ISSN 2250-3153

Parallel version of tree based association rule Mining Algorithm

***Jadhav Arvind Uttamrao**

Department of Computer Science, P.E.S.C.O.E. Aurangabad.
MH-India

Abstract- Tree Based Association Rule Mining (TBAR) algorithm decreases the number of data base scans during frequent item set mining to improve association rule mining process. In this paper, we present parallel version of TBAR algorithm. Parallel version of TBAR algorithm decreases the time required for association rule mining process by making candidate item set generation, frequent item set generation and association rule generation process parallel.

Keywords - Data mining, Association rules mining, Tree Data Structure, Parallel Programming, Scala, Ateji PX.

I. INTRODUCTION

Association rule mining[1] is to find out association rules that satisfy the predefined minimum support and confidence from a given database. The problem is usually decomposed into two sub problems. One is to find those itemsets whose occurrences exceed a predefined threshold in the database; those itemsets are called frequent or large itemsets. The second problem is to generate association rules from those large itemsets with the constraints of minimal confidence. Suppose one of the large itemsets is Lk, $Lk = \{I1, I2, \ldots, Ik\}$, association rules with this itemsets are generated in the following way: the first rule is $\{I1, I2, \ldots, Ik-1\} = \{Ik\}$, by checking the confidence this rule can be determined as interesting or not. Then other rule are generated by deleting the last items in the antecedent and inserting it to the consequent, further the confidences of the new rules are checked to determine the interestingness of them. Those processes iterated until the antecedent becomes empty.
All the association rules generated by association rule mining algorithms have some support and confidence. The support and confidence of association rule $A \rightarrow B$ is calculated by the formulae:

Support $(A \rightarrow B)$ = Probability $(A \cup B)$

Confidence $(A \rightarrow B)$ = Probability (B/A).

The first sub-problem can be further divided into two sub-problems: candidate large item sets generation process and frequent itemsets generation process. We call those item sets whose support exceed the support threshold as large or frequent itemsets, those itemsets that are expected or have the hope to be large or frequent are called candidate itemsets.

F. Berzal [9] introduced TBAR algorithm for association rule mining on relational databases. TBAR out performs Apriori. TBAR stores the v>, which 'a' is an attribute, 'v ' is the value of 'a'. If the value of 'a' in tuple t is 'v', then t contains pair <a: v >.
R. Agrawal [2] introduced three algorithms for parallelization of association rule mining algorithm. He divided the association rule mining task in two sub parts for parallelization. In first part he considered frequent itemset generation. For this part he gave three parallelization techniques. And second part association rule generation in this part he simply distributed the generated frequent itemsets among multiple processors.

The rest of the paper is organized as follows. In Section II, we will see the work in the field of association rule mining and the attempts made by researchers to improve association rule mining algorithm's performance. In Section III, we will see the details of original TBAR algorithm. Section IV gives the details about parallel version of TBAR algorithm. The Section V gives the experimental results of TBAR and parallel TBAR applied on different datasets.

II. LITERATURE SURVEY

Since, R. Agrawal [1] put forward the Apriori algorithm for association rule mining on transaction databases this is frequently researched area of Data Mining. Many researchers tried to put a new concept based on Apriori algorithm to make association rule mining process faster.

F. Berzal [3] then put the TBAR algorithm for Association rule mining on relational databases. His algorithm uses tree data structure to store frequent itemsets which decreases the number of database scans which results in fast association rule mining algorithm. De-Chang Pi [4] introduced the Super Tree Based Association Rule mining algorithm (STBAR) which is improved version of TBAR. STBAR works on transactional databases only. Unlike the <column name: value> pair in TBAR, STBAR uses $< column\ name: value: Flag >$. The flag will decide whether an item can concatenate the items found in the paths from the root of the tree to the current item or not.

