Parallel Data Mining on Large Scale PC cluster

Masaru Kitsuregawa, Takahiko Shintani, Masahisa Tamura, Iko Pramudiono

Institute of Industrial Science, The University of Tokyo
7-22-1, Roppongi, Minato-ku, Tokyo 106, Japan
{kitsure,shintani,masahisa,iko}@tkl.iis.u-tokyo.ac.jp

Abstract. PC cluster is recently regarded as one of the most promising platforms for heavy data intensive applications, such as decision support query processing and data mining. We proposed some new parallel algorithms to mine association rule and generalized association rule with taxonomy and showed that PC cluster can handle large scale mining with them. During development of high performance parallel mining system on PC cluster, we found that heterogeneity is inevitable to take the advantage of rapid progress of PC hardware. However we can not naively apply existing parallel algorithms since they assume homogeneity. We proposed the new dynamic load balancing methods for association rule mining, which works under heterogeneous system. 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. The experimental results confirm that the proposed approach can very effectively balance the workload among heterogeneous PCs.

1 Introduction

Recently commodity based PC cluster system is regarded as one of the most promising platforms for data intensive applications such as decision support query processing and data mining. The power of PC is superior to the workstation for integer performance and the price of PC is also much lower. So far extensive researches on parallel database processing algorithms have been done[3]. Currently parallel execution option is available for most of RDB products. Parallel engine is essential for large-scale data warehouse and is becoming popular nowadays. Thus combining the above two key trends, namely, parallel database processing on PC cluster would be a most cost-effective solution for large scale data warehousing.

We have built 100 node PC cluster system named NEDO-100 for data base applications. We implemented parallel RDB kernel on it. TPC-D benchmark and association rule mining were run on the machine [6, 16] and, it showed sufficiently high performance. We exemplified that the PC cluster can achieve considerably high cost-performance ratio.

We implemented high performance association rule mining system on the PC cluster. We have enhanced it with optimization to mine generalized association rule. In generalized association rules, application-specific knowledge in the form of taxonomies (*is-a* hierarchies) over items are used to discover more interesting rules. We introduce new parallel mining algorithms by taking the classification hierarchy into account[15]. Here we show that our system can handle large amount of transactions(1GBytes). [13]

One problem we faced in that project is "heterogeneity." The system we built[6, 16] was completely uniform. However, when we planned to increase the number of nodes, it was extremely difficult to find out the same machines. Since the development period of PC is very short, configuration of machines is changing so quickly. Once six months have passed, we have to introduce different type of PCs. Thus heterogeneity is inevitable.

Most of the parallel algorithms developed so far assume the system be uniform. Very few papers address heterogeneity problem[2]. If we apply the parallel algorithm developed for uniform parallel system to the heterogeneous environment, apparently we will see significant performance deterioration. A high performance node can process its allocated task quickly but node with less powerful processor or with low bandwidth disk usually takes longer time to finish. We picked up data mining as a data intensive application and tried to solve the heterogeneity problem.

In [5] we propose run time load balancing algorithms for association rule mining under heterogeneous PC cluster environment. Two strategies named candidate migration and transaction migration are developed. Details on these two will be given in later sections. PCs do not have to communicate each other before the execution in order to normalize the performance among different CPUs and disks etc. During executing data mining, the workload of each node is monitored autonomously and the system performance is controlled to be balanced by migrating candidates/transaction among nodes at runtime.

Section 2 explains the NEDO-100 PC cluster system. Section 3 briefly explains the association rule mining. Section 4 describes parallel algorithms for Apriori and for generalized association rules. Section 5 introduces the fundamental idea of load balancing for association rule mining. Section 6 discusses the future work and concludes the paper.

2 NEDO-100 PC Cluster

We have developed a large scale PC cluster consists of 128 PCs interconnected with 155 Mbps ATM and 10 Mbps Ethernet networks[6, 16]. The project was launched in 1995 and the equipments came at the end of 1996. The system started at February 1997.

Initially the PC cluster was made up of 100 PCs with 200 MHz Pentium Pro only and then we have added another 8 nodes but with more powerful 333 MHz Pentium II and 20 nodes with 450 MHz Pentium II since the performance of PC hardware had improved dramatically.

