Parallel Association Rule Mining on Heterogeneous System

Rakhi Garg
Computer Science Section,
MMV, Banaras Hindu University,
Varanasi

P. K. Mishra
Department of Computer Science,
Banaras Hindu University,
Varanasi

ABSTRACT
Association Rule Mining from transaction–oriented databases is one of the important process that finds relation between items and plays important role in decision making. Parallel algorithms are required because of large size of the database to be mined. Most of the algorithms designed were for homogeneous system uses static load balancing technique which is far from reality. A parallel algorithm for heterogeneous system is regarded as one of the most promising platforms for association rule mining. In this paper we propose a simple parallel algorithm for association rule mining on heterogeneous system with dynamic load balancing based on Par-Maxclique algorithm. We maintain one linked list at the scheduler end that keep track of load value of every processor and each processor is having a job queue associated with it which is served in First come first basis. On the basis of load value scheduler directs the migration of task from heavy loaded to least loaded processor in the cluster during the execution and thus balances load dynamically in a cluster.

Keywords
Parallel association rule mining, heterogeneous system, Par-MaxClique algorithm

1. INTRODUCTION
Most of the parallel association rule mining algorithm developed so far uses static load balancing for homogeneous systems [12]. Static load balancing initially partitions work among the homogeneous processors using some heuristics; no data movement for current load balancing during the execution is available.
If we apply the parallel algorithm developed for homogeneous system to heterogeneous environment, it will leads to significant performance deterioration [1]. Since in homogeneous system there is an equal distribution of job among the processors of the same speed, uses static load balancing technique whereas heterogeneous system has processors of different speeds in which one completes job earlier than the other due to speed mismatch [4]. In this case high speed processor executes the assigned job quickly and sits idle while low speed processor is still busy with the assigned job. It degrades the performance of the system. To utilize system processors efficiently and enhance the performance we design an algorithm that during execution checks the load of the processor and on the basis of that it moves the job from heavy loaded processor to least loaded one so that no processor sits idle till the completion of the whole jobs in a system.
The algorithm assumes heterogeneous environment where there is no prior knowledge of processing speed of processors. Initially, there is same number of jobs assigned to all the processors in a cluster by the scheduler assuming homogeneous environment by using static load balancing technique. During the execution it checks the load value of all the processors which is computed by using the job queues of every processor in a cluster, at the scheduler end. The load value of processors in a cluster are compared during the execution, the job is moved from the heavy loaded processor to the least loaded one and thus balances load dynamically in a cluster. A linked list containing the load values of all processors in a cluster are maintained at the scheduler end that gets updated during the execution of jobs. In this way load balancing becomes dynamic and involves movement of data from one processor to another.
Section 2 briefly explains association rule mining, its parallel algorithms and Par-MaxClique algorithm. Section 3 introduces the algorithm designed by us and section 4 describes the analysis of the proposed algorithm. At the end we give conclusion that comprise of our future work.

2. ASSOCIATION RULE MINING: PARALLEL ALGORITHM AND PAR-MAXCLIQUE ALGORITHM

2.1 Association Rule Mining
Let J = $\{i_1, i_2,\ldots,i_m\}$ be a set of items, D be a set of database transactions and T is transaction contains set of items such that $T \subset J$. Each transaction is associated with an identifier, called TID. Let A, B be set of items and T is said to contain A, iff $A \subset T$. An association rule is an implication of the form $A \Rightarrow B$ holds, where $A \subset J$ and also $B \subset J$ and $A \cap B$ is NULL. The rule $A \Rightarrow B$ holds in transactions set D with support s, where s is the percentage of both A and B. The rule has confidence c in the transaction set D if c is the percentage of transactions in D containing A that also contains B [3]. ARM is two step process:-
1. Find all frequent itemsets having minimum support.

2. Generate strong association rules having minimum confidence, from the frequent itemsets.

The efficiency of Association Rules Mining Algorithms can be enhanced by reducing the computational cost of association rules mining in four ways [13]:

- By reducing the number of passes over the database
- By sampling the database
- By adding extra constraints on the structure of patterns
- Through parallelization.

2.2 Parallel association rule mining algorithms

Although Apriori is a simplest sequential ARM algorithm designed so far but it has some limitations like large number of candidate itemsets were generated that scans database at every step.

