Dynamic Request Redirection for Mining System Services under Heterogeneous Distributed System

Dhanashree R. Kolhe, RKDF Institute of Science & Technology, Bhopal, dhanshreek2@gmail.com

Dr. Varsha Namdeo, Head of Department of computer, RKDF Institute of Science & Technology, Bhopal, Varsha_namdeo@yahoo.com

Abstract- Algorithms are used on very big data sets with high dimensionality. Therefore, fast processing can be applied for mining using association rules. The process of association rule mining consists of identifying frequent item sets and generating rules from the frequent item data sets. Finding frequent item sets is more complex in terms of CPU power consumption and computing resources utilization. Thus, majority of parallel apriori algorithms focus on parallelizing the process of discovering frequent item set. The computation of frequent item sets mainly consist of creating the candidates and counting them. In parallel frequent item sets mining algorithms addresses the issue of distributing the candidates among processors such that their creation and counting is effectively parallelized. This paper presents comparative study of these algorithms.

Keywords—Parallel data mining, frequent item sets, association rules, apriori algorithm.

I. INTRODUCTION

Accumulation of plentiful data from different sources of the society but a little knowledge situation has lead to knowledge discovery from databases which is also called data mining. Data mining techniques use the existing data and retrieve the helpful information from it which is not directly visible in the original data. As data mining algorithms deal with large amount of data, the primary concerns are how to store the data in the main memory at run time and how to increase the run time performance. Sequential algorithms cannot supply scalability, in terms of the data dimension, size, or runtime performance, for such large amount of 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 huge volumes of data to extract the useful knowledge. Association Rule Mining (ARM) or frequent item set mining is an important functionality of data mining. The apriori algorithm is one of the best algorithms for discover frequent itemsets from a transaction database. As data mining mainly deals with large volumes of data, the main issue is how to improve the performance of the algorithm. One way of improving the performance of apriori is parallelizing the process of generate frequent itemsets. The rest of the paper is organized as follows. In Section 2 related work is overviewed. In Section 3 concepts of association rule mining are discussed and apriori algorithm is described. In Section 4 comparative analysis of parallel apriori algorithms is given. In Section 5 conclusion is given.

II. RELATED WORK

Many parallel ARM algorithms have been given which represent transactions using either horizontal data format or vertical data format [4, 7]. In horizontal data format, data is presented as transaction ID versus items sold in each transaction whereas in vertical data format, data is presented as each item versus transaction ids in which the item is sold. There are several parallel association rule mining algorithms based on data set partitioning like Count Distribution, Data Distribution, Candidate Distribution, Common Candidate Partition, Parallel Partition [1, 5, 9, 10].

III. ASSOCIATION RULE MINING

A. *Basic Concept*

The basic concept of association rule mining is arises from the market basket analysis. Let D be the transaction database which composed 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 market. The items are the subset of the set of whole items I {I1, I2,..Im} in the market that are considered for analysis. An itemset consists of some combination of items which occur together or a single item from I. In Association rule mining X->Y, represents the dependency relationship between two different itemsets X and Y in DB. The dependency is at any time X is occurring in any transaction, there is a probability that Y may also occur in same transaction. This skill is based on two interesting measures.

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

 DOI : 10.18231/2454-9150.2018.0301

Support(X->7) = *P* (X U Y).

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

Apriori Algorithm

Apriori algorithm is the most established association rule mining algorithm. It is based on the apriori principle that all the nonempty (at least one) 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 preceive the count of each candidate in Ck where Ck represents candidate k- itemset. The count of each itemset in Ck is match up with a predefined minimum support count to find whether that itemset can be arranged in frequent k-itemsetLk.

Step 2: The join step

Lk is natural joined with itself to generate the next candidate k+1-itemset Ck+1. The main step here is the prune step which requires scanning the whole 1database for finding the count of each itemset in whole candidate k-itemset. If the database is enormous then it requires more time to find all the frequent itemsets in the DB.

