An Efficient Parallel Approach for Frequent Itemset Mining of Incremental Data

Mrs. Chetashri Bhadane, Dr. Ketan Shah, Mrs. Prajakta Vispute

Abstract - Frequent itemset mining is the essential step of data mining process. Further frequent itemset is a primary data obligatory for association rule mining. The Apriori and FP tree are conventional algorithms for mining frequent itemset and envisaging association rules based on it for knowledge discovery. The process of updating database continuously is known as incremental data mining. In real life, database updates recurrently where exactly conventional algorithms perform incompetently. If we could use the previous analysis to incrementally mine the frequent itemset from the updated database, the mining process would become more efficient and cost of mining process would be minimized. In this research, we propose a novel incremental mining scheme with a parallel approach for discovering frequent itemset. It uses a data structure called IMBT. It is a Incremental Mining Binary Tree which is used to record the itemset in an efficient way. Furthermore, our approach needs not to predetermine the minimum support threshold and scans the database only once.

Index Terms - Apriori , FUFP Algorithms, FP Tree Algorithm, Frequent Itemset, IMBT Structure, Incremental Data Mining, Parallel Data Mining,.

1 INTRODUCTION

Data mining [1] is one of the fastest growing fields in the computer industry. The data base system industry has an evolutionary path in the development of the functionalities like Data collection and database creation, data management and advanced data analysis which includes Data mining and data warehousing. Among the various data mining applications, mining association rules is an important one. The strategies for mining frequent itemset, which is the essential part of discovering association rules, have been widely studied over the last decade such as the Apriori, and FPgrowth. In the traditional frequent itemset mining algorithms, a strict definition of support is used for every item in a frequent itemset occurring in each supporting transaction.

However, in real-world applications, new transactions are usually inserted into databases. However, most mining methods did not involve in dynamic of data, that is, with time running, the algorithms can't effectively deal with the data set including new data and the old data, and even the future data, which may be important for theoretical analysis and practical application, for which the concept of Incremental data mining [3][4][6][7] is introduced.

1.1 Need of Incremental Mining

In real-world applications, transaction databases usually grow over time and the association rules mined from them must be re-evaluated. Some new association rules may be generated and some old ones may become invalid. Conventional batch-mining algorithms solve this problem by reprocessing the entire new databases when new transactions are inserted into original databases. They, however, require lots of computational time and waste existing mined knowledge.

The Apriori algorithm in which candidate itemsets are generated and tested level-by-level, may cause iterative database scans and high computational costs. The Frequent-Pattern-tree (FP-tree) structure for efficiently mining association rules without generation of candidate itemset was used to compress a database into a tree structure which stored only large items. Both the Apriori and the FP-tree mining approaches belong to batch mining.

In real-world applications, new transactions are usually inserted into databases incrementally. In this case, the originally desired large itemset may become invalid, or new large itemset may appear in the resulting updated databases. The solution to batch mining algorithms such as Apriori and FP-tree is incremental mining algorithms. Incremental mining provides the solution to this problem. Incremental data mining tries to use the previous results as the basis to incrementally mine the database with new transactions.

There are two performance issues on incremental data mining. The first one is the need to rescan the original database to enumerate the support count of the patterns when a database is updated. The second issue is how to deal with the threshold changes during the lifetime of the database.

2 RELATED WORK

At present there are many elegant and practical data mining algorithms, models and technologies present. These methods have good performances and practical perspectives for static data discovering association rules, decision tree classification algorithms etc.

Several methods exist for frequent itemset mining. The most traditional method is Apriori algorithm which uses the lexicographic tree [6] to record the itemset generates level-wise candidates and rescans the database to enumerate the support count of each itemset and finds frequent itemsets. Han et al. [8] proposed the FP-growth

algorithm using FP-tree data structure. FP-tree takes the advantage of common paths to record the sorted frequent items in transactions. In addition, FP-growth uses a header table to record the frequent items and point to the node in the FP-tree to improve the performance of mining frequent itemsets.

