Generate Frequent Item Set in Serial and Parallel Approach

Rinku Kadam[1] Renuka Gawade[2] Dhanashree Kumbhar[3] Dipali Londhe[4]
[1,2,3,4]Department of Computer Science & Engineering
[1,2,3,4]P K Technical Campus, Pune, Maharashtra, India

Abstract—Apriori Algorithms are used on very huge data sets with high dimensionality. Therefore parallel computing can be used for mining of association rules. The process of association rule mining consists of finding frequent itemsets and forming rules from the frequent itemsets. Finding frequent item sets is more expensive in terms of power and resources utilization. Thus majority of parallel apriori algorithms focus on parallelizing the process of frequent itemset formation. The computation of frequent itemsets mainly consists of creating the candidate keys and counting them. The parallel frequent item sets mining algorithms addresses the issue of distributing the candidate keys among processors such that their counting and creation is effectively parallelized. This paper presents comparative analysis of these algorithms.

Key words: Parallel Data Mining, Frequent Item Sets, Association Rules, Apriori Algorithm
General Terms: Data mining, parallel processing, apriori algorithm.

I. INTRODUCTION

Accumulation of abundant data from different sources of information but little amount of knowledge situation has led to knowledge discovery from databases which are called data mining. Data mining use the existing data and retrieve the useful information from it which is not directly visible in the huge data. As data mining algorithms deal with large data, the primary concerns are storage of data in the main memory at run time and how to improve the run time performance. Sequential algorithms cannot provide scalability, like data size, or runtime performance, for such huge databases. Because the data sizes are increasing to a large amount, high-performance parallel and distributed computing is used to get the advantage of more than one processor to handle this huge amount of data.

Data mining deals with large amount of data to extract the useful information. Association Rule Mining (ARM) or frequent item set mining is an important functionality of data mining. The apriori algorithm is one of the efficient algorithms for finding frequent item sets from a transaction database. As data mining mainly deals with large amount of data, the main concern is how to improve the performance of the algorithm. One way of increasing the performance of apriori is parallelizing the process of generating frequent item sets. The rest of the paper is implemented as follows. In Section 2 the related work is over viewed. In Section 3 the basic concepts of association rule mining are study and apriori algorithm is described. In Section 4 comparative analysis of the apriori algorithms is stated. In Section 5 conclusion is stated.

II. RELATED WORK

Parallel ARM algorithm represents transactions using either row data format or column data format [4][7]. In row data format, data is represented as transaction ID versus items sold in each transaction whereas in column data format, data is represented as each item versus transaction id in which the item is retail. The parallel ARM algorithms are classified as data set partitioning algorithms, Breadth-First Algorithms, Depth-First Algorithms, Sampling Algorithms and Incremental Update Algorithms [2][3]. There are several parallel association rule mining algorithms are proposed that based on data set partitioning like Count Distribution, Data Distribution, Candidate key Distribution, Common Candidate Partitioned, Éclat, Parallel Partition [1][5][9][10]. There are various surveys accomplish on these algorithms [2][6][7][8].

III. ASSOCIATION RULE MINING

A. Basic Concept

The association rule mining is generated from the market basket analysis. Let database D be the transaction database which consists of the transaction records {T1, T2... Tn} of the customers buying habits. Each transaction T consists of the items purchased by the customers in one or more visit of the market. The items are the subset of the set of items I {I1, I2... Im} in the market that are considered for analysis purpose. An item set consists of some combination of items that occur jointly or a single item from items. Association rule mining X->Y, represents the dependency relationship between two different itemsets X and Y in the database. Whenever X is come in any transaction, there is a probability that Y may also occur in the same transaction is said to be relationship. This occurrence is based on following two interesting measures.

Support: represents the percentage of transactions that contain X U Y and it is given by

Support$(X->Y) = P(X \cup Y)$

Confidence: It gives the percentage of transactions containing X that also contain Y and it is given by

Confidence$(X->Y) = P(X \cup Y)/P(X)$

B. Apriori Algorithm

Apriori algorithm is the more classical mining algorithm. It is based on the apriori principle that states all the nonempty subsets of a frequent item set must also be frequent. It is a two-step process.

1) Step 1: The Prune Step

It scans the whole database to find the count of each candidate in Ck where Ck represents candidate k- itemset. The count of each item set in Ck is compared with earlier defined minimum support count known as threshold to find whether that item set can be put in frequent k-item set Lk.

2) Step 2: The Join Step

Lk is joined with itself to generate the next candidate k+1 item set Ck+1. The major step here is the prune step that requires scanning the entire database for searching the count of each item set in every candidate k item set. If the database is large then it requires more time to find all the frequent item sets in the database.

IV. PARALLEL APRIORI ALGORITHMS

A. The Count Distribution Algorithm [10]

Each processor generates the partly support of all candidate item sets from its local database partition in parallel. At the end of every iteration the global supports are generated by exchanging the existing only in the part of support counts among all the processors. All the processors generate the whole candidate from Lk-1. Each processor thus independently calculates the partial supports of the candidates from its local database partition. Then each processor changes its local counts of Ck with all the other processors to generate the global Ck counts. Each processor then computes Lk from Ck. Once the global Lk has been decided, each processor builds Ck+1 in parallel manner and repeats the process until all frequent item sets are found.

