Enhancement in Performance of Processor Using Apriori Algorithm For Multicore System

Prantik Pancholi, Shital Khairnar, Jyoti Kamble, Prof. Amol Jadhao

Abstract— **Apriori Algorithm are the most popular algorithm of Association Rule Mining . Association rule mining is an important topic in data mining field.This algorithm is applied on very large data sets with high dimensionality. Therefore parallel computing can be apply on mining of association rules. The process of association rule mining consists of find frequent item sets and producing rules from the frequent item sets. Association Rule Mining is an area of the data mining that focuses on pruning candidate keys. This algorithm somehow has limitation and thus, giving the opportunity to do this research. Increased availability of the MultiCore processors is forcing us to re-design algorithms and applications so as to exploit the computational power from multiple cores finding frequent itemsets is more expensive in terms of computing resources utilization and CPU power. Thus majority of parallel apriori algorithms effect on parallelizing the process of frequent item set find. The parallel frequent itemsets mining algorithms addresses the issue of distributing the candidates among processors. Efficient algorithm to discover frequent patterns is important in data mining research.This paper represents the comparative study of these algorithms and study the benefit of memory mapping on Multi-Core processors.**

Index Terms— **Parallel data mining, frequent itemsets, Association rule mining, Apriori Algorithm.**

I. INTRODUCTION

Accumulation of big data from different sources of the society but a little knowledge situation has lead to knowledge find from databases which is called data mining.The data sources can include data warehouses, databases the Web,other information repositories that are streamed into the system dynamically. If it mines all the interesting patterns then a data mining algorithm is complete. It is often unrealistic and inefficient for data mining systems to generate all possible patterns. In the recent years, many network and other applications, which demand huge I/O overhead, are reported to be using a special I/O(Input/Output) feature known as mmap () to improve their performance. Data mining techniques are use the existing the data and retrieve the useful information from it which is not directly seeable in the original data. As data mining algorithms are deals with the large data, the primary concerns are how to improve the run time performance and how put the data in the main memory at run time . Sequential algorithms cannot support the scalability, in terms of the data dimension, size, runtime performance, for such large databases. Because the data sizes are increasing to a huge quantity, high-performance ditributed computing and parallel computing is used to get the benefits of more than one processor to handle these large quantities of data. multicore processor technology has improved dramatically as chip manufacturers are try to boost performance while minimizing power consumption.The apriori algorithm are one of the best algorithms for finding frequent itemsets from a transaction database. As the data mining mainly deals with the large volumes of data, the main concern is how to increase the performance of the algorithm. As multicore processors continue to scale, they supply the opportunity for performing more complicated and computation-intensive tasks in real-time. However, to take full advantage of multicore processing, these systems must exploit intra-task parallelism,One way of improving the performance of the apriori is parallelizing the process of producing frequent itemsets.

The rest of the paper is organized as follows. The related work is overviewed in Section II .The association rule mining and the basic concepts of apriori algorithm are described and discussed in Section III. A comparative analysis of the parallel apriori algorithms is given n Section IV .Conclusion is given in Section V.

II. RELATED WORK

Most of parallel ARM algorithms have been proposed which represent transactions using either vertical data format[4],[7] or horizontal data format . In horizontal data format, data is represented as transaction ID versus items sold in this transaction where as in vertical data format, data is represented as each item versus transaction ids in which the items are sold. The parallel ARM algorithms are classified as data set partitioning Algorithms, Depth-First Algorithms,Bredth-First Algorithms, Sampling

Manuscript received September, 2015.

Prantik Pancholi *Computer Engineering,Savitribai Phule Pune University / D.Y.P.I.E.T/ pune, India, / 9096984494*
Shital Khairnar*, Computer Engineering , Savitribai Phule Pune University / D.Y.P.I.E.T/ pune, India, / 9970913074*
Jyoti kamble*, Computer Engineering, Savitribai Phule pune University/ D.Y.P.I.E.T /pune, India, / 7387412394*
Amol Jadhao,Copmuter Engineering, Savitribai Phule Pune University/ D.Y.P.I.E.T/ pune,India

Algorithms, Incremental Update Algorithms[2],[3]. There are various parallel association rule mining algorithms proposed depends on data set of partitioning like Data Distribution, Count Distribution, Candidate Distribution, Common Candidate Partitioned, Parallel Partition [1],[5],[9],[10]. There are various surveys carried out on these algorithms[2],[6],[7],[8].