Hong et al. [7] proposed the FUFP algorithm for incremental mining. The algorithm is based on the FPtree structure that can be used to mine frequent itemsets without candidate generation. FUFP needs to rescan the original database in some cases when an infrequent itemset becomes frequent or the threshold is changed. However, if the database is large the FP-tree will be large and the space requirement for recursion is a also huge [12]. As a result, Lin and Hong et al. [9] proposed the Pre-FUFP algorithm to improve the performance over FUFP. Pre-FUFP sets two thresholds and records the pre-large itemsets, which can potentially become frequent itemsets when new transactions are added. Although the Pre-FUFP algorithm decreases the probability of rescanning the original database, the size of the original database still affects the cost when the database is updated. Further, C. H. Yang and D. L. Yang proposed IMBT (Incremental Mining Binary Tree) data structure for mining frequent itemsets more efficiently. One of the feature of this mining method using IMBT is, it do not entail minimum support threshold at the beginning of mining process which is mandatory in traditional mining methods.

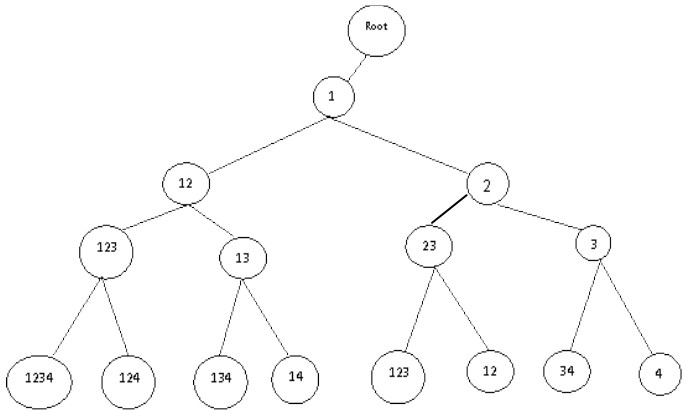

Fig 1. The IMBT tree

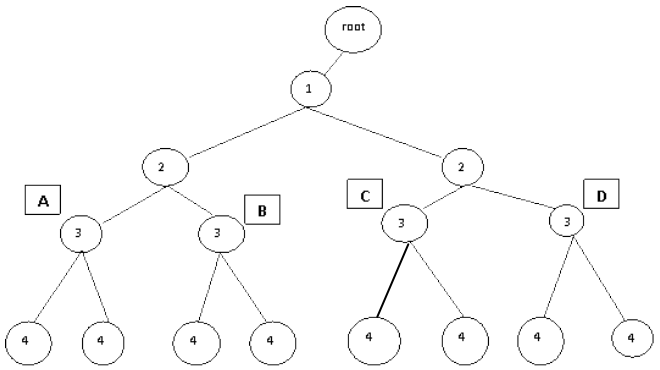

Fig 2. The simplified IMBT tree

3 EXISTING TECHNIQUES FOR INCREMENTAL MINING

3.1 Incremental Mining Method using IMBT

To make use of already mined result, C. H. Yang and D. L. Yang, proposed mining method using new tree structure rather than FPtree. It uses a tree structure called IMBT (Incremental Mining Binary Tree) to enumerate the support count of each itemset in an efficient way after the transactions are added or deleted. Instead of rescanning the database many times to enumerate the support count of the itemsets after the database update, it processes a transaction at a time and record the possible itemsets in a data structure that can reduce the processing and IO time.

3.2 IMBT - Incremental Mining Binary Tree

Here we will see how IMBT creation from given dataset. Consider a transaction I = {1, 2, 3, 4} for which IMBT is created as in Fig.1 and Fig.2 shows its simplified IMBT tree [6].

For simplicity, we use the numeric number to represent the item. The root is an empty node to hold the pointer of the first tree node where it stores the smallest 1-itemset encountered so far. Additional left or right child is added according to the items contained in a transaction. The construction of the IMBT tree is complete after the last transaction has been processed.

3.3 Adding transactions into Database

First using original database as given in Table 1, construct the IMBT as discussed above. An addition database DB+ containing the transactions to be added to the original ODB database shown in table 2. There are four items (0, 6, 7, 8) in the first transaction (Tid 001). In this case, we will insert new nodes and increase the support count of one existing itemset. The final IMBT [6] is as shown in figure 3.