To overcome these demerits parallel algorithms are designed on different platforms i.e. shared memory system (SMS) and distributed memory system (DMS). CCPD (Common Candidate Partitioned Database) and PCCD (Partitioned Candidate Common Database) was proposed by M. J. Zaki, M. Ogihara, S. Parthasarathy and W.Li. [3] for shared memory system. The main design issue was minimization/elimination of false sharing and the maintenance of good data locality [3, 4]. In case of CCPD serial I/O depreciates the performance while I/O overhead and disk contention for PCCD was unacceptable, resulting in slow-downs on more than one processor.

Count Distribution (CD), Data Distribution (DD) and Candidate Distribution (Cand Dist) were proposed by Rakesh Agrawal and J. Shafer [4, 5, 6]. The main design issue is minimization of communication and load balancing. Although CD minimizes communication since only count is exchanged among processors but it doesn't utilize memory as the entire hash tree is replicated on each processor. DD efficiently utilizes system memory but suffers from high communication overhead because of all to all broadcast to send the local database portion to every other processor and is unable to divide the work done on each transaction at every processor. Candidate distribution provides balancing of work load of all processors but the cost incorporated due to redistribution of the database and scanning local database partition repeatedly. To overcome these limitations of CD, DD, Candidate distribution algorithms, Non-Partitioned Apriori (NPA), Simply-Partitioned Apriori (SPA) and Hash-Partitioned Apriori (HPA) were proposed by T. Shintani and M. Kitsuregawa [4, 7]. It also has limitations like use of complicated internal structures and additional computation overheads as compared to DD, no proper utilization of memory and involves extra computation overheads.

On the other hand Fast Parallel Mining (FPM) for Distributed Memory System, developed by D. Cheung and Y. Xiao proposed FPM [8, 9] improves Count Distribution by adopting two pruning i.e. distributed and local and generates less number of candidates but is more sensitive to work load balance than data skewness. Another limitation is to obtain very high ARM efficiency, first database should be partitioned using Balanced k-means clustering and then FPM is executed on it.

After that Intelligent Data Distribution (IDD) and Hybrid Distribution (HD) were proposed by E-H. Han, G. Karypis and V. Kumar [6]. IDD too have some disadvantages like it involves use of complicated structures to partition items. As number of processors increases the number of candidates assigned to every processor reduces that leads to two problems (a) With fewer number of candidates per processor it's much difficult to balance the work and (b) the smaller number of candidates gives smaller HT and less computation work per data that reduces overall efficiency. Similarly there are limitations associated with HD also i.e. It involves extra computation to determine number of processor in a group at every pass.

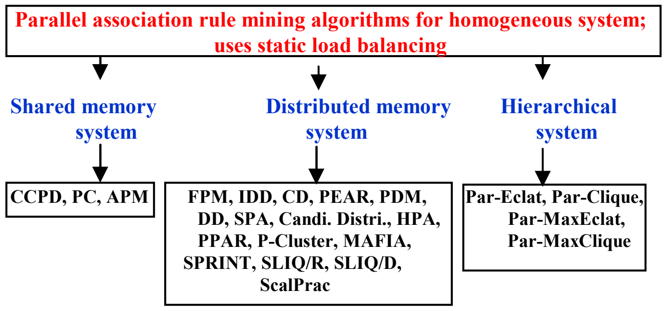

Parallel association rule mining algorithms for homogeneous system; uses static load balancing
Shared memory system
Distributed memory system
Hierarchical system
CCPD, PC, APM
FPM, IDD, CD, PEAR, PDM, DD, SPA, Candi. Distri., HPA, PPAR, P-Cluster, MAFIA, SPRINT, SLIQ/R, SLIQ/D, ScalPrac
Par-Eclat, Par-Clique, Par-MaxEclat, Par-MaxClique

Figure 1: List of Parallel association rule mining algorithm developed so far for homogeneous system that uses staic load balancing technique on different machines i.e. shared memory and distributed memory system [4]

All algorithms discussed above were implemented on dedicated homogeneous system and uses static load balancing technique based on the initial data decomposition for load assignment to the processors in the system. This is far from reality. A typical parallel database server has multiple users, and has transient loads. This calls for an investigation of dynamic load balancing schemes. Dynamic load balancing is also crucial in a heterogeneous environment, which can be composed of meta-and super-clusters, with machines ranging from ordinary workstations to supercomputers [4].

