Dynamic Request Redirection for Mining Services under Heterogeneous Client-server System

Prof. Ketaki Bhoyar*, Gurpreet Kaur Lohana*, Laxmi Nirmal*, Sunidhi Patil*, Rakesh Pandey*

*Department of Computer Science, DYPIEMR, Pune- 411033, Maharashtra, India

Abstract -**The main purpose of developing system is to find the independent workload for that we have to find the internal task of apriori algorithm with a systematic method called Dynamical Request Redirection and Resource Provisioning .Those internal task which have independent work load has to be executed in parallel manner. Apriori algorithm is a classical association rule mining algorithm, but it has problems about frequently scanning database and generating a large number of candidate item set. To solve this problem, we have proposed Frequent Item Mining system with parallel execution techniques. A large dataset, here is divided into number of small datasets i.e. chunks. The legacy single threaded variable size chunking method leaves much to be desired in states. This is achieved by inter or intra segment assignment. The proposed system achieves the speed-up using multiple threads in heterogeneous system over existing sequential system as well as utilize the computational power which now a days provided by recently lunched multicore processor . Parallelism is used to reduce time, increase performance and fast processing. It is implementing Apriori algorithm in serial and parallel manner and performing comparison of both on the basis of varying support-count and time using multithreading java.**

Keywords- **Apriori, Association Rule Mining, Heterogeneous System, Candidate Item Set , Resource Provision.**

I. INTRODUCTION

An Apriori is a frequent pattern mining algorithm used for finding association rules developed by Rakesh Agrawal and Ramakrishnan Srikant[4]. It operates on a list of transactions containing items (for example, products in a supermarket). Frequent instances of items with each other are mined by Frequent Item Calculation to find relationship among different items. A single transaction is called an Item set. Apriori uses a reduce support value as the main compulsion to determine whether a set of items is frequent. In start pass of the algorithm, it constructs the candidate 1-itemsets. The algorithm then generates the frequent 1-itemsets by excluding some candidate 1-itemsets if their support values are lower than the minimum support. After the algorithm finds all the frequent 1-itemsets, it joins the frequent 1-itemsets with each other to construct the candidate 2-itemsets and exclude some infrequent itemsets from the candidate 2-itemsets to create the frequent 2-itemsets. This process is replicated until no more candidate itemsets can be created from it.

Apriori has a bottleneck when the number of actions is very big since multiple database scans will be performed by apriori in every iteration. To improve the scalability of Apriori, different data structures and methods were constructed that can hike performance. These include:

Transaction Reduction: Atransaction that does not contain any frequent *k*-itemsets cannot contain any frequent (*k*+1)-itemsets. Therefore, such a transaction can be marked or detached from further consideration

Partitioning: The partitioning techniques can be used that requires just two database scanning to mine the frequent itemsets. It consists of two phases. In Phase I, the algorithm separate the transactions of *D*sinto *n* nonoverlapping partitions. If the minimum support count threshold for transactions in *D*sis *min sup*, then the minimum support count for a partition is *minsup, the number of transactions in that partition*. For each partition, all frequent itemsets within the partition are found. These are referred to asa local frequent itemsets. In Phase II, a second scan of *D*sis conducted in which the absolute support of each candidate is assessed in order to determine the global frequent itemsets. Partition size and number of partitions are set so that each partition can fit into main memory and therefore be read only once in each phase.

Sampling: The basic idea of the sampling approach is pick a random sample *AS*of the given data *DS*, and then search for frequent itemsets in A*S* instead of *DS*. In this way, we trade off some degree of efficiency. Sample size of A*S* is such that the search for frequent itemsets in A*S* can be done in main memory, and so only one scan of the transactions in A*S* is required overall.

*Dynamic itemset counting***:** A dynamic itemset counting technique [1] was the proposed in which the database is divided into blocks marked by start point. In new candidate itemsets can added at any start point, unlike in Apriori, which determines new candidate itemsets only immediately before each complete database scan. The technique is dynamic in that it estimates the support of all of the itemsets that have been counted so far, adding new candidate itemsets if all of their subsets are estimated to be usual.

II. LITERATURE SURVEY

Apriori algorithm is a primary algorithm for association rule mining. A supermarket wants to implement a bundling sale. They need to find the items buy or asset together frequently.

This procedure evaluates customer buying habits by recommending associations between the different items that customers put in their "shipping baskets". The result can help retailers establish or expand marketing strategies by getting to know which items are frequently bought together by customers. Apriori is a positive solution to this Association rules mining problem.