Table 1
The original database *ODB*

TID	Items
001	4, 5, 7

Table 2
Addition Database *DB+*

TID	Items
001	0, 6, 7, 8
002	0, 5, 7

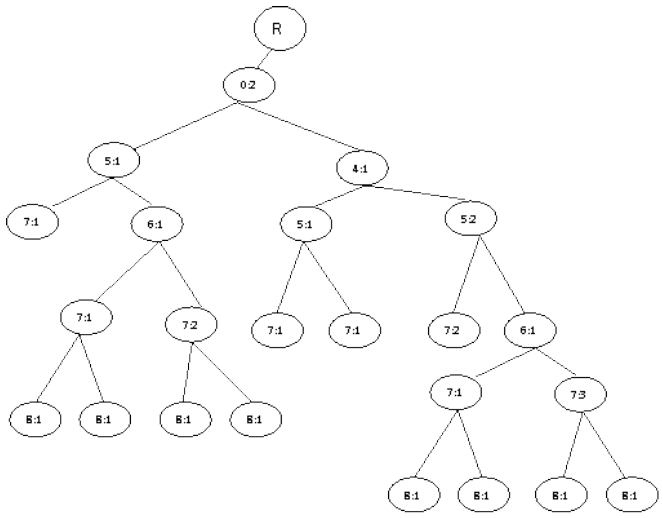

Fig 3: The IMBT after completing the process of all transactions in the database DB+

The advantage of this mining method is that it is not needed to rescan the original database after new transactions are added. Therefore, no matter what the original database is, the size of the database, or the length of the transaction, there is no effect on mining the database incrementally after the updates. Moreover, the cost of updating the IMBT is minimal since we do not scan the whole IMBT for node updates.

3.4 Mining frequent itemsets after the database update

Given an itemset X, if SupDB(X) is larger than or equal to the minimum support (min_sup) determined by users, the itemset X is called a frequent itemset. If an itemset X is infrequent, the left-side descendants of the itemset X are infrequent, and thus the algorithm will stop traversing the left-side descendants. This can improve the performance of mining process. When the database is updated (added or deleted), the FI-table also needs to be updated incrementally. After constructing the IMBT, we will traverse the tree to discover the frequent itemsets based on the minimum support specified by users. The mining results are kept in a FI-table for further reference. Since no support threshold is required during the tree construction, users can mine the frequent itemsets with any threshold before or after the database update.

The method using IMBT mines the newly added transactions by reusing the mined results from the original database and performance has been greatly improved. But this method faces the problem of limited memory space when IMBT gets processed in the main memory. IMBT structure can get too large to fit into main memory. When this occurs the process either dies or must start using virtual memory. Since the databases to be mined are incremental in nature and often very large (measured in gigabytes and even terabytes), applying parallel mechanism to the existing serial mining algorithms would be an attentive measure in data mining field.

4 Problem Definition

As the data sizes increase, from gigabytes to terabytes or even larger, sequential data mining algorithms may not deliver results in a reasonable amount of time. Even worse, as a single processor alone may not have enough main memory to hold all the data, a lot of sequential algorithms could not handle large scale problems or have to process data out of core, further slowing down the process. In parallel environment, by exploiting the vast aggregate main memory and processing power of parallel processors, parallel algorithms can have both the execution time and memory requirement issues well addressed [10].

5 Proposed System

To overcome this problem of limited memory space, a novel method is suggested in this paper which uses parallel approach for counting frequent itemsets using IMBT data structure. The main objective of this research is to parallelize IMBT creation and frequent itemsets counting, to be run on multiple machines at high speed when memory and processor limitations would make it impractical to run the algorithm on a single machine. Our research work mainly focuses on how this parallelism can be achieved efficiently without significant communication and processing overhead.