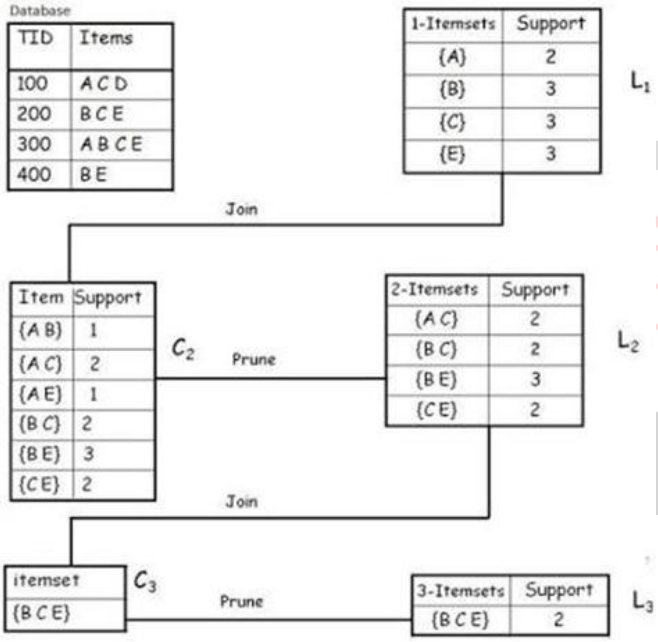


Figure 1: Example for Apriori Algorithm

IV. SYSTEM ARCHITECTURE

1 Single Core CPU Architecture

Single-core. A single-core processor is a microprocessor with a single core on a chip, running a single thread at any one time[1].

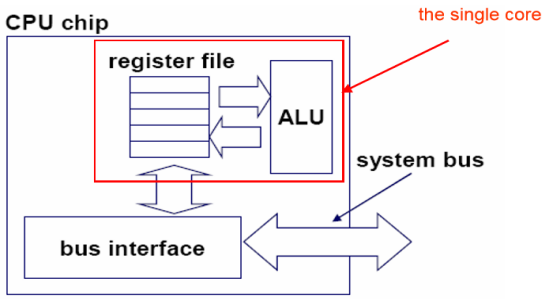


Figure 2

2 Multicore CPU Architecture

Multi core indicate two or more processors. But they differ from separate parallel processors as they are combined on the same chip circuit. A multi core processor developed message passing or shared memory inter core communication methods for multiprocessing.

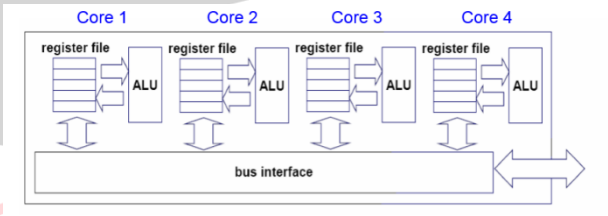


Figure 3

3.Serial Threading approach

A serial mining is defined to be a partially ordered set of events for consecutive and fixed-time intervals in a sequence. It mainly involves mining by using single thread environment. In this proposed system we are using serial mining on apriori algorithms in sequential fashion.

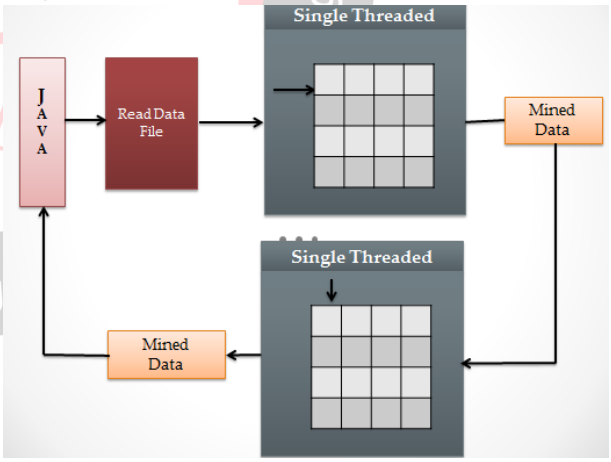


Figure 4

4 Multithreading approach

A parallel mining is defined to be a partially ordered set of events for concurrent and fixed-time intervals parallelly. It mainly involves mining by using multi-threading environment. In this proposed system we are using parallel mining on apriori algorithms in parallel fashion.

