bit sets instead of local frequent item sets and support count

The reasons that PBFI-Miner has high performance are as follow:

a) The structure of IFP-tree compresses the database sufficiently.

b) It doesn't need to create condition pattern tree, and uses bit operation to simplify support count operation.

c) It does not need to generate candidate item sets in the 478 mining process.

d) It divides the frequent items averagely as possible according to the heavy of mining task. So it achieves near optimal balancing between processors.

e) Each processor communicates with transferring bit sets but not local frequent item sets and support count, so it reduces the telecommunication traffic.

f) When the minimum support is small, the number of frequent item sets and calculation amount will increase. At that time, the above 5 reasons are more obvious.

4. IFP-tree

IFP-tree is a storage structure of frequent information. It is based on FP-tree [3]. In this tree, every node has 4 domains whose name is node-name, node-count, node-link and node-parent. Node-link uses to point the same name node link, and the node-parent uses to point the parent node. Furthermore, we create a head table named Htable for the convenience of traversal. The Htable has two domains with the name item-name and item-head. Node-name and item-head are all signed by the position value of the frequent item.

The construction algorithm of the IFP-tree is as follow:

1) It scans D first time to generate the frequent items and their support count, then creates Ll and position value table.

2) It gains the bit object of every item in D, then inserts the position value into the IFP-tree if the bit value is 1. The rest procedure is the same as literature.

The basic idea of PBFI-Miner is it uses bit object to express data and to improve FP-tree. According to the features of the IFP-tree and the length of the frequent item sets, it uses parting strategy to achieve near optimal balancing between processors. The processors construct bit objects themselves and communicate with each other by bit objects in order to reduce communication load. Every processor uses depth first strategy to mine frequent item sets. The basic method and process of the PBFI-Miner is as follow:

1) If there are n processors involved in mining the transaction database D, the process has four steps. First, D is divided into n parts (n sub-database) averagely. Second each processor P_i (i=1,2, ... ,n) scans its sub-database the first time, and gains the local l-item sets and its support counts. Third, each processor exchanges their local l-item sets and support counts, then merges the l-item sets in order to get the whole frequent items. At last, we get L1 and construct the position value table.

2) Each processor P_i constructs their local IFP-tree in the second scan of D. They adopt parting strategy to deal with frequent items. In order to achieve near optimal balancing between processors, they use the method of modulo n so that adjacent frequent items are distributed into different processors. Another advantage of adopting this method is the smaller of the position value, the heavier of the mining task.

3) Each processor constructs their local bit objects.

4) Each processor mines all frequent item sets with different frequent item as suffix.

5. Parallel FP-tree Constructing Algorithm (PFPTC)

FP-growth is a powerful algorithm to mine frequent patterns and it is non-candidate generation algorithm using a special structure FP-tree [5]. In order to enhance the efficiency of FP-growth algorithm, propose a novel parallel algorithm PFPTC to create sub FP-trees concurrently and a FP-tree merging algorithm called FP-merge which can merge two FP-trees into one FP-tree.

PFPTC - It is obvious that Constructing FP-tree is the key step in FP-growth algorithm, FP-tree contains

compressed message of frequent itemsets, it is easy to mine frequent itemsets from FP-tree. To construct FP-tree, we must scan data base twice, one for creating 1-itemset and one for build FP-tree. To mine useful information from database effectively, we need to solve the efficiency problem of mining to increase efficiency of mining algorithm. Developing parallel algorithm is an effective policy.

First, an initial scan of the database identifies the frequent 1-itemsets in order to enumerate the frequent items efficiently and then we divide the datasets among the available processors. Each processor is given an approximately equal number of transactions to read and analyze. As a result, the dataset is split in n equal sizes. Each processor enumerates the items appearing in the transactions at hand. After enumeration of sub occurrences, a global count is necessary to identify the frequent items. This count is done in parallel where each processor is allocated an equal number of items to sum their sub supports into global count. Finally, in a sequential phase infrequent items with a support less than the support threshold are weeded out and the remaining frequent items are sorted by their frequency.

To build sub FP-trees requires a second complete I/O scan from the dataset where each processor reads the same number of transactions as in the first phase. Using these transactions, each processor builds its own frequent pattern tree that starts with a null root, but need not to build head table and node link.