Kun-Ming Yu, Jiayi Zhou and Wei Chen Hsiao proposed a parallel and distributed mining algorithm based on FP-tree structure, Load Balancing FP-Tree (LFP-tree) [11]. The algorithm divides the item set for mining by evaluating the tree's width and depth and proposed a simple and trusty calculate formulation for loading degree. The experimental results show that LFP-tree can reduce the computation time and has less idle time compared with Parallel FP-Tree (PFP-tree) and has better speed-up ratio than PFP-tree when number of processors grow [11]. But the problem is that it involves the maintenance of complex tree structure.

Masaru Kitsuregawa and Takahilus Shintani, Masahisa Tamura and Iko Pramudiono, proposed Parallel Data Mining on large scale PC Cluster, the new dynamic load balancing methods for association rule mining, which works under heterogeneous system [14]. In this, two strategies, called candidate migration and transaction migration are proposed. Initially first one is invoked. When the load imbalance cannot be resolved with the first method, the second one is employed, which is costly but more effective for strong imbalance.

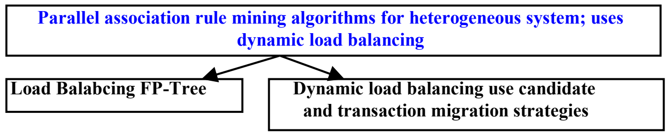

Parallel association rule mining algorithms for heterogeneous system; uses dynamic load balancing
Load Balabcing FP-Tree
Dynamic load balancing use candidate and transaction migration strategies

Figure 2: List of Parallel association rule mining algorithm developed so far

2.3 Par-MaxClique algorithm

A completely different design characterizes the equivalence class based algorithms Par-Eclat, Par-MaxEclat, Par-Clique and Par-MaxClique proposed by Zaki, Parathasarathy, Oghihara and Li [2]. Par-MaxClique algorithm gives more accurate frequent itemsets as it uses clique clustering which is more accurate than equivalence class clustering [2, 10].This method utilizes a vertical database format, a hybrid search, and generate only the longest maximal frequent itemsets and some non frequent itemsets. The algorithms utilize the structural properties of frequent itemsets to facilitate fast discovery. The items are organized in a subset lattice search space, which is decomposed into small independent chunks or sub-lattices, which can be solved in memory. Efficient lattice traversal techniques are used, which quickly identify all the frequent itemsets via simple tid-list intersections [2].

Basically Par-MaxClique algorithm is divided into three phases i.e. initialization phase, asynchronous phase and final reduction phase [2, 10]. In the initialization phase it generates clusters from L_2 using uniform hypergraph cliques and partition the clusters and the tid-list among the processors. In the asynchronous phase the frequent itemsets are computed independently by each processors from the cliques assigned to it. At last, the final reduction phase produces the aggregate results and outputs the associations between the frequent itemsets.

EXAMPLE OF PAR-MAXCLIQUE ALGORITHM

Let database contains A,C,D,T and W four itemsets and 6 transactions are:-

Transactions	A	C	D	T	W
T1	1	1	0	1	1
T2	0	1	1	0	1
T3	1	1	0	1	1
T4	1	1	1	0	1
T5	1	1	1	1	1
T6	0	1	1	1	0

Tid-list is computed as: T(A) = {1,3,4,5}; T(C)={1,2,3,4,5,6}; T(D)={2,4,5,6} and T(W)={1,2,3,4,5}. During the initialization phase the tid-list is communicated among the processors and support counts for 2-itemsets are read. e.g. support count for AC ={1,3,4,5} = 4 which is counted by the intersection of the tid list of A and C. Similarly the support counts of AD, AT, AW, CD, CT, CW, DT, DW and TW are 2,3,4,3,4,4,3,2,3 and 3 respectively. Let us assume that minimum support = 3 so AD and DT will be discarded.

Frequent 2- itemsets are :-	Equivalence classes are:-
AC,AT,AW,CD,CT,CW,DW,TW	[A]: C T W
	[C]: D T W
	[D]: W
	[T]: W

By applying the hypergraph clique for clustering to L2, the set of potential maximal cliques per equivalence class are generated.