 DOI : 10.18231/2454-9150.2018.0301

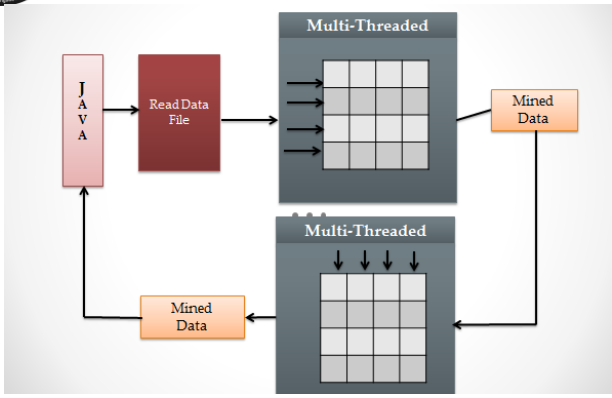


Figure 5

V. MODULAR APPROACH

1. Authentication GUI

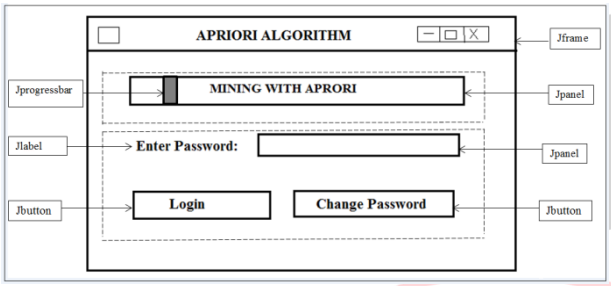


Figure 6

1.1 Database Design

Transaction:-

A transaction is a unit of work that is performed against a database.

It is a very small unit of a program and it may contain several low level tasks.

Item :-

unit of data containing in a record describing a particular attribute.

A data item describes an atomic sate of a particular object concerning when looking at databases.

1.2 Database Example

Database:-

it is an organized collection of data (transactions), queries, reports, views, other object.

A database is a collection of information that is organized so that it can easily be accessed, managed and updated.

Ex1: - Database 1(Fruits): - This database contains 10 itemset and 20 transaction. (items: mango, banana, apple …etc.)

Ex2:-Database 2(Medicines): - this database contains 20 itemsets and 25 transactions (items: Lomofem, disprin, Decold, crocin, Codlever tabs…etc.)

Ex3: - Database 3(Book stall): - this database contains 25 itemset and 100 transactions (items: applied science, TOC, MIT, BAI, SSDA, SDMT…etc.)

1.3 Serial and Parallel Mining

Candidate Generation: - The Apriori Algorithm identifies item set which are sub set of at least transaction in the database. Apriori uses bottom up approach where frequent subset are extended one item at a time a step known as candidate generation.

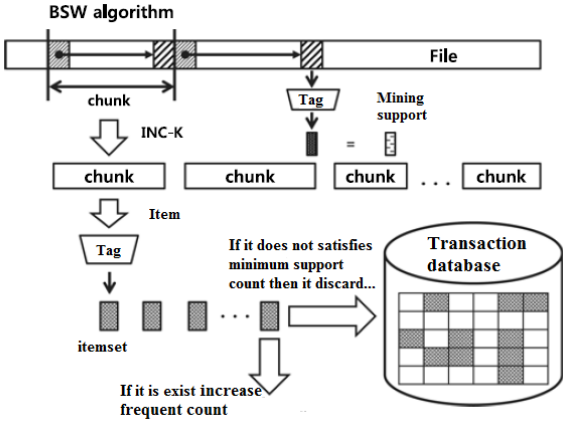


Figure 7

VI. PARALLEL APRIORI ALGORITHMS

A. *Count Distribution Algorithm*

Every processor generates the partial support of all candidate itemsets from its local database partition in parallel. In 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 Lk-1. Each processor thus independently computes the partial supports of candidates from its local database partition. Then each processor exchanges 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 determined, every processor builds Ck+1 in parallel and repeats the process until all frequent item sets are found.[11]

Advantages:

It cut down the communication cost as only counts are exchanged among the processors and speeds up the addition process as only vector addition is to be carried out in place of matching the candidates first and then finding their sum.

Disadvantages:

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

B. *Data Distribution Algorithm*

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

 DOI : 10.18231/2454-9150.2018.0301

Advantages:

It utilizes the total system memory with ease by generating disjoint candidate sets on each processor. The addition need not to be execute since local frequent itemsets are dislocate.

