Global Journal of Computer Science and Technology
Software & Data Engineering
Volume 13 Issue 5 Version 1.0 Year 2013
Type: Double Blind Peer Reviewed International Research Journal
Publisher: Global Journals Inc. (USA)
Online ISSN: 0975-4172 & Print ISSN: 0975-4350

Efficient Distributed Algorithm using Association Rule Mining for Large Database

By N.K. Sharma & Dr. R.C. Jain
S.A.T.I. Engineering College, India

Abstract - Day by day data is increasing due to effectively computerization, implementation, and digitization in various sectors i.e. science, research, industry, business and many other areas. Data mining is the process of extracting valuable and useful information from this very large database.

Association rule is a concept in which buyer usually by a specific combination of different products together while association rule mining is an important technique to show relationship between various items stored in the database. In this paper we take one master tree called root and various branches process the database rather than a multiple FP-tree.

Keywords : knowledge discovery, FP-tree, a-priori, association rule(s), parallel processing and frequent item set.

GJCST-C Classification : H.2.8

Strictly as per the compliance and regulations of:

Efficient Distributed Algorithm using Association Rule Mining for Large Database

N.K. Sharma [α] & Dr. R.C. Jain [σ]

Abstract - Day by day data is increasing due to effectively computerization, implementation, and digitization in various sectors i.e. science, research, industry, business and many other areas. Data mining is the process of extracting valuable and useful information from this very large database.

Association rule is a concept in which buyer usually by a specific combination of different products together while association rule mining is an important technique to show relationship between various items stored in the database. In this paper we take one master tree called root and various branches process the database rather than a multiple FP-tree.

Keywords : knowledge discovery, FP-tree, a-priori, association rule(s), parallel processing and frequent item set.

I. Introduction

Utilization of information technology in various sectors, the database is increasing day by day. Computerization of land records, registration of vehicles, a collection of various taxes and conduction of any competitive examination etc. require certain attention. Research has been performed on serial data mining but it is not suitable for huge data. In this situation parallel data mining is the viable solution [1] [2] [3] [4]. In parallel data mining available work is to be divided in various processors to get a solution as fast as possible. There are three components to the work i.e. computation, access to the data set and communication between the processor. .Access to the data set is most costly, followed by communication, with computation being relatively cheap [5] [7] [8].

There are three basic strategies [16] [17] [18] for parallelization:

1. Each processor has access to the whole data set, but each processors place in a different part of the search space, starting from a randomly chosen initial position and this strategy is termed as independent search.
2. Each processor restricts itself to generate a particular subset of the set of possible concepts. It completes in two stages. On the first stage, each processor generates complete concepts, but with restriction on the variable values in the same position. (Examine only a subset of the rows of the data set) In the second stage, each processor generates partial concepts but the variables can take any values (examine a subset of the columns) and this strategy is termed as parallelized sequential data mining algorithm.
3. Each processor makes on a partition of the data set (rows) and executes a sequential algorithm. So it builds entire concepts locally not globally and this strategy is termed as replicated sequential data mining algorithms.

Parallel Algorithms provide scalability to large data sets and hence improve response time. Its execution time depends on input size and also on the architecture of the parallel computer. All the algorithms proposed for parallel in shared nothing architecture can also be implemented for distributed architecture because many large databases are distributed in nature.

Apriori algorithms generate candidate item sets and scan databases as many times as the length of the longest frequent item sets whereas in FP-tree algorithms database scan only twice.

a) Existing Strategies

Research has been conducted by various researchers while dealing with large database. Out of which parallel system is found as one of a good strategy. Marfuz (2003) introduced two kinds of methods, parallel association rule mining (PARM) and distributed association rule mining (DARM). In, PARM divides a database into several local databases, and uses parallel multiprocessor shared-nothing environment to mine local databases [19]. The processors need to communicate with each other for the global counts. In case of DARM, association rules discovered from the geographically distributed data sets. The main drawback of DARM is the communication cost.