The configuration of each PC showed in Table 1. The details of the development of this system has been written elsewhere. [6, 16] We have performed

Table 1. Configuration of each PC

Node#	$1 \sim 100$	$101 \sim 119$	$120 \sim 127$
CPU	Pentium Pro 200MHz	PentiumII 450 Mhz	PentiumII 333 MHz
Main Memory	64MB	256MB	64MB
Disk Drive for databases	Seagate Barracuda (Ultra SCSI, 4.3GB) Seagate Cheetah (Ultra SCSI, 9.1GB)	IBM DTTA-371440 (EIDE, 14.4GB)	Seagate Cheetah (Ultra SCSI, 9.1GB)
ATM NIC	Interphase 5515 PCI ATM Adapter		
OS	Solaris2.5.1 for x86	Solaris2.6 for x86	Solaris2.6 for x86

numerous experiments of data intensive applications on the system that prove the applicability of this system. Remarkably, since the system use low-cost commodity PC, it offers extremely good cost performance compared to mainframe based database system currently in the market.

Some basic performance measurement of NEDO-100 are: point-to-point throughput using TCP/IP protocol over ATM exceeds 110Mbps with message size 8KB-16KB, roundtrip latency measured to be 448μs and disk read throughput about 8 MB per second.

3 Association Rule Mining

3.1 Association Rule

An example of an association rule is " if a customer buys A and B then 90% of them buy also C " . Here 90% is called the *confidence* of the rule. Another measure of a rule is called the *support* of the rule that represents the percentage of transactions that contain the rule.

The problem of mining association rules is to find all the rules that satisfy a user-specified minimum support and minimum confidence, which can be decomposed into two subproblems:

1. Find all combinations of items, called large itemsets, whose support is greater than minimum support.
2. Use the large itemsets to generate the rules.

Here we briefly explain the Apriori algorithm for finding all large itemsets, proposed in [9].

In the first pass, support count for each item is incremented by scanning the transaction database. All items that satisfy the minimum support are picked out. These items are called large 1-itemset. Here k-itemset is defined as a set of k items. In the k-th pass, the candidate k-itemsets are generated using set of large $k-1$-itemsets. Then the support count of the candidate k-itemsets is incremented by scanning the transaction database. At the end of scanning the transaction data, the large k-itemsets which satisfy minimum support are determined. The process is repeated until no candidate itemsets generated.

3.2 Generalized Association Rule with Taxonomy

In most cases, items can be classified according to some kind of "is a" hierarchies. [11] For example "Sushi is a Japanese Food" and also "Sushi is a Food" can be expressed as taxonomy. Here we categorize sushi as descendant and Japanese food and food are its ancestors. By including taxonomy as application specific knowledge more interesting rules can be discovered.

Cumulate algorithm [11] is the first algorithm to mine generalized association rule mining. It is based on Apriori algorithm, and it is extended with optimizations that make use of the characteristics of generalized association rule such as pruning itemsets containing an item and its ancestors at second pass and pre-computing the ancestors of each item.

4 Highly Parallel Data Mining

4.1 Parallel Association Rule Mining

J.S.Park, et.al proposed bit vector filtering for association rule mining and naive parallelization of Apriori [10, 8], where every node keeps the whole candidate itemsets and scans the database independently. Communication is necessary only at the end of each pass. Although this method is very simple and communication overhead is very small, memory utilization efficiency is terribly bad. Since all the nodes have the copy of all the candidate itemsets, it wastes memory space a lot.

In [14] Hash Partitioned Apriori(HPA) was proposed in 1996. The candidate itemsets are not copied over all the nodes but are partitioned using hash function. Then each node builds hash table of candidate itemsets. The number of itemsets at second pass is usually extremely high, sometimes three orders of magnitude larger than the first pass in a certain retail transaction database which we examined. When the user-specified support is low, the candidate itemsets overflow the memory space and incur a lot of disk I/O.

While reading transaction data for support counting, HPA applies the same hash function to decide where to send the transactions and then probe the hash table of candidate itemsets to increase the count. Although it has to exchange transaction data among nodes, utilization whole memory space through partitioning the candidates over nodes instead of duplication results in better parallelization gain.

Hybrid approach between candidate duplication and candidate partitioning is proposed at [4] at 1997. The processors are divided into some number of groups. Within each group, all the candidates are duplicated and among groups, candidates are partitioned.

4.2 Parallel Algorithms for Generalized Association Rule Mining

In this subsection, we describe our parallel algorithms for finding all large itemsets on shared-nothing environment proposed in [15].