Many researchers tried to improve performance of association rule mining algorithm by using parallel programming. For example R. Agrawal [2] gave three approaches for parallelisation of association rule mining. Eui-Hong Han [5] brought forward aparallel algorithm based on Apriori algorithm.R. Osmar [6] gave a parallel algorithm which mines association rule without candidacy generation. Zaki et al. [7] proposed the Common Candidate Partitioned Database (CCPD) and the Partition Candidate Common Database (PCCD) algorithms, which both

are Apriori-like algorithms. J. Park [8] put the parallel data mining (PDM) algorithm. This algorithm is a parallel implementation of the sequential Dynamic Hashing and pruning (DHP) algorithm and it inherited its problems, which makes it impractical in some cases.

III. TBAR ALGORITHM

F. Berzal [3] introduced an algorithm for association rule mining from relational databases based on tree data structure. TBAR works similar to many of the algorithms. It divides association rule mining process in two steps. The first one is frequent itemsets generation and second is association rule generation from frequent itemsets. In first step TBAR store all frequent itemsets in tree data structure. In this algorithm each node of tree is of type <a: v>, where a column name and v is value. The tree will contain as many number of nodes for column a as the number of different values present in column a. The number of levels in tree will not be greater than the number of columns in database. The TBAR algorithm is as follows:

```
set.Init (MinSupport);
itemsets = set.Relevants(1);
k = 2;
while (k ≤ columns && itemsets ≥ k)
{
   itemsets = set.Candidates(k);
   if (itemsets > 0)
   itemsets = set.Relevants(k);
   k + +;
}
```

In this algorithm the first step will initialize the data structure. In the second step the algorithm will scan database and will generate L1 which will contain all the <column name: value> pairs. Then the algorithm will iterate for n number of times where n is the number of columns present in the database and in each iteration it will generate candidate itemset Ck and after scanning database for setting frequency of all the members of Ck it will generate Lk by removing the members which have support lass than minimum support.

The example database for TBAR association rule mining is:

A	B	C
0	0	0
0	0	1
0	1	1
1	1	1
1	1	1

Table 1: Example input data to TBAR algorithm.

The tree generated for above database will look like:

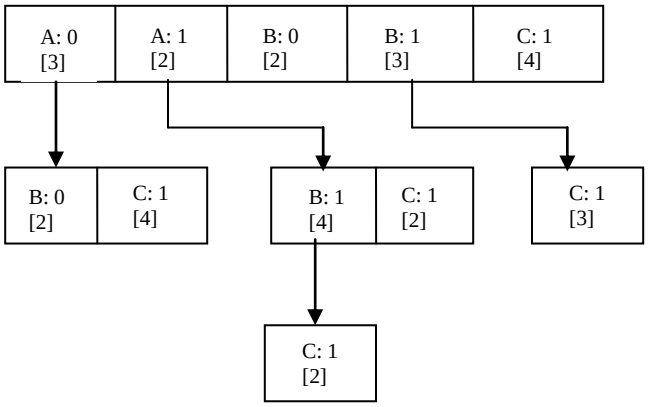

Fig 1: Tree generated by TBAR on input data in Table 1.

Then the algorithm will traverse the tree to generate frequent itemsets and will generate association rules from them. The association rule generation method will require two iterators one will extract all frequent itemsets with k>=2 from the tree. The second iterator will give all the proper subsets of given frequent itemset.

IV. PARALLEL VERSION OF TBAR

R. Agrawal [2], in his paper gave three different algorithms for parallelization of association rule mining algorithm. The first is count distribution. In this algorithm the task of support counting is distributed among the processors. The processors scans there local database and will generate local count. Then all the local counts are collected by processor to generate global count.

The second algorithm is data distribution algorithm.In this algorithm the processors will generate there different candidate item sets and count support value for them. During every pass the processor swill exchange their candidate itemsets which will require fast network support otherwise the performance of algorithm will decrease.

The third algorithm is candidate distribution. In this algorithm the processors are made to work independently by partitioning both data and the candidates. In Parallel version of TBAR algorithm the candidate itemset generation and relevant itemset generation is parallel. We are using shared memory model with threads in our implementation. The threads created by parallel loops will process on shared database to generate results. As we are using MySQL as backend the task of data read cannot be made to execute parallel because of limitation of MySQL drivers. But the task of candidate item set generation can be made parallel because it does not contain any database interactions. Once the Candidate itemset is generated by parallel threads then we have to sort the itemsets.