Generated Maximal cliques per class:-

[A]: ACTW, ACW, ATW, ACT

[C]: CDW, CTW

Maximal cliques for equivalence class A

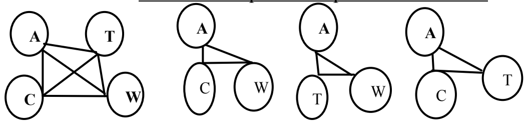


Figure 3: Equivalence class and Uniform Clique clustering

Here, two cliques and equivalence class are generated and are distributed on the processors to achieve equal load balancing. It is clear that this algorithm uses static load balancing and distributes the load among processor without having knowledge of its speed i.e. purely homogeneous system.

As compared to Count Distribution and Candidate Distribution parallel algorithm for association rule mining, Par-MaxClique algorithm outperforms because it utilizes the aggregate memory of the parallel system, decouples the processors right in the beginning by repartitioning the database so that each processor can compute independently, use vertical database layout which clusters the transactions containing an itemset into tid-list without scanning the database and computes the frequent itemsets by simple intersections on two tid-lists without having an overhead of maintaining complex data structures[2].

Inspite of this it has limitation that it uses static load balancing for homogeneous system which is far from reality because a database server has multiple systems with different configurations and speeds. This needs dynamic load balancing schemes.

We try to deal with the problem of parallel mining of the association rules in such a heterogeneous environment where there is no prior knowledge of the processing speeds of the processors in the system.

3. PROPOSED ALGORITHM

In the proposed algorithm, we initially distribute the load among the processors assuming that they are of the same speed since we don't have prior knowledge of speed of the processors in a system. During the execution it seems that the speed of one is faster than the other as it executes the job assigned to it and sits idle while other one is still busy in executing the job assigned to it. In that case faster processor will retrieve data from the slower one and executes the new assigned task. It is version of Par-MaxClique algorithm for heterogeneous system.

In this algorithm we consider the clusters where host act as a scheduler and assigns jobs/ task to the processors within it.

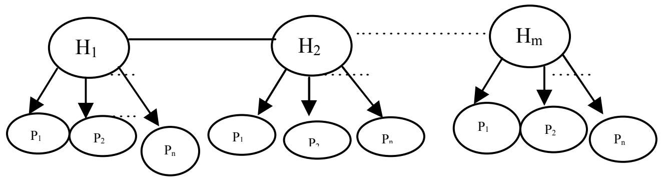
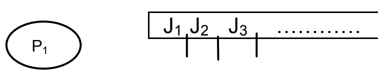


Figure 4: $H_1,H_2,..H_m$ represents m number of host and $P_1,P_2,..P_n$ represents the n number of processors attached to each cluster

Let us assume that there are m number of host and n number of processors attached to each host. Here, database is equally partitioned among the entire host in the system. Initially load balancing is done in each processor using Par-MaxClique algorithm for homogeneous system i.e. all processors gets equal work load. We also maintain job queue with each processor which keeps track of the number of jobs assigned to the processor for execution and also maintain a linked list at the host i.e. scheduler end that keep tracks of the loads of all the nodes in a cluster. This list is adjusted whenever job is scheduled at a node or job completes at the assigned node. The load of a processor is the number of jobs in its job queue.