Non Partitioned Generalized association rule Mining : NPGM NPGM copies the candidate itemsets over all the nodes. Each node can work independently.

Hash Partitioned Generalized association rule Mining : HPGM HPGM partitions the candidate itemsets among the nodes using a hash function like in the hash join, which eliminate broadcasting.

Hierarchical HPGM : H-HPGM H-HPGM partitions the candidate itemsets among the nodes taking the classification hierarchy into account so that all the candidate itemsets whose root items are identical be allocated to the identical node, which eliminates communication of the ancestor items. Thus the communication overhead can be reduced significantly compared with original HPGM.

H-HPGM with Fine Grain Duplicate: H-HPGM-FGD In the case the size of the candidate itemsets is smaller than available system memory, H-HPGM-FGD utilizes the remaining free space. H-HPGM-FGD detects the frequently occurring itemsets which consists of the any level items. It duplicates them and their all ancestor itemsets over all the nodes and counts the support count locally for those itemsets like in NPGM.

4.3 Transaction Dataset

We use synthetic transaction data generated using procedure in [9]. For large scale experiments of generalized association rules we use the following parameters : (1)the number of items is 50,000, the number of roots is 100, the number of levels is 4–5, fanout is 5, (2)the total number of transactions is 20,000,000(1GBytes), the average size of transactions is 5, and (3)the number of potentially large itemsets is 10,000.

4.4 Performance Evaluation Results

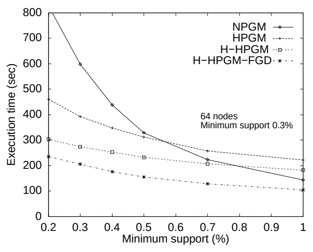


Fig. 1. Execution time

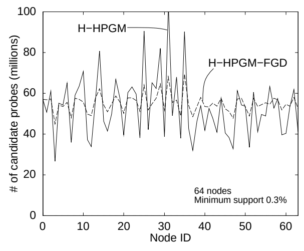


Fig. 2. Candidate probes

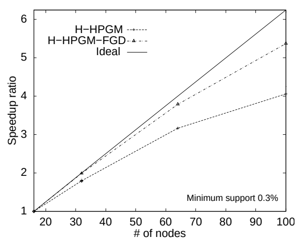


Fig. 3. Speedup ratio

We show the execution time at pass 2 of all parallel algorithms varying the minimum support in Figure 1. The execution time of all the algorithms increases when the minimum support becomes small. When the minimum support is small, the candidate partitioned methods can attain good performance. H-HPGM-FGD significantly outperforms other algorithms.

Next, the workload distribution of H-HPGM and H-HPGM-FGD is examined. Figure 2 shows the number of candidate probes to increment the support count in each node at pass 2. In H-HPGM, the distribution of the number of probes is largely fractured, since the candidate itemsets are partitioned in the unit of hierarchy of the candidate itemsets. H-HPGM-FGD detects the frequently occurring candidate itemsets and duplicate them. The support counting process for these duplicated candidate itemsets can be locally processed, which can effectively balance the load among the nodes.

Figure 3 shows the speedup ratio with varying the number of nodes used 16, 32, 64 and 100. The curves are normalized by the execution time of 16 nodes system. H-HPGM-FGD attains higher linearity than H-HPGM. Since H-HPGM duplicates no candidate itemsets, the workload skew degrades the linearity. The skew handling methods detect the frequently occurring candidate itemsets and duplicate them so that the remaining free memory space can be utilized as much as possible. In Figure 3, H-HPGM-FGD achieves good performance on one hundred nodes system.

5 Dynamic load balancing on heterogeneous PC cluster

5.1 Run Time Load Balancing Methods

The parallel algorithms so far proposed assume homogeneous parallel processing environment. In [5], we propose dynamic load balancing algorithms for heterogeneous parallel systems, where each node might have different type of CPU, and different kinds of disks, etc. We choose "flat" association rule mining based on HPA rather than generalized one as the application to give clearer insight on how they work.

As described in the section 4, HPA sends each node the itemsets and probes them against its own candidate itemsets hash table. [14] If a node is assigned more candidate itemsets, it will receive more itemsets from other nodes during counting phase. This means that we can adjust the workload of each node by adjusting the amount of candidate itemsets. If the load of a certain node is higher than the other nodes, we take some of the candidate itemsets from that node and give them to the other nodes. Then the itemsets that are originally directed to that node are now redirected to the new nodes to which the removed itemsets are relocated. Thus the counting workload is migrated from the original node to the other nodes. We name this strategy Candidate Migration.