b) Our Strategies

In this paper, huge database divided into several small databases as per number of processors. The processors have access to only their local database. Processors can generate initial P tree local to their partition. The sequential process of database scanning is divided among the 'n' processors. In one scan, all the information about the transactions local to the processor store in their memory in the form of local P tree data structure and local counts of each item can be simultaneously calculated. Items which have the

Author α : Assistant Professor, Government Engineering College Ujjain, M.P., India. E-mail : nksharma070965@gmail.com
Author σ : Director, S.A.T.I. Engineering College Vidisha, M.P., India. E-mail : dr.jain.rc@gmail.com

support count more than the minimum support are to be included in the subsequent generation of FP tree and the support count stored at each processor is its local count. The local counts of each processor are summed up to get the global counts and the processor can simultaneously prune its infrequent items to get the frequent item set. No information lost is possible in generating FP tree as the FP tree generated at each site, if combined together will exactly replicate the global FP tree. In sequential algorithm, the conditional FP tree of each item present in the header table generates one after the other while in parallel mining, total items in the header table says m; can be divided among n total processors in the distributed environment. This division of work should be such that the amount of processing for generating the patterns at each processor is comparable. The division of items among the processors should be in such a way that the processor which gets the item least support count also gets the item with the highest support count. The items with lower support counts should be equally divided among the nodes. The next step is to calculate the global conditional pattern base of the item. If local conditional FP trees of an item are collected from all the processors and send to the destination processor, the global conditional FP tree can be generated. Hence the conditional FP tree can be sent in the form of conditional pattern base from which it is generated.

II. Work Already Done

Count Distribution (CD), Data Distribution (DD), Candidate Distribution, Intelligent data distribution (IDD) and Hybrid distribution (HD) algorithm briefly described [13] [14] [15] below:

a) Count Distribution (CD)

In which all processors generate the entire candidate set from Lk-1. Each processor can thus independently get partial support of the candidates from its local database partition and sum up to get global counts by exchanging local counts with other processors. Once global frequent sets have been determined, next candidate item sets can be determined in parallel at all processors. The focus is on minimizing communication. It does so even at the expense of carrying redundant computations in parallel. The aggregate memory of the system is not exploited effectively.

b) Data Distribution (DD)

Uses the total system memory by generating disjoint candidate sets on each processor. However, to generate the global support, each processor must scan the entire database in all iterations. In DD algorithm Contention is a major problem due to this processor may remain idle at the time of communication.

c) Candidate Distribution

Algorithm partitions the candidates during iteration l, so that each processor can generate disjoint candidates independent of other processors. The partitioning uses heuristics based on support, so that each processor gets an equal amount of work. The choice of the redistribution pass involves a tradeoff between decoupling processor dependence as soon as possible and waiting until sufficient load balance can be achieved. No local data send, only global values exchanged and data are received asynchronously and the processors do not wait for the complete pruning information to arrive from all the processors but repartitioning is expensive.

d) Intelligent Data Distribution (IDD)

The locally stored portions of the DB can be sent all the other PEs by using the linear-time ring-based all-to-all broadcast. Although DD divides the candidates equally among the processors, it fails to divide the work done on each transaction. IDD algorithms switch to CD once the candidates fit in the memory. Instead of a round-robin candidate partitioning, IDD performs a single-item, prefix based partitioning. Pseudo code for data movements using sending Buffer (SBuf) and receiving Buffer (RBuf).

Hybrid distribution (HD) algorithm combines CD and IDD. It partitions the P processors into G equal – sized groups, where each group is considered a super processor. Count Distribution is used among the G super processors, while the P/G processors in a group use Intelligent Data Distribution. The database is horizontally partitioned among the G super processors, and the candidates are partitioned among the P/G processors in a group.