5.1 Design

Initially the input file will be divided into separate 'chunks'. Then parallelism will be applied to each chuck of input file. As this method will use basic data structure IMBT and will be for incremental mining, it will not use any minimum support threshold for IMBT tree creation. Once local IMBT will be created on separate nodes, frequency count of each itemsets will be counted on individual nodes. The generated frequent itemsets will be merged together to generate a super-set of the actual frequent itemsets in the dataset as a whole. Finally, minimum support threshold specified by user will be applied to super-set of all itemsets for counting frequent itemsets. The intuitive designing of the proposed approach

is given in following figure 4. For implementation of this approach Map Reduce Framework is an appropriate solution.

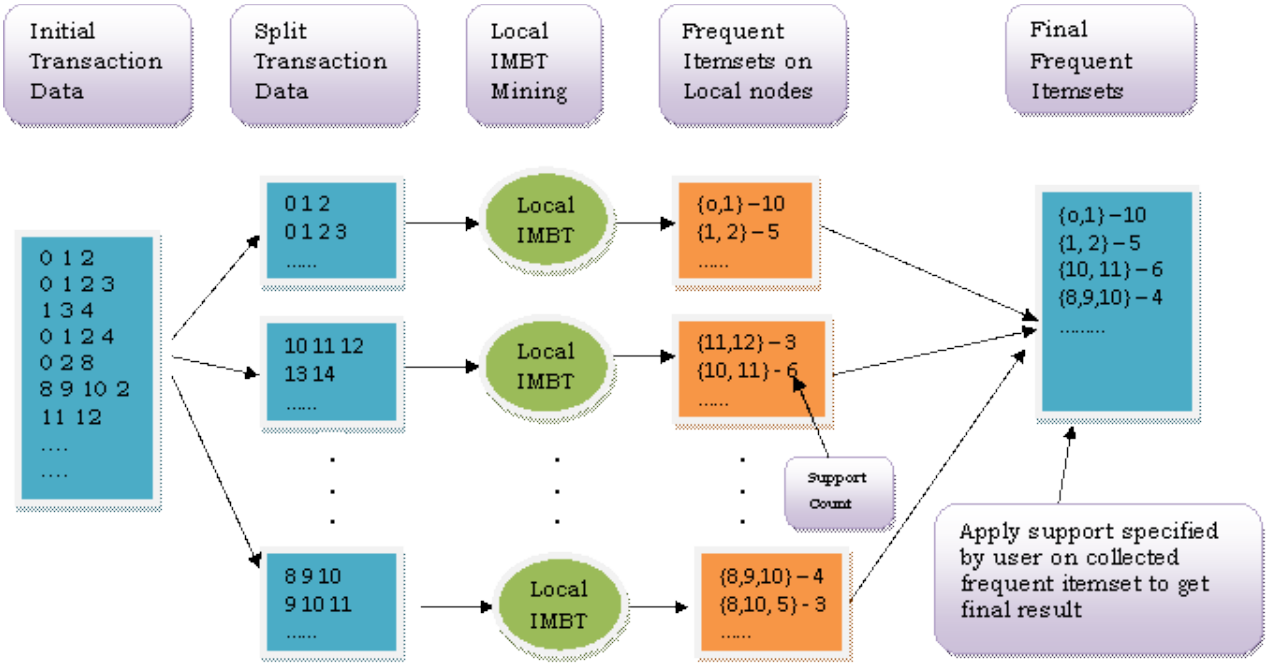

Fig 4. Parallel IMBT tree creation & frequent itemset counting

5.2 MapReduce Architecture

MapReduce is standard software architecture, developed by Google, which aids in the design and execution of large scale data processing tasks. There are two primary components to the architecture: MAP and REDUCE. The MAP step takes in a chunk of input data and emits key-value pairs which represent that data. In the word count example below the keys are the text of each word, and the value is a count (one). The reduce step received key-value pairs in which all identical keys are guaranteed to arrive at the same reduce function. In the word-count example the reduce function simply sums the values for each key and outputs the key with a total count. The output from all reduce steps is appended to an output file.

5.3 Methods for Parallelism

When it comes to the parallel computation techniques, data has to distribute among different nodes. Data Parallelism and Task Parallelism are two approaches using which data distribution task can be accomplished. Each one of these methods specifies their data distribution policy and based on it, node's work format. Data parallelism focuses on distributing the data across different parallel computing nodes. In data parallelism each processor performs the same task on different pieces of distributed data.