The Candidate Migration is possible if the node still has candidate itemsets to be migrated. itemsets. If the skew is high, there arises the case where migrating all the candidates is still not sufficient. In order to handle such situation, we need yet another strategy to migrate workload.

Let's examine the HPA algorithm again in more detail. Each node has two major task. One is to receive the itemset sent from other nodes, probe it against the hash table and increment the count corresponding to that itemset. The other task is to read the transactions from the disk, generate the itemsets and send them to the nodes determined using hash function. We use the former task for Candidate Migration.

Now we consider the latter task. Actually, the itemset generation from transactions is rather complicated process. This workload could be migrated. The node with heavy workload reads the transactions from the disk and it does not do itemset generation itself but just sends the transactions to the light nodes. We name this strategy Transaction Migration.

Transaction Migration incurs overhead of network transfer for each transaction. Thus, we put priority to the Candidate Migration. Initially heavy node migrates candidate itemsets only. When there are no candidate itemsets remained to migrate, then it migrates transactions.

5.2 Migration Plan Derivation

We propose dynamic load balancing methods during the execution of data mining to cope with the skew in heterogeneous system. In this approach, a coordinator node collects necessary information from all the nodes, calculate estimated remaining processing time for that node $\mathrm{rest}T_i$ and controls the distribution of workload.

Since the goal is to have all nodes complete their job at the same time, our method dynamically controls the load allocated for each node so that every node has the same $\mathrm{rest}T_i$.

The skew is defined as follow,

$$skew = \frac{\max(\mathrm{rest}T_i) - \min(\mathrm{rest}T_i)}{\mathrm{avg}(\mathrm{rest}T_i)} \tag{1}$$

$$\begin{cases} skew \leq threshold : no\ skew \\ skew > threshold : skew\ exists \end{cases} \tag{2}$$

Coordinator can judge that the load control is needed if this value exceeds some certain threshold. Then it makes a plan for Candidate Migration If skew still presents, it also creates another plan for Transaction Migration. The above procedure is periodically invoked. Coordinator checks the skew condition every fixed time interval. The complete load balancing is difficult by any means. Error gradually accumulates. Once it becomes beyond the threshold, the coordinator tries to balance the workload again.

5.3 Experimental Environment and Transaction Dataset

In order to simplify the problem and to show clearly the effectiveness of our load balancing approach on heterogenous environment, we have made performance evaluation on a group of four nodes each with different CPU power, disk performance and data distribution as shown in table 2. The parameters used are described in table 3. In practice, we are forced to mine database in various situation, so data distribution is skewed. We put least amount of data to node 4 while employing fast microprocessor in order to artificially generate skew. Apparently experiment with dataset 2 has higher skew than that with dataset 1. This is used for transaction migration experiments. And in all of the experiments, the *skew* value was set to 0.2.

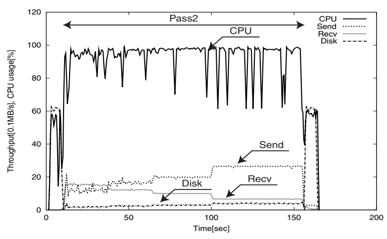
Table 2. Configuration for heterogenous experiments

	Node 1	Node 2	Node 3	Node 4
Proc.	P.Pro	P.Pro	P.Pro	P.II
Clock	200MHz	200MHz	200MHz	333MHz
Disk	SCSI	SCSI	SCSI	IDE
DataSet1	40MB	20MB	10MB	10MB
DataSet2	80MB	20MB	10MB	10MB

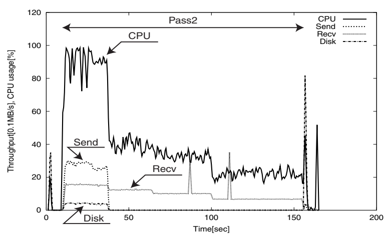
Table 3. Datasets for heterogenous experiments

	DataSet1	DataSet2
Number of transactions	1000000	1500000
Avg. size of transactions	20	20
Number of items	5000	5000