After the tree is built we made the task of association rule generation multi-threaded which will produce association rules from tree in parallel.

V. EXPERIMENTAL RESULTS

We implemented TBAR algorithm in both Java and Scala using Java Data Base Connectivity (JDBC) to communicate with database. Parallel version of TBAR is also implemented in both Java and Scala.

For implementation of parallel version of TBAR in Java we used the tool called Ateji PX [9]. Ateji PX is extension of Java which gives out an easy way for converting sequential Java code to parallel Java code. We can execute the parallel operation on either single multicore processor or in cluster or on multi core gpu.

Conversion of Java code to parallel Ateji PX code is very easy process [10], we did this conversion for parallelisation of For loops only. The format of parallel For loop is:

For || (inti: I)

This will create multiple threads of the code present in for loop. The default number of threads created depends on the availability of resources. We can control the number of threads created and we can also assign name to every thread branch.

On other side Scala is programming language which combines characteristics of both object oriented and functional programming language [11]. The special thing provided by Scala called Parallel Collection allows us to perform all the tasks on Collection objects parallel such as sorting, searching, reversing, etc. We can also make the loops in Scala parallel by using the Parallel Collection class. In both the cases we used MySQL as back end database. Both Java and Scala give same results and almost same performance. The Parallel version of TBAR has been applied on several datasets available on UCI Machine Learning Database Repository [12].

Sr no.	Data Set Name	Number of Attributes	Number of Instances
1	Chess	37	3196
2	Mushroom	22	8124
3	Soybean	35	307
4	Zoo	18	101

Table 2: List of Datasets on which Parallel TBAR is applied

Parallel version of TBAR gives the better results for above datasets over sequential version of TBAR the results are:

Sr no.	Data Set Name	Time in milliseconds		Enhancement in speed
		Sequential TBAR	Parallel TBAR	
1	Chess	21376	17142	19.80%
2	Mushroom	4500	3160	29.77%
3	Soybean	6820	5611	17.73%
4	Zoo	3402	3230	5.05%

Table 3: Experimental Results on datasets in Table 2

The comparison of sequential and parallel TBAR is shown in following graph:

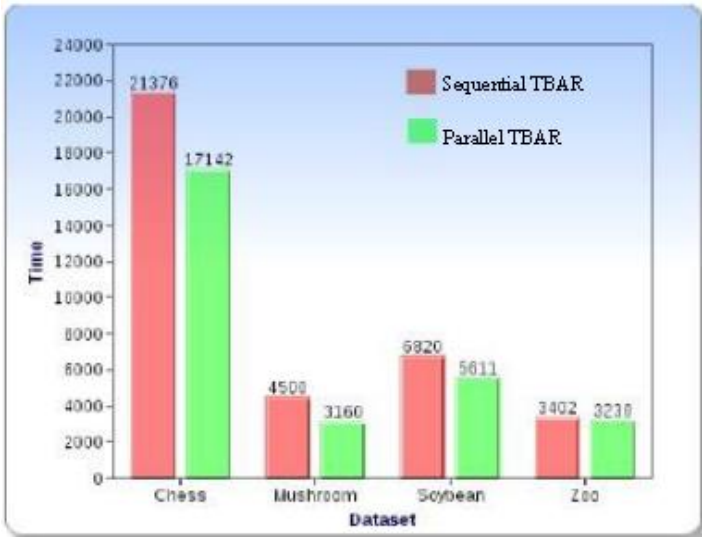

Fig. 2: Comparison graph of TBAR and parallel TBAR on datasets in Table 2.