The advantages of this algorithm are:

i. Provide good load balance.
ii. Handle much larger databases as compared to CD.
iii. Provide enough computation work by maintaining a minimum number of Candidates per PE cut down the amount of data movement to 1/G of the IDD.
iv. Keep processors busy, especially during later iterations.

e) FP-Growth Algorithm

FP stands for Frequent Patterns. This algorithm [6] makes use of an FP-tree structure, which is an extended prefix-tree structure for storing compressed, crucial information about frequent patterns. FP-Growth is an efficient method for mining complete set of frequent patterns by pattern fragment growth. Efficiency of mining is achieved with three techniques. First, a large database is compressed into a highly condensed, much smaller data structure, which avoids costly repeated database scans. Second, FP tree-based mining adopts a pattern fragment growth method to avoid the costly generation of a large number of

candidate sets. And third, a partitioning based, divide and conquer method is used to decompose mining task into a set of smaller tasks for mining confined patterns in conditional databases, which dramatically reduces the search space. FP-Growth is efficient for mining both long and short frequent patterns.

It avoids the costly candidate generation and it has better performance and efficiency than Apriori like algorithms but it takes two complete scans of the database and it uses a recursive routine to mine patterns from a conditional pattern base.

f) P-Tree Algorithm

A Pattern Tree [9] [10], unlike FP Tree, which contains the frequent items only, contains all the items that appear in the original database. We can obtain a P-tree through one scan of database and get the corresponding FP-tree from the P-tree later. An FP tree is a sub-tree of the P-tree with a specified support threshold, which contains those frequent items that meet this threshold and hereby excludes infrequent items. We do this by checking the frequency of each node along the path from root to leaves. It uses the same mining process as used by FP-Growth algorithm [6].

It scans the original database only once and in case support threshold changes, we need not re-scan the database but it uses a recursive mining process.

g) Inverted Matrix Algorithm

This association rule-mining algorithm is based on the conditional pattern concept [6]. The algorithm [11] [12] is divided into two main phases. The first one, considered pre-processing, requires two full I/O scans of the dataset and generates a data structure called Inverted Matrix. In the second phase, the Inverted Matrix is mined using different support levels to generate association rules. The mining process might take in some cases less than one-full I/O scan of the data structure in which only frequent items based on the support given by the user are scanned and participate in generating the frequent patterns. The Inverted Matrix layout combines the horizontal and vertical layouts with the purpose of making use of the best of the two approaches and reducing their drawbacks as much as possible. 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.

For computing frequencies, it relies first on reading sub-transactions for frequent items directly from the Inverted Matrix [6]. Then it builds independent relatively small trees for each frequent item in the transactional database. Each such tree is mined separately as soon as they are built, with minimizing the candidacy generation and without building conditional sub-trees recursively.

It uses a simple and non-recursive association rule mining process and the inverted matrix can be made disk resident, so it performs well for large data sets but it makes two scans of the original database and the complexity of developing an inverted matrix is a bit high.

III. Proposed Methodology

a) Mining Process

To find out frequent itemsets having minimum support and to generate strong rules having minimum confidence, Apriori algorithm is a bottom – up search based algorithm in which any subset of large item set must also be large.

b) Proposed Assumptions

- Horizontally partitioned database.
- Sites communication through message passing which reduce the number of messages passed and confine substantial amount of processing at local sites.

IV. Proposed Algorithm

- Select database and then arrange it in required proper format.
- Generate the initial P-Tree and find out the local counts of the items.
- Generate global counts of items by exchanging the local counts at each site.
- Generate a FP - Tree at each processor using its local data
- Distribute the frequent items such that comparison of computation required at each site is comparable
- Generate the pattern base for each item and start sending them to the corresponding processor
- Generate conditional FP-Trees using the data received from all the sites and those generated locally
- At the allocated processor mine frequent patterns for the corresponding items.

