Comparative Study of Apriori Algorithms for Parallel Mining of Frequent Itemsets

Avani M. Sakhapara
Assistant Professor
K. J. Somaiya College of Engineering
Mumbai University

Bharathi H. N.
Associate Professor
K. J. Somaiya College of Engineering
Mumbai University

ABSTRACT
Apriori Algorithms are used on very large data sets with high dimensionality. Therefore parallel computing can be applied for mining of association rules. The process of association rule mining consists of finding frequent item sets and generating rules from the frequent item sets. Finding frequent itemsets is more expensive in terms of CPU power and computing resources utilization. Thus majority of parallel apriori algorithms focus on parallelizing the process of frequent item set discovery. The computation of frequent item sets mainly consist of creating the candidates and counting them. The parallel frequent itemsets mining algorithms addresses the issue of distributing the candidates among processors such that their counting and creation is effectively parallelized. This paper presents comparative study of these algorithms.

General Terms
Data mining, algorithms, parallel processing, apriori

Keywords
Parallel data mining, frequent itemsets, association rules, apriori algorithm

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

Data mining deals with large volumes of data to extract the useful knowledge. Association Rule Mining (ARM) or frequent itemset mining is an important functionality of data mining. The apriori algorithm is one of the best algorithms for finding frequent itemsets from a transaction database. As data mining mainly deals with large volumes of data, the main concern is how to improve the performance of the algorithm. One way of improving the performance of apriori is parallelizing the process of generating frequent itemsets.

The rest of the paper is organized as follows. In Section 2 the related work is overviewed. In Section 3 the basic concepts of association rule mining are discussed and apriori algorithm is described. In Section 4 a comparative analysis of the parallel apriori algorithms is given. In Section 5 conclusion is given.

2. RELATED WORK
Many parallel ARM algorithms have been proposed which represent transactions using either horizontal data format or vertical data format [4,7]. In horizontal data format, data is represented as transaction ID versus items sold in each transaction whereas in vertical data format, data is represented as each item versus transaction ids in which the item is sold.
The parallel ARM algorithms are broadly 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 proposed based on data set partitioning like Count Distribution, Data Distribution, Candidate Distribution, Common Candidate Partitioned, Eclat, Parallel Partition [1,5,9,10]. There are various surveys carried out on these algorithms[2,6,7,8].

3. ASSOCIATION RULE MINING
3.1 Basic Concept
The basic concept of association rule mining is originated from the market basket analysis. Let D be the transaction database which consists of the transaction records {T1,T2,…,Tn} of the customers. Each transaction T consists of the items purchased by the customers in one visit of the super market. The items are the subset of the set of whole items I {I1, I2,…., Im} in the super market that are considered for analysis. An itemset consists of some combination of items which occur together or a single item from I. Association rule mining X->Y, represents the dependency relationship between two different itemsets X and Y in the database. The relationship is whenever X is occurring in any transaction, there is a probability that Y may also occur in the same transaction. This occurrence is based on two interesting measures.

Support: It represents the percentage of transactions in D that contain $X \cup Y$ and it is given by

support(X->Y) = $P(X \cup Y)$.

Confidence: It gives the percentage of transactions in D containing X that also contain Y and it is given as confidence(X->Y) = $P(Y/X)$.

3.2 Apriori Algorithm

Apriori algorithm is the most classical association rule mining algorithm. It is based on the apriori principle that all the nonempty subsets of a frequent itemset must be frequent. It is a two step process.

Step 1: The prune step
It scans the entire database to find the count of each candidate in C_k where C_k represents candidate k-itemset. The count of each itemset in C_k is compared with a predefined minimum support count to find whether that itemset can be placed in frequent k-itemset L_k.

Step 2: The join step
L_k is natural joined with itself to generate the next candidate k+1-itemset C_{k+1}. The major step here is the prune step which requires scanning the entire database for finding the count of each itemset in every candidate k-itemset. If the database is huge then it requires more time to find all the frequent itemsets in the database.

4. PARALLEL APRIORI ALGORITHMS

4.1 Count Distribution Algorithm [10]

Each processor generates the partial support of all candidate itemsets from its local database partition in parallel. At the end of each iteration the global supports are generated by exchanging the partial support counts among all the processors. All the processors generate the entire candidate from L_{k-1}. Each processor thus independently compute the partial supports of the candidates from its local database partition. Then each processor exchanges its local counts of C_k with all the other processors to generate the global C_k counts. Each processor then computes L_k from C_k. Once the global L_k has been determined, each processor builds C_{k+1} in parallel and repeats the process until all frequent itemsets are found.