For each transaction read by a processor, the item list is sorted in descending order according to their frequency. These sorted transaction items are used in constructing the sub FP-Trees.

QFP-growth: mining frequent pattern without conditional FP-trees construction. FP-growth algorithm must generate a huge number of conditional FP-trees recursively in process of mining, it may take too much time and space. We can avoid conditional FP-trees generation in frequent pattern mining. To solve this problem, we develop QFP-growth, an extended FP-growth method for frequent pattern mining from FP-tree.

We have proposed an efficient parallel implementation of an FP-Tree constructing algorithm and an improved method QFP-growth for constructing frequent patterns from FP-tree. QFP-growth can generate frequent patterns without conditional pattern-base and conditional FP-tree generation, and the frequent patterns are generated in order.

6. Parallel Association Rules Mining Algorithm with Privacy preserving (PARMA-P)

A simple and effective method of parallel association rules mining which is based on privacy protection is Parallel Association Rules Mining Algorithm with Privacy preserving (PARMA-P) [4]. It could achieve effective concealment of frequent item-set. Then the association rules by the means of using imported Hash assignment strategy in frequent item sets of FP sub tree could be protected.

The proposed algorithm in PARMA-P applies privacy protection method on the parallel mining algorithm. The principle of it is based on the hypothesis of taking the host side as a trust, using input-based allocation strategy of taking the sub-machine IP as a hash factor so as to achieve the dual mapping of Hash frequent item sets, ultimately to protect the privacy of association rules. The algorithm for data privacy protection plays a good role and is practical.

PARMA-P algorithms which could protect frequent item sets by using heuristic technology and input Hash allocation strategy, that is, host as the trust side is responsible for primary coding and the distributions of sub-FP trees by using Hash function. When the applications are applied by sub computers, then, it will take the method of coding reversed with the composite sets of the results of sub parallel data mining to realize the privacy protect of association rules.

Description of PARMA-P –

Hash distribution and coding –

Suppose {α1,α2,.....,αn } is a frequent 1-itemsets, then the subset of FP tree which suffix is αi(1,2,.....,n) being called Suffix FP-sub-tree group. The front pointers of those are put in linear list and being set up distribution flags: '0' indicates undistributed and '1' indicates distributed. When there is application from sub-computer, the host allocates the sub tree according to Hash function randomly. The results of distribution should be stored in the database which will be used in the process of reduction of frequent sets. The conflicts produced by the process of distribution can be resolved by linear sampling method and then complete the mapping of frequent 1-itemsets to a two dimensional table by the following hash table.

When the application has been sent out, the host verified the identity of sub computer, allocate the suffix FPsub trees through Hash function, meanwhile update IP and flag. The privacy protection of data is implemented by

the use of heuristic technique, here, mainly by the use of mapping with frequent item sets.

III. CONCLUSION

Mining frequent patterns from transactional database is a significant problem in data mining research. There are many sequential or parallel methods had been proposed to solve this problem. However, the execution time increases quickly with an increase database size or a decrease in the given threshold. In mining extremely large transactions, we should neither work on algorithms that build huge memory data structures nor on algorithms that scan the massive transactions many times. Communicating the frequent patterns can be considered a costly task that consumes most of the mining time.

IV. REFERENCES

[1]. Mohammad El-Hajj , Osmar r. Za¨ıane ," Parallel Association Rule Mining with Minimum Inter-Processor Communication", *proceedings of the 14th internationa*l 2003 IEEE

[2]. Bundit Manaskasemsak, Nunnapus Benjamas, Arnon Rungsawang," *Parallel Association Rule Mining based on FI-Growth Algorithm*" , 2007 IEEE

[3]. Junrui Yang, Yashuang Yang," A Parallel Algorithm for Mining Association Rules","*IEEE 2010 International Conference on Networking and Digital Society*"

[4]. Xue Ping Zhang1,YanXia, Nan Hua," Privacy 3arallel Algorithm for Mining Association Rules and Its Application in HRM*"," 2009 Second International Symposium on Computational Intelligence and Design*"

[5]. Yong-jie lan, Yong qiu," Parallel Frequent Item sets Mining Algorithm without Intermediate Result"," *Proceedings of the Fourth International Conference on Machine Learning and Cybernetics, Guangzhou, 18-21 August 2005*"