III. ASSOCIATION RULE MINING

A. Basic concept

Association rule mining are the rules which are originated from the UCOI repository dataset i.e. market basket analysis. Let D be the transaction of the database which consists of the transaction records {Ti1,Ti2,....,Tin} of the customers. Each transaction Ti consists of the items which is purchased by the customers in first visit of the super market. The items are the subset of the set of whole items Item {Im1, Im2,...., Imn} in the super market that are considered for analysis. A itemset consists of some combination of items which may occur a single or together item from Im. Association rule mining P->Q, represents the dependency relationship between two different itemsets P and Q in the database. The relationship is whenever P is occurring in any transaction, there is probability that Q may also occur in the same transaction. This occurrence is depends upon two interesting measures.

Support: It is used to represents the percentage of transactions in D that contain P U Q and it is given by

support(P->Q) = *P*(P U Q).

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

B. Apriori Algorithm

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

Step 1: The prune step

It scans the whole database to find the count of each candidate in Ck where it represents candidate k- itemset. The count of each itemset in Ck is compared with a predefined minimum support count to finds whether that itemset can be placed in frequent k-itemset L_k.

Step 2: The join(connect) step

L_k is natural joined with itself to produce the next candidate k+1-itemset C_{k+1}. The long step here is the prune step which requires scanning the whole database for finding the count of this itemset in candidate k-itemset. If the database is large then it requires long time to find all the frequent itemsets in the database.

Example: A database consist of five transactions. Let the min support = 50% and min confidence f = 80%. As it shows the transaction in Figure I Step 1: In this step find all Frequent Itemsets, as shown in Figure I Frequent Itemsets:
{A},{B},{C},{E},{A,C},{B,C},{B,E},{C,E},{B,C,E}
Step 2: produce strong association rules from the frequent itemsets.[2]

Results are shown in following table

Table(I)

TID	ITEMS
100	ACD
200	BCE
300	ABCE
400	BE

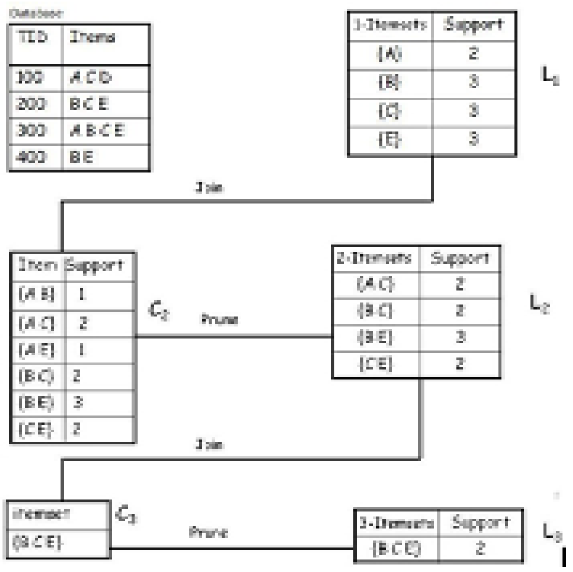

Fig I frequent itemsets

TABLE II SUPPORT & CONFIDENCE

Rules	Support (PQ)	Support (P)	Confidance
{A}->{C}	2	2	100
{B}->{C}	2	3	66.66
{B}->{E}	3	3	100
{C}->{E}	2	3	66.66
{B}->{C E}	2	3	66.66
{C}->{B E}	2	3	66.66
{E}->{B C}	2	3	66.66
{C}->{A}	2	3	66.66
{C}->{B}	2	3	66.66
{E}->{B}	3	3	100
{E}->{C}	2	3	66.66
{C E}-> {B}	2	2	100
{B E}-> {C}	2	3	66.66
{B C}-> {E}	2	2	100