Traditional methods waste lot of time to resolve the problems or decision making for profitable business. Data mining formulate databases for finding unknown or hidden patterns, finding anticipating information that experts may omit. Hence, this paper reviews the various trends of data mining and its relative applications from past to present and discusses how adequately can be used for targeting profitable customers in campaigns and utilize the multiple cores of the processor for faster execution. Apriori algorithm [1] and [13] was recommended by R Agarwal , R Srikant and Vincent S. Tseng for exploring frequent item set for Boolean association rule, It deliver a frequent item set in transactional database as an output. It's an efficient algorithm for finding frequent items. Disadvantage is it generates massive number of candidate item set. Repeatedly scanning the transaction databases. Record filter approach [4] only those transactions are considered to determine the support count of candidate set whose length is greater than the length of candidate item set. If length of candidate item set is k, only transaction whose length is at least k is considered as k length candidate set cannot exist in the transaction record whose length is is then this approach takes less time as compared to classical apriori algorithm this it improves the efficiency of apriori algorithm and memory management. It removes the complexity of process. Disadvantage is memory optimization.

AVI algorithm [7] Transaction database is vertical, item set union and identification intersection is used. For item set X, t (X) = {tid | tid is transaction id, t belongs to D and t supports X}; for the identification set Y, i(Y) = y belongs to Y item set(y), item set (y) that y corresponding to the transaction item set [5]. Sampling method [6] chooses the arbitrary sample S for given database D, and then investigate frequent itemsets in the S rather in D. As we have to scan only sample S instead of whole database D, it recover time. This approach sacrifices some efficiency. [11]The client assignment problem in client-server systems is the problem of NP-hard. We will manage the client and server by using the distributed system and synchronization and multithreading[12]. The concept of dynamically allocation and request redirection of the system taken from[10].

III. TECHNOLOGY USED FOR IMPLEMENTATION

3.1 Multithreading in java

Multithreaded program involves multiple threads of control within a single program on single or multiple environments. In multithreaded programming model, a single process can have multiple, concurrent execution paths on CPUs. Thread affinity benefit a thread to run on a specific subset of processors. Multithreaded programming is written in many programming languages with an improvement of setting an affinity to threads. Java supports to develop multithreaded programming, while it does not contain any method to set an affinity for threads on CPU.

In fact, Java adopts threads to enable the entire environment to be no synchronous. This helps to lower inefficiency by preventing the waste of CPU cycles. Javamultithreading system is assembled upon the Thread class, its methods, and its companion interface Runnable. The Thread class specifies several methods that help on operate threads. After a computational job is designed and realized as a set of tasks, an optimal assignment of these tasks to the processing elements in a given architecture needs to be determined.

This problem is called the scheduling problem and is known to be one of the most challenging problems in parallel and distributed computing. The intention of scheduling is to determine an assignment of tasks to processing elements in order to optimize certain performance indexes.

- Permits multiple independent threads to execute SIMULTANEOUSLY on the SAME core.
- Weaving together multiple \threads" on the same core.
- Example: if one thread is waiting for a floating point operation to complete, another thread can use the integer units.

3.2 Multi core and Many Core Systems

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.

3.3 PROPOSED SYSTEM

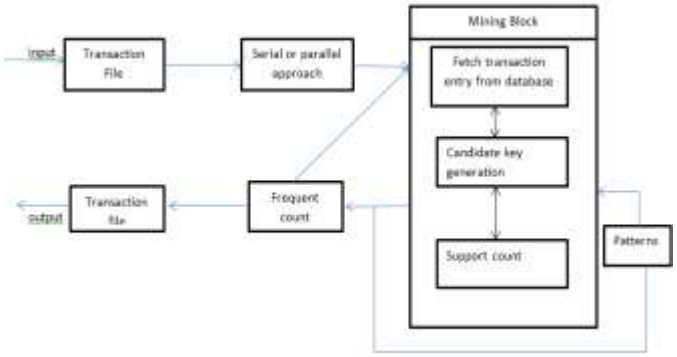

Fig 1.System Archtitecture

Our approach focuses on the algorithm part of Apriori by trying to maximize the workload being executed in parallel and to minimize, as much as possible, the synchronization point delays by allowing a portion of the superset generation

forming the candidates to be generated in parallel as well. Only the remaining part of the superset will be generated at the synchronization point.

The proposed algorithm is described in the following steps:

1. Select one processor to be the master, the other N-1 processors are slaves
2. Create a local Hash Table, at the master; we refer to it as G during the remaining part of this project.
3. Initialize K, the itemset size, to 1
4. Generate the 1-itemsets normally using the master processor only
5. At the master, divide the k-itemsets into N-1 groups and assign each group to exactly 1 of the available N-1 processors. Then for each processor:
    a. Lookup the support of each local itemset in G, if it's not found, compute support.
    b. Prune itemsets that have their support less than the minimum support where minimum support is the threshold value to decide upon whether to drop or keep an itemset. It is also global to the entire system.
    c. Generate the candidates of the K+1 itemsets from the local items and find their support.
    d. Store the results of the K+1 itemsets (the itemset and its support) into G.
    e. Send the master processor a signal indicating end of processing.
6. At the master processor, at the end of each arrival of a finalization signal:
    a. Generate the K+1 superset using itemsets in G.
7. Increment K by 1.
8. Go to step 2 and repeat until there are no more itemsets to generate.