1) Advantages:

It reduces the communication cost as only counts are interchanged among the processors and speed up the aggregation process as only vector aggregation is to be carried out rather than matching the candidates first and then finding their sum.

2) Disadvantages:

It does not use the total memory of the system efficiently as the entire hash tree is replicated on each processor.

B. The Data Distribution Algorithm [10]

It generates the frequent 1-item set by using count distribution algorithm. It then separate the candidate sets into disjoint sets which are assigned to different processors. Each processor compute the support counts for the item sets in its local candidates by scanning the local partition and the remote partition to generate the local frequent item sets in all iterations. At the end of each iteration, processors exchange local frequent item sets with the other processors so that each processor has the complete Lk for generating Ck+1.

1) Advantages:

It uses the total system memory efficiently by generating disjoint candidate key sets on each processor. The aggregation need not be carried out since the local frequent item sets are disjoint.

2) Disadvantages:

It suffers from huge transmission cost.

C. The Candidate Distribution Algorithm [10]

For the initial passes it uses either Count Distribution or Data Distribution algorithm. In some pass I which is heuristically decided, this algorithm divides the frequent itemsets Lk-1 among the processors in this way each processor can produces unique candidate sets independent of the other processors. Selective data replicated so that each processor can calculate the count of the candidate set independent of other processor.

1) Advantages:

It eliminates the processor dependence so that without synchronizing, the processors can proceed independently at the end of each pass.

2) Disadvantages:

It is subjected from huge communication cost of redistributing the database and scan the local partitions repeatedly. The communication gains of independent processing are not sufficient to offset the database redistribution cost.

D. The Common Candidate Partitioned Algorithm (CCPD) [5]

It is same as the count distribution algorithm. It uses shared memory architecture. Each processor produces the candidate item sets in parallel and stores them in a hash structure which is shared among every processor. Each processor scans its local partition to calculate the support counts of the candidates and automatically updates the counts of the candidates in the shared hash structure.

1) Advantages:

There is no need for the processor to exchange the count and carry out the summation to obtain the global count of the candidate.

2) Disadvantages*:*

The complex hash structure is used which incur additional overhead of maintaining and searching as it is poor cache locality.

E. Fork-Join Parallelism

Initially process begins as a single process: master thread. We can make some part of the program to work in parallel by creating multiple child threads. Until the parallel region construct is encountered, master thread can execute in serial mode. Master thread initialize a team of parallel child threads (fork) that concurrently execute statements in the parallel region. The work sharing construct divides the work among the every thread. After executing the statements in the parallel region, team threads synchronize and enumerate (join) but master continues (Figure 1).

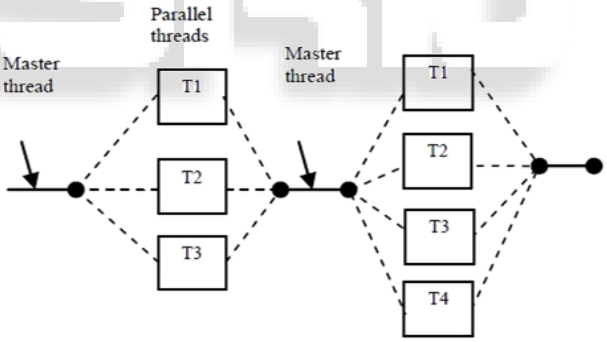

Fig. 1: Fork-Join Parallelism

F. Multi core

Multi core refers two or more processors. But they differ from independent parallel processors as they all are integrated on the same chip circuit [7],[8].A multi core processor implement methods like message passing or shared memory inter core communication for multiprocessing. If the number of threads are less than or equal to the number of cores, different cores are allocated to each thread and threads run independently on multiple cores. (Figure 2) If the numbers of cores are not equal to the number of threads, the cores are shared among the threads. Any application that can be threaded, this thread can be mapped efficiently to multi-core, but the improvement in performance gained by the use of multi core processors depends on the fragment of the program that can be parallelized [Amdahl's law] [10].

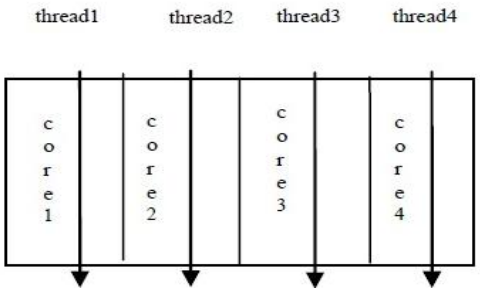

Fig. 2: Independent threads on the cores

G. Parallel Partition Algorithm [1]

It presents the parallelization of the sequential partition algorithm. It uses client server architecture. The coordinator act as server and the processor act as client. It consists of four phases.