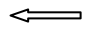


Figure 5: Job queue of processor

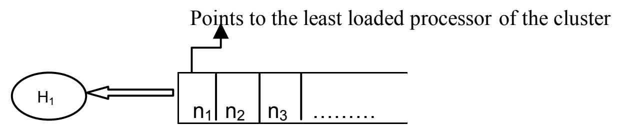


Figure 6: Linked list maintained by the host/scheduler

Table 1: Pseudo code for parallel association rule mining algorithm for heterogeneous environment

```
Begin
/*  Initialization Phase  */

 1.   Generate $L_2$ from 2-itemset support counts
 2.   Generate clusters from $L_2$ using uniform hypergraph cliques
 3.   Partition clusters among the processors
 4.   Scan local database partition
 5.   Transmit relevant tid-list to other processors
 6.   Received tid-list from other processors.
 7.   First, we compute the job queue and linked list of each processor and scheduler respectively. Initially, all processor have the same load value since jobs are equally distributed among the processors as in Par-MaxClique algorithm for homogeneous system.

/* Asynchronous Phase */
 8.   For each assigned cluster $C_2$, compute Frequent Itemsets

 9.   During execution, each processor updates its job queue and the linked list at the scheduler is also gets updated accordingly.

/* Communication OR Complete and offer Phase */
 10.  If job queue of all processors are empty then go to step 13
 11.  Else

 Begin
 The scheduler compares the load value of all the processors within the cluster and if any difference is found then perform the following :-
 (i)   Job from heavy loaded processor say $P_i$ is taken and gets assigned to least loaded processor say $P_j$.
 (ii)  Job queues of the $P_i$ and $P_j$ are adjusted accordingly.
 (iii) The link list at the scheduler is also adjusted accordingly.
 12.  Go to asynchronous phase i.e. step 8.
 End

/* processing completes at each processor and then comes final or reduction phase */
 13.  Aggregate Results and Output Associations
 14.  STOP
```

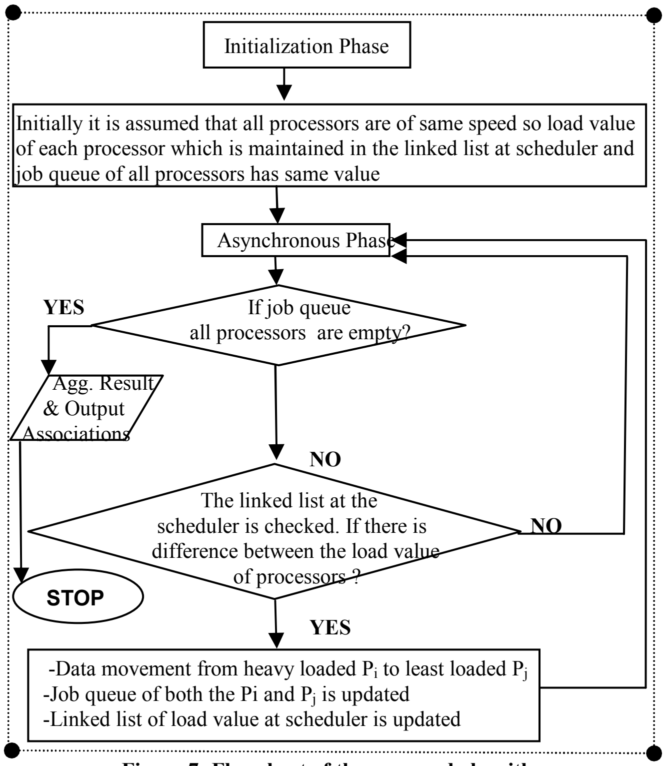

Initialization Phase
Initially it is assumed that all processors are of same speed so load value of each processor which is maintained in the linked list at scheduler and job queue of all processors has same value
Asynchronous Phase
YES
If job queue all processors are empty?
Agg. Result & Output Associations
NO
The linked list at the scheduler is checked. If there is difference between the load value of processors ?
NO
STOP
YES
-Data movement from heavy loaded P_i to least loaded P_j
-Job queue of both the Pi and P_j is updated
-Linked list of load value at scheduler is updated

Figure 7: Flowchart of the proposed algorithm

4. ANALYSIS OF PROPOSED ALGORITHM

The initialization phase and asynchronous phase remains same as for Par-MaxClique algorithm for homogeneous system [4] that uses static load balancing scheme. Because this algorithm utilizes the aggregate memory of the parallel system, decouples the processors right in the beginning, uses vertical database layout for clustering by performing simple intersection on two tid-lists and avoid the generation of all the subsets of a transaction and checking them against the candidate hash tree during support counting [2]. And also, since we are not aware of the processing speeds of the processors prior execution so load balance is same as in the case of Par-MaxClique parallel association rule mining algorithm for homogeneous system initially.

In the communication step i.e. complete and offer phase the processing speeds of processors are compared and checked by maintaining the linked list at the scheduler and the job queue at each of the processor respectively. If a difference in the load value of the processors is found, it means there is speed mismatch and immediately the work is migrated from heavy loaded processor to the least loaded one. The decrease in the processing speed of the processor in the system won't affect the efficiency of the whole system as its work gets executed by the least loaded processor or processor with empty job queue.