IV. FIGURES AND TABLES

4.1 Result Analysis

Transaction * Items	Serial Approach(ms)	Parallel Approach(ms)
20*10	**3.0**	**17**
25*20	**442**	**39**
100*20	**27613**	**181**

V. CONCLUSION

From serial approach we will make conclusion that by serial execution of input may be assigned to one of all CPU cores and load balancing is not maintained . From parallel approach we will make conclusion that the proposing system will give same input as that of serial execution but it will divide that process into task and each task will be assigned to single CPU .Hence parallel execution will make all CPU cores to utilize fully until it completes execution. So load balancing will done among all cores.

Hence the proposing system will reduce time and space complexity as than that of serial approach.

ACKNOWLEDGEMENTS

I would like to take this opportunity to thank my internal guide Prof. KetakiBhoyarfor giving me all the help and guidance I needed I am really grateful to them for their kind support. Their valuable suggestions were very helpful. I am also grateful to Prof. P. P. Shevatekar, Head of Computer Engineering Department, DYPIEMR for his indispensable support, suggestions.

REFERENCES

[1] R. Agrawal and R. Srikant. Fast algorithms for mining association rules. IBM Research Report RJ9839, IBM Almaden Research Center, San Jose, California, June 1994.

[2] Youjip Won, Kyeongyeol Lim, and Jaehong Min. "MUCH: Multithreaded Content-Based File Chunking", IEEE Transaction on Computers, VOL. 64, NO. 5, ISSN: 0018-9340, May 2015.

[3] SsJugendra Dongre, S. V. Tokekar, and GendLal Prajapati, "The Role of Apriori Algorithm for Finding the Association Rules in Data Mining" International Conference on Issues and Challenges in Intelligent Computing Techniques (ICICT),IEEE Catalogue Number: CFP1463W-DVD ISBN: 978-1-4799-2899-6, 2014.

[4] Sheila A. Abaya, "Association Rule Mining based on Apriori Algorithm in Minimizing Candidate Generation",In:International Journal of Scientific & Engineering Research Volume 3, Issue 7, July-2012.

[5] Mamta Dhanda," An Approach To Extract Efficient Frequent Patterns From Transactional Database",In: International Journal of Engineering Science and Technology (IJEST), Vol.3 No.7 July 2011, ISSN:0975-546.

[6] Junjie Qian, Du Li, Witawas Srisa-an, Hong Jiang and Sharad Seth, "Factors Affecting Scalability of Multithreaded Java Applications on Manycore Systems", 2015 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS) 2015, pp. 167-168, doi:10.1109/ISPASS.2015.7095800, 2015.

[7] Jaishree Singh*, Hari Ram*, Dr. J.S. Sodhi, "Improving the efficiency of Apriori algorithm by transaction reduction", International Journal of Scientific and Research Publications, Volume 3, Issue 1, ISSN 2250-3153, January 2013.

[8] Zaki MJ (1999) "Parallel and distributed association mining: a survey. Concurrency," IEEE (Volume: 7, Issue: 4), Page(s): 14–25, Special issue on Parallel Mechanisms for Data Mining, ISSN: 1092-3063.

[9] Bala Dhandayuthapani Veerasamy, Dr. G.M. Nasira "Setting CPU Affinity in Windows Based SMP Systems Using Java"IJSER(Volume: 3, Issue: 4) , April-2012, ISSN: 2229-5518.

[10] Wenhua Xiao, Weidong Bao, Xiaomin Zhu, Member, IEEE, Dynamic Request Redirection and Resource Provisioning for Cloud-based Video Services under Heterogeneous Environment, IEEE Transactions on Parallel and Distributed Systems Volume: 27, Issue: 7, November 22, 2016

[11] Yuqing Zhu, Member, IEEE, "Efficient Client Assignment for Client-Server Systems", IEEE TRANSACTIONS ON NETWORK AND SERVICE MANAGEMENT, VOL. 13, NO. 4, DECEMBER 2016

[12] Teng Li, Student Member, IEEE, Performance Modeling and Predictive Scheduling for Distributed Stream Data Processing, IEEE TRANSACTIONS ON BIG DATA, VOL. 2, NO. 4, OCTOBER-DECEMBER 2016

[13] Vincent S. Tseng, Cheng-Wei Wu, Philippe Fournier-Viger, and Philip S. Yu, Fellow, IEEE, "Efficient Algorithms for Mining the Concise and Lossless Representation of High Utility Itemsets," IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING, VOL. 27, NO. 3, DECEMBER 2016