This algorithm is an approach based on two-stage frequency, Find all the item sets with the support greater than the minimum support, which is called frequent item;

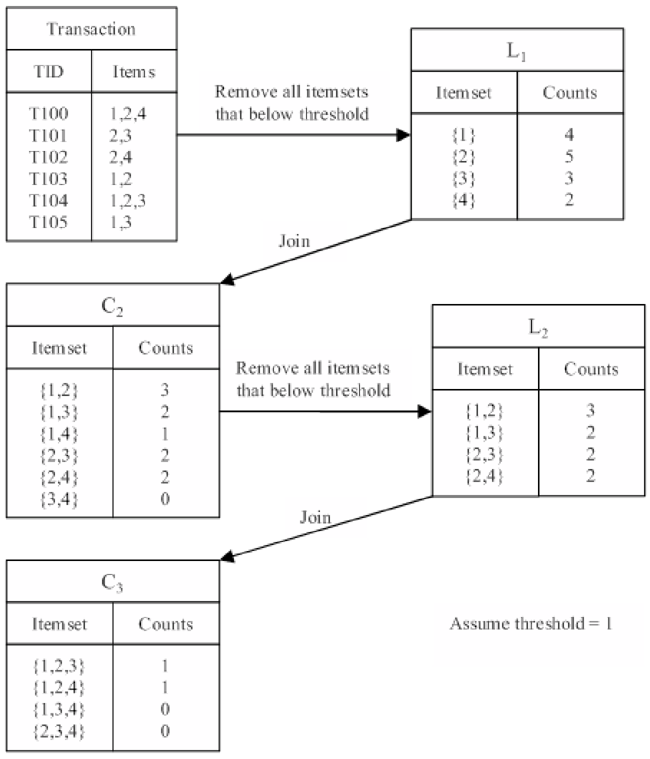

Fig II: Working of Apriori Algorithm.

(2) Based on the above obtained frequent set, all the association rules will be produced, and for each frequent item set A, all the nonempty subset a of A will be found, Apriori algorithm called two sub-processes.

IV. PARALLEL APRIORI ALGORITHMS

A. Count Distribution Algorithm [10]

Each processor produces the partial support of all candidate itemsets which is from its local database partition in parallel. At the end of this iteration the global supports are produced by exchanging the partial support counts between all the processors. All the processors generate the whole candidate from L_{k-1}. Each processor thus independently counts the partial supports of the candidates from its local database partition. Then processor exchanges its local counts of Ck with all the other processors to produce the global Ck counts. Each processor then computes L_k from C_k. Once the global Lk has been determined, each processor builds Ck+1 in parallel and repeats the process until all of the frequent item sets are found

Advantages:

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

Disadvantages:

Because the whole hash tree is replicated on each processor, it doesn't utilize the total memory of the system efficiently.

B. Data Distribution Algorithm [10]

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

Advantages:

It utilizes the complete system memory efficiently by producing disjoint candidate sets on each processor. The summation need not be carried out because the local frequent itemsets are disjoint.

Disadvantages:

It suffers from huge communication cost.

C. Candidate Distribution Algorithm [10]

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

Advantages:

It deletes the processor dependence so that the processors can proceed independently without synchronizing at end of this pass.

Disadvantages:

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

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

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

Advantages:

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

Disadvantages:

It uses complicated hash structures which incur additional overhead of maintaining and searching as poor cache locality

*E. **Fork-Join Parallelism***

Initially programs begin as a single process: master thread. We can make some part of the program to work in parallel by creating child threads. Master thread executes in serial mode until the parallel region construct is encountered. Master thread creates a team of parallel child threads (fork) that simultaneously execute statements in the parallel region. The work sharing construct divides the work among all the threads. After executing statements in the parallel region, team threads synchronize and enumerate (join) but master continues (Figure I)