Disadvantages:

It suffers from huge communication cost.[11]

C. Candidate Distribution Algorithm

In the initial passes it uses either Count Distribution or Data Distribution algorithm. Then in some pass I which is heuristically determined, this algorithm break down the frequent itemsets L_{k-1} among the processors in such a way that each processor can generate exclusive candidate sets independent of each processor can calculate the counts of the candidate set independent.

Advantages:

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

Disadvantages:

It suffers from massive communication cost of redistributing the database and scans the local partitions repeatedly. The communication hike of independent processing are not sufficient to offset the database redistribution cost.[11]

D. Common Candidate Partitioned Algorithm (CCPD)

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

Advantages:

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

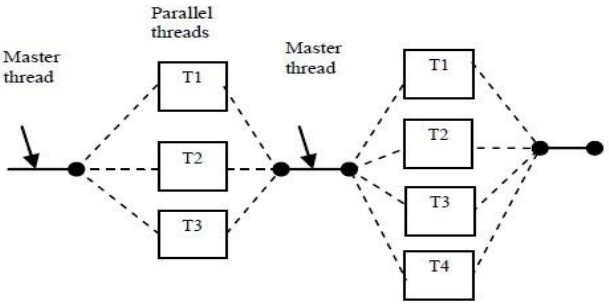
Disadvantages:

It uses complicated hash structures which obtain additional overhead of maintaining and searching as it is poor cache locality.[6]

E. Fork-Join Parallelism

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

Figure 8: Fork-Join Parallelism



F. *High Performance Computing (HPC)*

"Multicore System of HPC framework" technique improve the performance of data mining algorithm which uses sequential processing they are modified using the theory of parallel processing. Still there is a chance of performance improvement of Apriori algorithm by using Parallel Processing. HPC systems, also popularly referred as Supercomputers, generally capitalize on aggregating computing power in a way that delivers much higher performance than one could get out of a typical single desktop computer or workstation in order to solve large problems in engineering, or business. They are used for a wide range of computationally in depth tasks in various fields, comprise quantum mechanics, weather forecasting, climate research, oil and gas exploration, molecular modeling and or physical simulations. HPC systems have been shifting from expensive massively parallel architectures to clusters of commodity computers to take advantage of cost and performance benefits.G. Multi core

Multi core assign two or more processors. But they differ from independent parallel processors as they are integrated on the similar chip circuit [7,8]. A multi core processor implement message passing or shared memory inter core communication methods in multiprocessing. If the number of threads are less than or equal to the number of cores, separate core is provided to each thread and threads run independently on multiple cores. (Figure 1) If the number of threads are more than the number of cores, the cores are distributed among the threads. Any application that can be threaded can be mapped effortlessly to multi-core, but the improvement in performance gained by the usage of multi core processors depends on the portion of the program that can be parallelized. [Amdahl's law][10]

 DOI : 10.18231/2454-9150.2018.0301

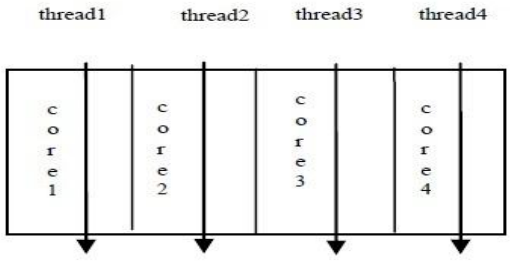


Figure 9: Independent thread on core

VII. 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] KhadidjaBelbachir, HafidaBelbachir, "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] RuowuZhong, 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] AzizGinwala, Priyankakonde, PriyankaBhalekar, MayuriJambhulkar ,Prof.SunilYadav "Performance Enhancement Scheme for Multithreading Application Using Chunking Mechanism".

[4] 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.

[5] Xindong Wu , Vipin Kumar, J. Ross Quinlan, JoydeepGhosh, 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, SrinivasanParthasarathy, MitsunoriOgihara, 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, SrinivasanParthasarathy, MitsunoriOgihara, 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, SrinivasanParthasarathy, MitsunoriOgihara, Wei Li, "A Localized Algorithm for Parallel Association Mining", Proceedings of the ninth annual ACM symposium on Parallel algorithms and architectures, ACM 1997.

[11] RakeshAgrawal, John C. Shafer, "Parallel Mining of Association Rules", IEEE Transactions on Knowledge and Data Engineering, December 1996.

 DOI : 10.18231/2454-9150.2018.0301