V. Implementation Example Database

To expound the effectiveness of the approach specified above a Student counseling data set (SCDS) has been considered, in which a separate agency has conducted an examination and on the basis of merit candidates have filled online choices of institutions and branches. The total number of combination of the choices was around 1200 and the total filled choices for institution were around 25 lakhs. To process such large data sequentially, it will roughly take 10 to 15 minutes. Hence for quick response it becomes necessary to use distributed architecture. We have taken Student counseling data set (SCDS), with three attributes, namely RollNo, Choice_Sequence and Branch_ID. Similarly Branch_ID consists of branch ID and college

name. Following Tables show Student Admission Data Set (SADS). Relational view of the same is being illustrated below.

Student Choice Data		
Roll No	Student Choice_ Sequence	Branch_ ID
1001	1	4
1002	2	2
1003	3	6
1004	4	11
1005	5	7
1006	6	9

Table 1 : Student Choice Data

Branch ID Generation		
Branch_ ID	College_Name	Branch
1	C1	CS
2	C1	IT
3	C1	EC
4	C2	CS
5	C2	IT

Table 2 : Branch ID Generation

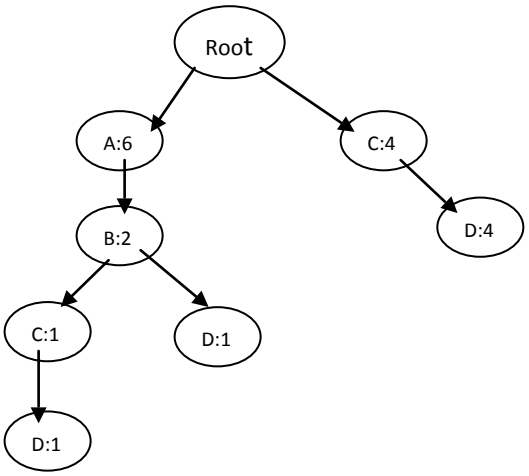
Above specified data is then treated with the proposed approach and transactional data set is obtained for the above given data which is expounded in Table 3.

Transactional Data Set				
Transaction_ID	A	B	C	D
T1	1	1	1	1
T2	1	0	1	1
T3	1	1	0	1
T4	1	0	1	1
T5	1	0	1	1
T6	1	0	1	1

Table 3 : Transactional Data Set for Student Choice Data of Table 1

VI. Tree Generated

On the basis of data mentioned in table 3, a tree is generated by the software



VII. Rules Generated and Result Analysis

a) Rules Generated

Rules are generated by taking minimum support is 75% and minimum confidence is 80%. The generated rules are shown in Table 4.

Rule	Confidence
C1 -> C3	83.33333
C3 -> C1	100
C1 -> C4	100
C4 -> C1	100
C3 -> C4	100
C4 -> C3	83.33333
C1, C3 -> C4	100
C4 -> C1, C3	83.33333
C1 -> C3 ,C4	83.33333
C3, C4 -> C1	100

Table 4 : Rules Generated

b) Result Analysis

Depending on the number of transactions the run time is calculated using Sequential FP–Tree algorithm as well as by using the proposed distributed approach and the comparison is expounded in Table 5.

S.No.	No. of Transactions	Run Time using Seq. FP -Tree (in seconds)	Run Time using Proposed Distributed approach (in seconds)
1	1000	5	6
2	5000	9	8
3	7500	16	11
4	10000	23	15

Table 5 : Result Analysis

Scale up shows that the proposed algorithms handles larger problem sets when more processors are available. Scales up experiments were performed where the size of the database was increased in direct proportion to the number of nodes in the system. The speedup gives decrease in the response time with the increase of number of processors. All the experiments are performed on a Xeon 2.8 GHz machine with 4 GB RAM running on Windows 7 platform. All the programs are written in JAVA platform.

VIII. Conclusion

The main focus of the paper is to design a mining algorithm for distributed environment using just a single database scan and distributing the computation work equally within the processors. The proposed algorithm allows efficient mining of patterns as 'n' items are considered for finding patterns simultaneously, where n is the number of processors. Computation time is reduced as the workload is divided amongst various processors. A processor receives the patterns from each processors one after the other and if one machine is slow other machines are also effected. Hence the process should be improved by the simultaneous construction of conditional pattern tree along with the improvement in the receive operation.