Advantages:
It minimizes the communication cost as only counts are exchanged among the processors and speeds up the summation process as only vector summation is to be carried out rather than matching the candidates first and then finding their sum.

Disadvantages:
Since the entire hash tree is replicated on each processor, it does not utilize the total memory of the system efficiently.

4.2 Data Distribution Algorithm [10]

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

Advantages:
It utilizes the total system memory efficiently by generating disjoint candidate sets on each processor. The summation need not be carried out since the local frequent itemsets are disjoint.

Disadvantages:
It suffers from huge communication cost.

4.3 Candidate Distribution Algorithm [10]

For the initial passes it uses either Count Distribution or Data Distribution algorithm. Then in some pass I which is heuristically determined, this algorithm divides the frequent itemsets L_{k-1} among the processors in such a way that each processor can generate unique candidate sets independent of the other processors. Data is selectively replicated so that each processor can calculate the counts of the candidate sets independent of other processors.

Advantages:
It removes the processor dependence so that the processors can proceed independently without synchronizing at the end of each pass.

Disadvantages:
It suffers from huge communication cost of redistributing the database and scans the local partitions repeatedly. The communication gains of independent processing are not sufficient to offset the database redistribution cost.

4.4 Common Candidate Partitioned Algorithm(CCPD) [5]

It is same as the count distribution algorithm. It uses shared memory architecture. Each processor generates the candidate itemsets in parallel and stores them in a hash structure which is shared among all the processors. Each processor scans its local partition to calculate the support counts of the candidates and atomically updates the counts of the candidates in the shared hash structure.

Advantages:
There is no need for the processors to exchange the counts and carry out the summation to obtain the global counts of the candidates.

Disadvantages:
It uses complex hash structures which incur additional overhead of maintaining and searching as it is poor cache locality.

4.5 Equivalence Class Transformation Algorithm(Eclat) [9]

It has four phases namely the initialization phase, transformation phase, asynchronous phase and the final reduction phase.

Phase 1: Initialization phase
Each processor computes the frequent 2-itemsets L_2. Each processor scans the local partition and generates frequent 2-itemsets L_2. Each processor then communicates the support counts of L_2 to the other processors to construct global frequent 2-itemsets.

Phase 2: Transformation phase
L_2 is partitioned using equivalence class partitioning and these partitions are then assigned to the processors. Each processor scans its local database and constructs partial tid lists for frequent 2-itemsets. Each processor sends the tid lists for the itemsets belonging to the other processors so that each processor can construct the global tid lists for

the itemsets in its equivalence classes. At the end of this phase the database is redistributed.

Phase 3: Asynchronous phase
Each processor independently computes the frequent itemsets eliminating the need of synchronization among the processors.

Phase 4: Final reduction phase
The results are accumulated from each processor.

Advantages:
It groups the related frequent itemsets together using the equivalence class partitioning. It utilizes the total memory of the system by partitioning the candidates into disjoint sets using equivalence class partitioning. Since the processors are decoupled right in the beginning, no synchronization is required. It uses vertical database layout which facilitates fast counting using simple intersections. Each processor computes all the frequent itemsets from one equivalence class before proceeding to the next. It does not pay the cost of building and searching complex data structures. Since it uses vertical database layout the local partitions are scanned only once.

Disadvantages:
It needs to generate and redistribute the vertical tid lists whose size is comparable to the original database. It may need roughly twice the disk space as the algorithm uses horizontal layout for the initial phase and vertical layout thereafter.

4.6 Parallel Partition Algorithm [1]

It presents the parallelization of the sequential partition algorithm. It uses client server architecture. The coordinator acts as server and the processors act as clients. It consist of four phases.

Phase 1: Each processor scans its local partition and generates local frequent itemsets and sends it to the coordinator.

Phase 2: After receiving the local frequent itemsets from all the processors, the coordinator takes the union of all these local frequent itemsets to generate global candidates. The coordinator sends these global candidates to all the processors. At the end of this phase, all the processors have identical candidate itemsets.

Phase 3: Each processor computes the local support counts of the global candidates and sends them to the coordinator.

Phase 4: The coordinator generates the global frequent itemsets by taking the summation of the local support counts received from the processors.

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.

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

Table 1. Comparative analysis of parallel apriori algorithms

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

5. 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 Eclat 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.

6. 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.