It assigns job to the processors in a cluster dynamically and thus maintains the load balance in a heterogeneous system. There is minimum communication and data movement involved between the processors because it's a scheduler who checks the load balance of the processors and involved in the data movement between the desired processors. This minimizes the inter processors communications.

5. CONCLUSION

In this algorithm, we raised the issue of parallel association rule mining in heterogeneous environment where no prior knowledge of relative processing speeds of the processors are known or say assumed. It has good features of the Par-MaxClique parallel association rule mining algorithm for homogeneous environment which outperforms count distribution, data distribution and candidate distribution algorithm for parallel association rule mining. It also exploits vertical database layout, asynchronous counting process and dynamic load balancing technique that leads to efficient utilisation of the processors and enhance the performance of the heterogeneous system with minimum communication cost involvement.

In future, we try to perform dynamic load balancing in between the clusters and minimize the communication delay to enhance the performance of the system and will do detailed analysis of the proposed algorithm. Also we will try to incorporate fault tolerance in the system.

6. REFERENCES

[1] Masahisa Tamura and Masaru Kitsuregawa, "Dynamic Load Balancing for Parallel Association Rule Mining on Heterogeneous PC Cluster System", Proceedings of the 25th VLDB Conference, Edinburgh, Scotland, 1999, pp. 163.

[2] Mohammed J. Zaki, Srinivasan Parthasarthy, Mithsunori Ogihara and Wei Li, "Parallel Algorithms for Discovery of Association Rules", Data Mining and knowledge Discovery, © 1997 Kluwer Academic Publishers, pp. 360, 364.

[3] M. J. Zaki, S. Parthasarathy and W. Li., "Parallel data mining for association rules on shared memory multi-processors". In Supercomputing96, November 1996, Pittsburg, pp. 17-22.

[4] M. J. Zaki, "Parallel and Distributed Association Mining: A Survey", IEEE, 1999, pp. 2-3, 10-13.

[5] R. Agrawal and J. Shafer, "Parallel mining association rules", IEEE Trans. On Knowledge and Data Engineering, December 1996, 8(6): pp. 962-969.

[6] E-H. Han, G. Karypis and Vipin Kumar, "Scalable parallel data mining for association rules", In ACM SIGMOD Conf. Management of Data, May 1997, pp. 279-284.

[7] T. Shintani and M. Kitsuregawa, "Hash based parallel algorithms for mining association rules". In 4th Intl. conf. Parallel and Distributed Info. Systems, December 1996, pp. 20-25.

[8] D. Cheung and Y. Xiao, "Effect of data skewness in parallel mining of association rules", in 2nd Pacific-Asia Conference on Knowledge Discovery and Data Mining, April 1998, pp. 51-55.

[9] Arun k. Pujari, A Book on Data mining techniques, Universities Press (India) Ltd., Hyderabad, 2001, First edition, pp. 94-100.

[10] Mohammed J. Zaki, "Parallel and Distributed Data Mining: An Introduction", C.-T. Ho (Eds.): Large-Scale Parallel Data Mining, ©Springer-Verlag Berlin Heidelberg 2000. LNAI 1759, pp. 9.

[11] Kun-Ming Yu, Jiayi Zhou and Wei Chen Hsiao, "Load Balancing Approach Parallel Algorithm for Frequent Pattern Mining", V. Malyshkin (Ed.): PaCT 2007. © Springer-Verlag Berlin Heidelberg 2007.LNCS 4671, pp. 623–631.

[12] Jochen Hipp, Ulrich G¨untzer, Gholamreza Nakhaeizadeh, "Algorithms for Association Rule Mining – A General Survey and Comparison", SIGKDD explorations copyright © 2000, ACM SIGKDD, July 2000, Volume 2, Issue 1, pp. 58-61.

[13] Sotiris kotsiantis, Dimitris kanellopoulos, "Association Rule Mining: A Recent Overview", GESTS International Transactions on Computer science and Engineering, Vol. 32 (1), 2006, pp. 73.

[14] Masaru Kitsuregawa and Takahilus Shintani, Masahisa Tamura and Iko Pramudiono, "Parallel Data Mining on large scale PC Cluster", H. Lu and A. Zhou (Eds.): WAIM 2000, © Springer-Verlag Berlin Heidelberg 2000. LNCS 1846, pp. 15–26, 2000