Acknowledgements

Our sincere thanks to our colleague Mr. Manoj Yadav, Software Consultant in Bhopal, Madhya Pradesh, India for the immense support you have provided us for publishing this paper.

References Références Referencias

1. R. Agrawal, T. Imienski and A. Swamy, "Database Mining: A Performance Perspective, IEEE Tran. On Knowledge and Data Engg." December, 1991.
2. R. Agrawal, T. Imielinski, and A. Swami, "Mining association rules between sets of items in large databases", In Proc. of the ACM SIGMOD Conference on Management of Data, Washington, D.C., May 1993.
3. Margaret H. Dunham, Yongqiao Xiao, Southern Methodist University, Dallas, Texas and Le Gruenwald, Zahid Hossain, University of Oklahoma, Norman, UK, "A Survey of Association Rules".
4. Jong Soo Park, Ming-Syan Chenand Philip S. Yu, "An effective hash-based algorithm for mining association rules," In Proceedings of 1995 ACM-SiGMOID international Conference on Management of Data, 1995.
5. Mohammed J. Zaki, Rensselaer Polytechnic Institute, "Association Mining: A Survey", IEEE Concurrency, 1999.
6. Jiawei Han, Jian Pei, and Yiwen Yin, "Mining frequent patterns without candidate generation", Technical Report CMPT99-12, School of Computing Science, Simon Fraser University, 1999.
7. Mohammad El-Hajj and Osmar R. Zaiane, "Parallel Association Rule Mining with Minimum Inter-process Communication", In Proc. of IEEE Int. Conf. On Database and Expert Systems Applications, 2003.
8. R. Agarwal and J. Shafer, "Parallel Mining of Association Rules," IEEE Transactions on Knowledge and Data Engineering, Vol. 8, No. 6, December 1996.
9. D. W. Cheung, V.T. Ng, A.W. Fu and Y Fu, "Efficient Mining of Association Rules in Distributed Databases", IEEE Tran. On Knowledge and Data Engg. December, 96.
10. A.Y. Zomya, T.E. Ghazawi and O. Frieder, "Parallel and Distributed Computing for Data Mining", IEEE Concurrency, Oct./Nov. 1999.
11. Skillicorn, "Strategies for Parallel Data Mining", IEEE Concurrency, Nov. 1999.
12. A. Mueller, "Fast and Sequential Algorithms for Association Rule Mining." A comparison, Tech Report CS-TR-3515, Univ. of Maryland, College Park. Md. 1995.
13. Margaret H. Dunham and Yongqiao Xiao, Southern Methodist University, Dallas, Texas and Le Gruenwald, Zahid Hossain, University of Oklahoma, Norman UK, " A survey of Association Rules."
14. Mohammed J. Zaki, Rensselaer Polytechnic Institute, "Parallel and Distributed Association Mining: A Survey", IEEE Concurrency 1999.
15. E.H. Han, G. Karypis, and V. Kumar, "Scalable Parallel Data Mining for Association Rules," IEEE Transactions on Knowledge and Data Engineering, Vol. 12, No. 3, May/June 2000.
16. T. Shimomura and S. Shibusawa, "Performance Evaluation of Distributed Algorithms for Mining Association Rules on Workstation Cluster", IEEE 2000.
17. Hao Huang, Xindong Wu and Richard Relue, "Association Analysis with One Scan of Databases", IEEE 2002.
18. O.R. Zaiane, M.E. Hajj, "Parallel Association Rule Mining with Minimum Inter-Processor Communication", 2003 IEEE, 14th International Workshop on Database and Expert Systems Applications.
19. Fan Wu, Ya-Han Hu, Tz Ke Wu, "A Novel and Efficient Distributed Data Mining Algorithm Based on Frequent Pattern-Tree", Int'l Conf. Data Mining | DMIN'09 |.

This page is intentionally left blank