5.4 Performance Evaluation Results

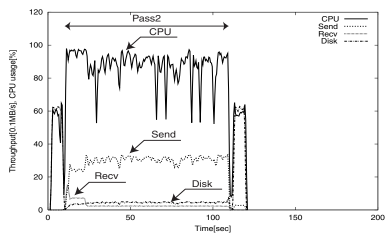


(a)Node1

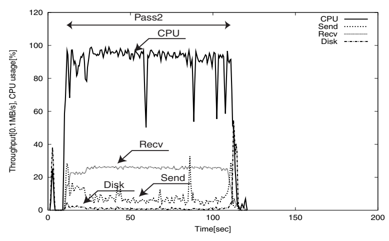


(b)Node4

Fig. 4. Execution trace without load balancing (DataSet1)



(a)Node1



(b)Node4

Fig. 5. Execution trace with Candidate Migration (DataSet1)

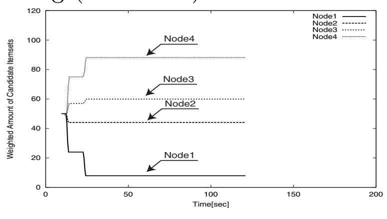


Fig. 6. Migration trace for weighted amount of candidate itemsets(DataSet1)

Table 4. Number of candidate itemset and large itemset for DataSet1 and DataSet2

	DataSet1		DataSet2	
	C	L	C	L
Pass 1	5000	989	5000	982
Pass 2	488566	54	481671	51
Pass 3	42	4	38	4
Pass 4	0	0	0	0

Heterogenous configuration experiment with Dataset 1 for candidate migration The numbers of candidate itemsets(C) and large itemsets(L) resulted from data mining of dataset 1 with 0.7% minimum support are shown in table 4. The execution traces without any load migration are shown in Figure 4. The figure shows four different resource usage: CPU, disk, interconnection network (send/receive). Horizontal axis is elapsed time and vertical axis denotes utilization ratio for CPU and data transfer throughput for disk read and interconnection network. The network throughput is divided into two parts, send throughput and receive throughput.

We only show traces for Node 1 and Node 4 since the space is limited. In the first half of second pass Node 1 is too busy receiving k-itemsets from other nodes, and could not even afford to read its own transaction data. On the other hand, Node 4 with more powerful CPU and less data finishes reading its data in first 40 seconds and idles for the rest of time. The total execution time is 164.03 seconds.

When we apply Candidate Migration strategy, candidate itemsets are reallocated as soon as skew is detected. The traces are shown in Figure 5. Every node completes its task at almost the same time indicating the skew is eliminated and workload is evenly distributed. The processing time is also greatly improved to only 120.21 seconds.

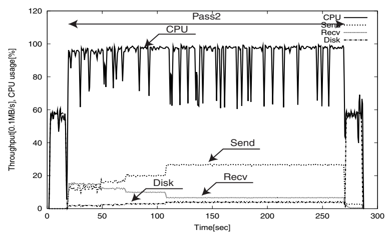
Figure 6 shows the trace of weighted candidate itemsets of all the nodes. We can see that Node 1 and Node 2 migrate their candidate itemsets to Node 3 and Node 4. The amount of migrated itemsets gradually increases and finally converged to a certain value.

Heterogenous configuration experiment with Dataset 2 for both candidate migration and transaction migration We did an experiment with more skewed environment using dataset 2. Result of data mining using dataset 2 and 0.7% minimum support is also shown in table 3. Node 1 is becoming the bottleneck of the parallelization as shown in Figure 7. The total execution time is 287.09 seconds.

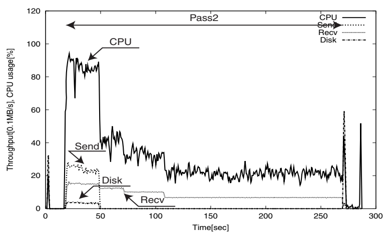
By introducing the Candidate Migration, performance can be improved. The processing time is reduced to 198.36 seconds. However since the load is extremely concentrated at Node 1, as Figure 8 shows, Candidate Migration alone can not get rid of that skew completely. Node 4 finishes reading out the transactions from disk at around 125 seconds. We can see that all candidate itemsets of Node 1 has been transferred to other nodes, as shown at Figure 11. Thus Candidate Migration can not migrate workload any more.

When we introduce Transaction Migration in addition to Candidate Migration, we can achive almost perfect load balancing as shown at Figure 9. Node 1 sends its transaction data and delegates the generation of k-itemsets to other nodes. The elimination of skew records processing time of 182.18 seconds.