1) Phase 1: Each processor scans its local partition and produces local frequent itemsets and sends it to the coordinator.
2) Phase 2: After the phase 1, it receives the local frequent itemsets from all the processors; the coordinator takes the union of all these local frequent itemsets to produces global candidates. The coordinator sends these global candidates to all the processors. At the end of the phase, all the processors have identical candidate itemsets.
3) Phase 3: Each processor calculates the local support counts of the global candidates and sends them to the coordinator.
4) Phase 4: The coordinator generates the global frequent itemsets by taking the summation of the local support counts received from the processors.

1) Advantages:

It needs to synchronize in only two phases. Since it uses vertical database layout, simple intersections are used which speed up the counting process.

2) Disadvantages:

It transfers frequent itemsets, global candidates and counts in three phases and it does not generate accurate frequent itemsets.

Algorithm	Number of Messages Exchanged	Type of messages exchanged	Synchronization Required	Database Layout	Architecture
Count Distribution Algorithm	n(n-1) (in each pass)	local counts (in each pass)	Yes (after each pass)	Horizontal	Shared-Nothing
Data Distribution Algorithm	n(n-1) (in each pass)	local frequent itemsets (in each pass)	Yes (after each pass)	Horizontal	Shared-Nothing
Candidate Distribution Algorithm	n(n-1) (in initial passes)	local frequent itemsets for initial passes and database is repartitioned	No	Horizontal	Shared-Nothing
Common Candidate Partitioned Database Algorithm	none	not applicable	Yes (after each pass)	Horizontal	Shared-Memory
Equivalence Class Transformation Algorithm	n(n-1) (in two phases)	local counts in initialization phase and tid lists in transformation phase and database is repartitioned	No	Horizontal and Vertical	Shared-Nothing
Parallel Partition Algorithm	3n (in three phases)	local frequent itemsets in phase 1, global candidates in phase 2 and local counts in phase 3	Yes (in phase 2 and phase 4)	Vertical	Client-Server

Table 1. Comparative analysis of parallel apriori algorithms

V. Conclusion

The performance of the parallel apriori algorithms depends on the processing time and the data communication cost. The data communication cost can be reduced by using client- server architecture like Parallel Partitioning Algorithm and exchanging only the counts as in Count Distribution Algorithm. The processing time depends on the database layout, number of times the database is scanned and the size of the candidates generated. Vertical database layout speeds up the searching process as demonstrated in the Apriori Algorithm and reduces the database scanning time. Thus a parallel apriori algorithm using client-server architecture with only counts exchanged and using vertical database layout can achieve balanced trade-off between the processing time and the data communication cost and using multicore processing power we can easily reduce overhead of mining process.

References

[1] Khadidja Belbachir, Hafida Belbachir, "The Parallelization of Algorithm Based on Partition Principle for Association Rules Discovery", In Proceedings of International Conference on Multimedia Computing and Systems(ICMCS), IEEE, May 2012.
[2] Ruowu Zhong, Huiping Wang, "Research of Commonly Used Association Rules Mining Algorithm in Data Mining", In Proceedings of International Conference on Internet Computing and Information Services (ICICIS), IEEE, September 2011.

[3] V.Umarani, Dr.M.Punithavalli, "A Study On Effective Mining Of Association Rules From Huge Databases", International Journal of Computer Science and Research (IJCR), Vol. 1 Issue 1, 2010.
[4] Xindong Wu , Vipin Kumar, J. Ross Quinlan, Joydeep Ghosh, Qiang Yang ,Hiroshi Motoda, "Top 10 Algorithms in Data Mining", Knowledge and Information Systems, Volume 14, Issue 1, pp 1-37, Springer, January 2008.
[5] Mohammed J. Zaki, Srinivasan Parthasarathy, Mitsunori Ogihara, Wei Li, "Parallel Data Mining for Association Rules on Shared-Memory Systems", Data Mining and Knowledge Discovery, Springer, 2001.
[6] Eui-Hong (Sam) Han, George Karypis, Vipin Kumar, "Scalable Parallel Data Mining for Association Rules", IEEE Transactions on Knowledge and Data Engineering, Volume: 12, Issue: 3, May/June 2000.
[7] Mohammed J. Zaki, "Parallel and Distributed Association Mining: A Survey", IEEE Concurrency, Vol 7, Issue 4, pp 14-25, October 1999.
[8] Mohammed J. Zaki, Srinivasan Parthasarathy, Mitsunori Ogihara, Wei Li, "Parallel Algorithms for Discovery of Association Rules", Data Mining and Knowledge Discovery, Vol 1, Issue 4, pp 343-373, Springer, December 1997.
[9] Mohammed J. Zaki, Srinivasan Parthasarathy, Mitsunori Ogihara, Wei Li, "A Localized Algorithm for Parallel Association Mining", Proceedings of the ninth annual ACM symposium on Parallel algorithms and architectures, ACM 1997.
[10] Rakesh Agrawal, John C. Shafer, "Parallel Mining of Association Rules", IEEE Transactions on Knowledge and Data Engineering, December 1996.