The parallel version of TBAR takes variable amount of time for different values of minimum support. The time taken by parallel TBAR to find association rules on mushroom dataset for different values of minimum support is:

Sr no.	Minimum Support	Time in milliseconds	
		Parallel TBAR	Sequential TBAR
1	50	9547	11657
2	60	1110	3188
3	70	562	2694
4	80	563	2625
5	90	1079	2438

Table 4: Experimental results on Mushroom dataset

The graphical representation of above table is:

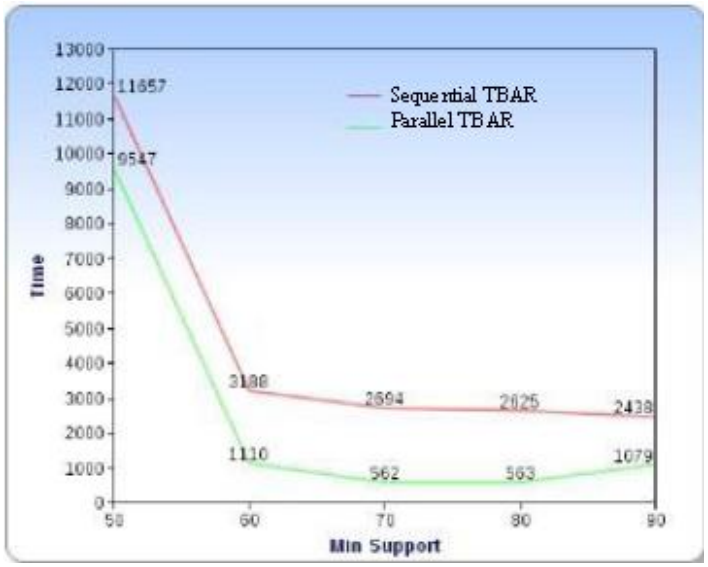

Fig 3: Comparison graph of TBAR and Parallel TBAR on Mushroom dataset.

VI. CONCLUSIONS

We presented Parallel version of Tree Based Association Rule mining Algorithm. TBAR itself outperforms Apriori algorithm by using tree data structure to store frequent itemsets. In parallel version of TBAR we made the tree generation and association rule mining from tree parallel to improve performance of the algorithm. Parallel version of TBAR decreases the time complexity of association rule mining process by increasing the use of all processor cores.

REFERENCES

[1] Agrawal R., Imielinski T. and Swami A. N., "Mining Association Rules between Sets of Items in Large Databases," ACM SIGMOD International Conference on Management of Data, 1993, pp. 207-216.

[2] R. Agrawal and J. C. Shafer, "Parallel Mining of Association Rules," IEEE Transaction on Knowledge and Data Engineering, 1996, pp. 962-969.

[3] Fernando Berzal, Juan-Carlos Cubero and Nicolas Marin, "TBAR: An Efficient Method for Association Rule Mining in Relational Databases," Data & Knowledge Engineering, 2001, pp. 47-64.

[4] De-chang Pi, Xiao-Lin Qin, Wang-FengGuand Ran Cheng, "STBAR: A More Efficient Algorithm for Association Rule Mining," Fourth International Conference on Machine Learning and Cybernetics, Guangzhou, 2005, pp. 18-21.

[5] Eui-Hong Han, George KarypisandVipinKumar, "Scalable parallel data mining for mining association rules," IEEE Transaction on Knowledge and data engineering, 2000, pp. 337-352.

[6] Osmar R. Zaiane, Mohammad El-Hajj and Paul Lu,"Fast Parallel Association Rule Mining Without Candidacy Generation," ICDM 2001, Proceedings IEEE International Conference,2001, pp. 665-668.

[7] M. J. Zaki, M. Ogihara, S. Parthasarathy, and W. Li,"Parallel datamining for association rules on shared-memory multi-processors," In Supercomputing, 1996.

[8] J.S. Park, M.S. Chen and P.S. Yu,"Efficient parallel data mining for association rules,"ACM Int'l Conf. Information and Knowledge Management, 1995, pp. 31-36.

[9] Patrick Viry,"Ateji PX for Java Parallel programming made simple," Ateji White Paper, 2010.

[10] Patrick Viry,"Simplicity and Performance for Multicore Enabling our Data-Intensive Operations,"Ateji White Paper, 2010.

[11] Martin Odersky, Lex Spoon and Bill Venners,A comprehensive step-by-step guide Programming in Scala ,2nd edition.

[12] http://archive.ics.uci.edu/ml/datasets.html

AUTHORS

First Author – Jadhav Arvind, M-tech in Web Technology, Assistant Professor at PES COE Aurangabad.