For our research work data parallelism will be a good choice. As real life dataset used to be large, keeping it on each node will need more memory space. 'Database Sharding' is the one method to achieve proper data distribution on different node. Database sharding breaks down large datasets into smaller chunks called "shards" and spreads those across a number of different systems.

6 Conclusion

In this paper we have discussed the traditional data mining techniques such as Apriori, FP Tree, and FP growth. Although these techniques give the efficient performance, they cannot handle the real world dynamic data. So the incremental mining technique FUFP (Fast Updated Frequent Pattern) is proposed which uses basic data structure FP tree again. However, if the database is large the FP-tree will be large and the space requirement for recursion is also very huge. Also it still needs to rescan the original database in some cases when an infrequent itemset becomes frequent or the threshold is changed. As a solution to this rescanning problem an incremental mining technique using IMBT is proposed to enumerate the support count of each itemset in an efficient way after the new transactions are added or deleted. Since no support threshold is required during the tree construction, this method allows users to mine the frequent itemsets with any threshold before or after the database update. Method using IMBT mines the newly added transactions by reusing

the mined results from the original database and performance has been greatly improved. But this method faces the problem of limited memory space when IMBT gets processed in the main memory. IMBT structure can get too large to fit into main memory.

So we have proposed a novel method in which database will be distributed among different nodes and local IMBT will be created using which frequent itemsets will be counted incrementally. Intuitively we can say that this method will be more efficient than existing non-parallel incremental methods such as Apriori and FPtree methods.

REFERENCES

[1] Chen, M. S., Han, J., & Yu, P. S. "Data mining: An overview from a database perspective", IEEE Transactions on Knowledge and Data Engineering," 8(6), 866–883,1996

[2] J. Han, J. Pei, Y. Yin, "Mining Frequent Itemsets without Candidate Generation," ACM SIGMOD International Conference on Management of Data, 2000.

[3] D. W. Cheung, J. Han, V. T. Ng, C. Y. Wong, "Maintenance of discovered association rules in large databases: An incremental updating approach," In The twelfth IEEE international conference on data engineering, pp. 106–114, 1996

[4] C. W. Lin, T. P. Hong, W. H. Lu, "The Pre - FUFP algorithm for incremental mining," Expert Systems with Applications, 2008

[5] C. I. Ezeife, Y. Su, "Mining incremental association rules with generalized FP-tree," In Proceedings of the 15th conference of the Canadian society, pp.147–160, 2002

[6] C. H. Yang and D. L. Yang, "IMBT-A Binary Tree for Efficient Support Counting of Incremental Data Mining," 2009 International Conference on Computational Science & Engineering

[7] T. P. Hong, C. Y. Wang and Y. H. Tao, "A new incremental data mining algorithm using pre-large itemsets," Intelligent Data Analysis, Vol. 5, No. 2, 2001, pp. 111-129.Fghh

[8] Gosta Grahne, Member, IEEE, and Jianfei Zhu, Student Member, "Fast Algorithms for Frequent Itemset Mining Using FP-Trees," IEEE Transactions On Knowledge And Data Engineering, Vol. 17, No. 10, October 2005 1347.F

[9] Rakesh Agrawal and John C. Shafer, "Parallel mining of association rules," IEEE Trans. On Knowledge and Data Engineering, 8(6):962-969, December 1996.

[10] Jianwei Li, Ying Liu, Wei-keng Liao, Alok Choudhary, "Parallel Data Mining Algorithms for Association Rule and Clustering," 2006 by CRC Press, LLC

[11] Frequent Patterns without Candidate Generation: A Frequent-Pattern Tree Approach," Data Mining and Knowledge Discovery, 8, 53–87, 2004

[12] Jian Pei, Jiawei Han, Hongjun Lu, Shojiro Nishio, Shiwei Tang, Dongqing Yang, "H-Mine: Fast and space-preserving frequent pattern mining in large databases," Data Mining and Knowledge Discovery, 8, 53-87, 2004