Figure 10 shows the trace of amount of weighted candidate itemset and amount of migrated transactions for Node 1 and Node 4. No candidate itemsets is left at Node 1. Node 1 also sends out transactions to the other nodes and Node 4 receives some of the transactions from Node 1.

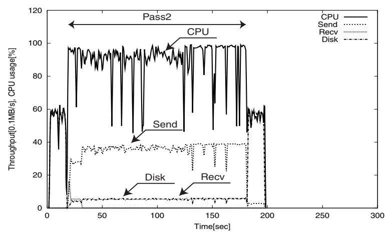


(a)Node1

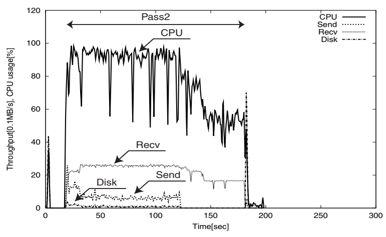


(b)Node4

Fig. 7. Execution trace without any load balancing(DataSet2)

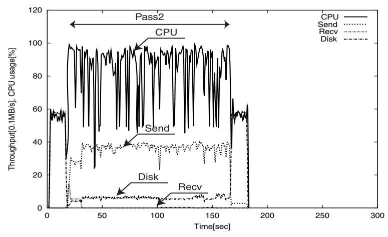


(a)Node1

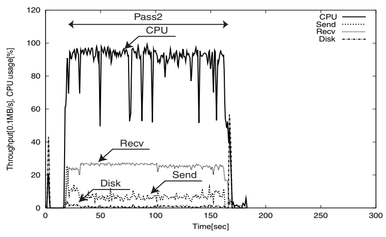


(b)Node4

Fig. 8. Execution trace with Candidate Migration(DataSet2)

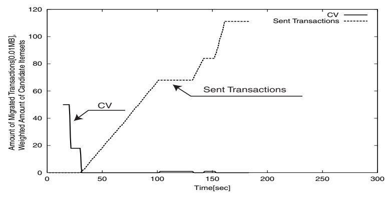


(a)Node1

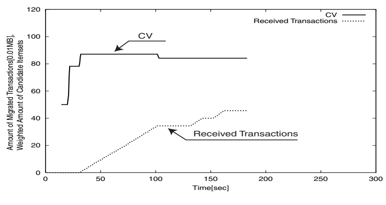


(b)Node4

Fig. 9. Execution trace with both Candidate and Transaction Migration(DataSet2)



(a)Node1



(b)Node4

Fig. 10. Migration trace of weighted candidate itemsets and transactions(DataSet2)

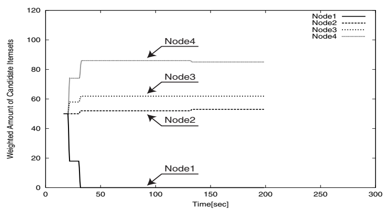


Fig. 11. Migration trace for weighted candidate itemsets (DataSet2)

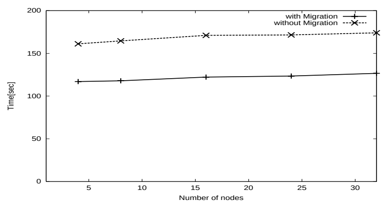


Fig. 12. Scale-up results(DataSet1)

Experiment for scalability of proposed load balancing method We scaled up the system by multiplying the configuration we used so far. We used the configuration of a group of 4 nodes as multiplication unit and expanded the system from 4 nodes to 8, 12, 16, 24 and 32 nodes.

The results are shown in Figure 12. Execution time increases slightly as the number of nodes increases. As the number of nodes increases, the overhead time for synchronization becomes non-negligible.

6 Conclusion

We examined the effectiveness of parallel algorithms on large scale parallel computer system using the large amount of transaction dataset. Our system is consisted with one hundred of PCs. As far as the authors know, there has no research on parallel data mining over such large scale systems using a large amount of transaction dataset. Through several experiments, we showed H-HPGM-FGD could attain sufficiently high performance and achieve good workload distribution on one hundred PC cluster system.

We proposed dynamic load balancing strategies for parallel association rule mining on heterogeneous PC cluster system. Due to the short development period of recent PCs, it is inevitable that the PC cluster system becomes heterogeneous. In order to utilize all the system resources as fully as possible, we have to make the program adaptive to its runtime environment.