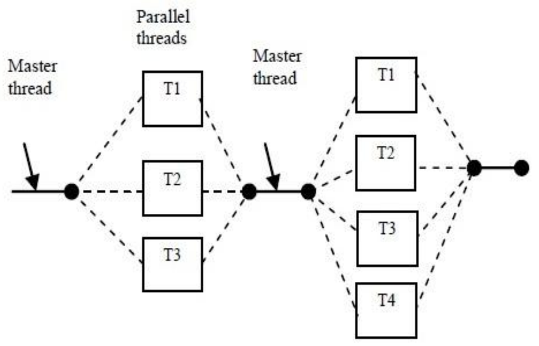

Fig III: Fork-Join Parallelism

F.Multi core

Multicore refers two or more processors. But they are different from independent parallel processors as they are integrate on the same chip circuit [7],[8].A multi core processor implement message passing or shared memory intercore communication methods for multiprocessing.If the number of threads are equal to or less than the number of cores, separate core is allocated to each thread and threads are run independently on multiple cores. (Figure IV) If number of threads are more than the number of cores, the cores are shared between the threads.

Any application that can be threaded can be mapped efficiently to the multicore, but the improvement in performance gained by the use of multicore processors depends upon the fraction of the program that can be parallelized.[Amdahl's law] (10)

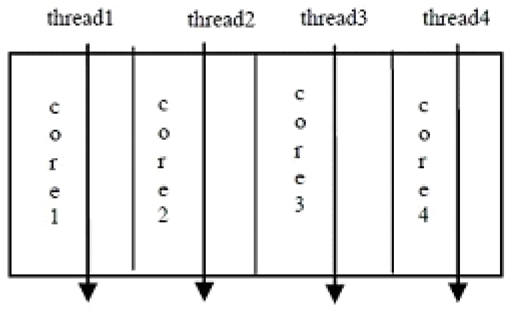

Fig IV: Independent Threads on each Core

G.Parallel Partition Algorithm [1]

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

Phase 1: In first phase each processor scans its local partition and produce the local frequent itemsets and sends it to the coordinator.

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

Phase 3: In this phase each processor computes the local support counts of the global candidates and sends to the coordinator.

Phase 4: The coordinator produces 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:

Table I(a)

Algorithm	Type of messages exchanged	Number of Messages Exchanged
Count Distribution Algorithm	local counts (in each pass)	n(n-1) (in each pass)
Data Distribution Algorithm	local frequent itemsets (in each pass)	n(n-1)(in each pass)
Candidate Distribution Algorithm	local frequent itemsets for initial passes and database is repartitioned	n(n-1) (in initial passes)
Common Candidate Partitioned Database Algorithm	not applicable	none
Equivalence Class Transformation Algorithm	local counts in initialization phase and tid lists in transformation phase and database is repartitioned	n(n-1) (in two phases)
Parallel Partition Algorithm	local frequent itemsets in phase 1, global candidates in phase 2 and local counts in	3n (in three phases)

Table I(b)

Algorithm	Synchronization Required	Architecture	Database Layout
Count Distribution Algorithm	Yes (after each pass)	Shared-Nothing	Horizontal
Data Distribution Algorithm	Yes (after each pass)	Shared-Nothing	Horizontal
Candidate Distribution Algorithm	No	Shared-Nothing	Horizontal
Common Candidate Partitioned Database Algorithm	Yes (after each pass)	Shared-Memory	Horizontal
Equivalence Class - Transformation Algorithm	No	SharedNothing	Horizontal and Vertical
Parallel Partition Algorithm	Yes (in phase 2 and phase 4)	Client-Server	vertical

V. CONCLUSION

The performance of the parallel apriori algorithms based on the data communication cost and performance of time . 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 based on the database layout, the database is scanned in numbers of times and the size of the candidates generated. Vertical database layout speeds up the searching process as demonstrated in the Apriori Algorithm and decreases 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",

[4] International Journal of Computer Science and Research

(IJCR), Vol. 1 Issue 1, 2010.

[5] 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.

[6] 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.

[7] 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.

[8] Mohammed J. Zaki, "Parallel and Distributed Association Mining: A Survey", IEEE Concurrency, Vol 7, Issue 4, pp 14-25, October 1999.

[9] 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.

[10] 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.