We adopted HPA(Hash Partitioned Apriori) algorithm for underlying parallel association rule mining. We proposed two kinds of dynamic load balancing strategies for parallel association rule mining, Candidate Migration and Transaction Migration.

In order to clearly show the effectiveness of our approach, we set up rather simple 4 node cluster with two kinds of PCs and varied the size of dataset for each PC. We demonstrated the feasibility of our approach showing the execution trace. By examining the trace, we confirmed that the proposed scheme effectively works to remove workload inbalance. Candidate Migration works under medium skew environment. If the skew is high and candidate migration can not sufficiently help, the system automatically invokes the Transaction Migration. In addition, we also showed the scalability experiments. We increased the size of the system from 4 nodes to 32 nodes. We found sufficient scalability can be archived.

Our experiments have showed that PC cluster, with its scalable performance and high cost performance is a promising platform for data intensive applications such as data mining.

References

1. D.W. Cheung, J. Han, V.T. Ng, A.W. Fu, and Y. Fu. “A Fast Distributed Algorithms for Mining Association Rules.” In *Proc. of PDIS*, pp. 31–42, Dec. 1996.
2. H. M. Dewan, M. A. Hernandez, K. W. Mok, S. J.Stolfo “Predictive Dynamic Load Balancing of Parallel Hash-Joins Over Heterogeneous Processors in the Presence of Data Skew.” In *Proc. of PDIS*, pp. 40–49, 1994.
3. D. DeWitt and J. Gray “Parallel Database Systems: The Future of High Performance Database Systems.” In *Communications of the ACM*, Vol. 35, No. 6, pp. 85–98, Jun. 1992.
4. E.-H.Han and G.Karypis and Vipin Kumar ”Scalable Parallel Data Mining for Association Rules.” In *Proc. of SIGMOD*, pp. 277–288, May. 1997
5. M. Tamura, M.Kitsuregawa. ”Dynamic Load Balancing for Parallel Association Rule Mining on Heterogeneous PC Cluster System”. In *Proc. of VLDB*, 1999.
6. M. Kitsuregawa, T. Tamura, M. Oguchi “Parallel Database Processing/Data Mining on Large Scale ATM Connected PC Cluster.” In *Euro-PDS*, pp. 313–320, Jun. 1997
7. M.J.Zaki, S.Parthasarathy, M.Ogihara and W.Li “Parallel Algorithms for Discovery of Association Rules”. Data Mining and Knowledge Discovery, Dec. 1997.
8. J.S.Park, M.-S.Chen, P.S.Yu ”Efficient Parallel Algorithms for Mining Association Rules” In *Proc. of CIKM*, pp. 31–36, Nov. 1995
9. R. Agrawal and R. Srikant. ”Fast Algorithms for Mining Association Rules”. In *Proc. of VLDB* , pp. 487–499, Sep. 1994.
10. R. Agrawal and J. C. Shafer. “Parallel Mining of Associaton Rules”. In *IEEE TKDE*, Vol. 8, No. 6, pp. 962–969, Dec. 1996.
11. R. Srikant, R. Agrawal. ”Mining Generalized Association Rules”. In *Proc. of VLDB*, 1995.
12. S.Parthasarathy and M.J.Zaki and W.Li “Memory Placement Techniques for Parallel Association Mining.” In *Proc. of KDD*, pp. 304–308, Aug. 1998
13. T.Shintani, M. Oguchi, M.Kitsuregawa. ”Performance Analysis for Parallel Generalized Association Rule Mining on a Large Scale PC Cluster”. In *Proc. of Euro-par*, 1999.
14. T. Shintani and M. Kitsuregawa “Hash Based Parallel Algorithms for Mining Association Rules”. In *Proc. of PDIS*, pp. 19–30, Dec. 1996.
15. T. Shintani, M. Kitsuregawa “Parallel Mining Algorithms for Generalized Association Rules with Classification Hierarchy.” In *Proc. of SIGMOD*, pp. 25–36, 1998.
16. T. Tamura, M. Oguchi, M. Kitsuregawa “Parallel Database Processing on a 100 Node PC Cluster: Cases for Decision Support Query Processing and Data Mining.” In *Super Computing 97::High Performance Networking and Computing*, 1997
17. Y. Xiao and D. W. Cheung “Effect of Data Skewness in Parallel Data Mining of Association Rules ”. In *Proc. of PAKDD*, pp. 48–60, Apr. 1998.

This article was processed using the LaTeX macro package with LLNCS style