Parallel Data Mining on ATM-Connected PC Cluster and Optimization of its Execution Environments

Masato OGUCHI[1,2] and Masaru KITSUREGAWA[1]

[1] Institute of Industrial Science, The University of Tokyo
7-22-1 Roppongi, Minato-ku Tokyo 106-8558, Japan
[2] Informatik4, Aachen University of Technology
Ahornstr.55, D-52056 Aachen, Germany
oguchi@tkl.iis.u-tokyo.ac.jp

Abstract. In this paper, we have constructed a large scale ATM-connected PC cluster consists of 100 PCs, implemented a data mining application, and optimized its execution environment. Default parameters of TCP retransmission mechanism cannot provide good performance for data mining application, since a lot of collisions occur in the case of all-to-all multicasting in the large scale PC cluster. Using a TCP retransmission parameters according to the proposed parameter optimization, reasonably good performance improvement is achieved for parallel data mining on 100 PCs.
Association rule mining, one of the best-known problems in data mining, differs from conventional scientific calculations in its usage of main memory. We have investigated the feasibility of using available memory on remote nodes as a swap area when working nodes need to swap out their real memory contents. According to the experimental results on our PC cluster, the proposed method is expected to be considerably better than using hard disks as a swapping device.

1 Introduction

Looking over the recent technology trends, PC/WS clusters connected with high speed networks such as ATM are considered to be a principal platform for future high performance parallel computers. Applications which formerly could only be implemented on expensive massively parallel processors can now be executed on inexpensive clusters of PCs. Various research projects to develop and examine PC/WS clusters have been performed until now[1][2][3]. Most of them however, only measured basic characteristics of PCs and networks, and/or some small benchmark programs were examined. We believe that data intensive applications such as data mining and ad-hoc query processing in databases are quite important for future high performance computers, in addition to the conventional scientific applications[4].

Data mining has attracted a lot of attention recently from both the research and commercial community, for finding interesting trends hidden in large transaction logs. Since data mining is a very computation and I/O intensive process,

parallel processing is required to supply the necessary computational power for very large mining operations. In this paper, we report the results on parallel data mining on ATM-connected PC clusters, consists of 100 Pentium Pro PCs.

2 Our ATM-connected PC cluster and its communication characteristics

We have constructed a PC cluster pilot system which consists of 100 nodes of 200MHz Pentium Pro PCs connected with an ATM switch. An overview of the PC cluster is shown in Figure 1. Each node of the cluster is equipped with 64Mbytes main memory, 2.5Gbytes IDE hard disk, and 4.3Gbytes SCSI hard disk. Solaris ver.2.5.1 is used as an operating system.

All nodes of the cluster are connected by a 155Mbps ATM LAN as well as an Ethernet. HITACHI's AN1000-20, which has 128 port 155Mbps UTP-5, is used as an ATM switch. Interphase 5515 PCI ATM adapter and RFC-1483 PVC driver, which support LLC/SNAP encapsulation for IP over ATM, are used. Only UBR traffic class is supported in this driver.

TCP/IP is used as a communication protocol. TCP is not only a very popular reliable protocol for computer communication, but also contains all functions as a general transport layer. Thus the results of our experiments must be valid even if other transport protocol is used, for investigating reliable communication protocols on a large scale cluster.

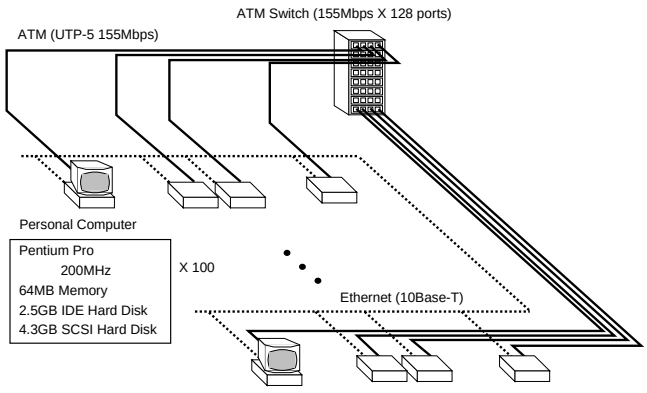

Fig. 1. An overview of the PC cluster

3 Parallel data mining application and its implementation on the cluster

3.1 Association rule mining

Data mining is a method of the efficient discovery of useful information such as rules and previously unknown patterns existing among data items embedded in large databases, which allows more effective utilization of existing data. One of the best known problems in data mining is mining of the association rules from a database, so called "basket analysis"[5][6]. Basket type transactions typically consist of transaction id and items bought per-transaction. An example of an association rule is "if customers buy A and B then 90% of them also buy C". The best known algorithm for association rule mining is Apriori algorithm proposed by R. Agrawal of IBM Almaden Research[7].

In order to improve the quality of the rule, we have to analyze very large amounts of transaction data, which requires considerably long computation time. We have studied several parallel algorithms for mining association rules until now[8], based on Apriori. One of these algorithms, called HPA(Hash Partitioned Apriori), is implemented and evaluated.

Apriori first generates candidate itemsets, then scans the transaction database to determine whether the candidates satisfy the user specified minimum support. At first pass (pass 1), a support for each item is counted by scanning the transaction database, and all items which satisfy the minimum support are picked out. These items are called large 1-itemsets. In the second pass (pass 2), 2-itemsets (length 2) are generated using the large 1-itemsets. These 2-itemsets are called candidate 2-itemsets. Then supports for the candidate 2-itemsets are counted by scanning the transaction database, large 2-itemsets which satisfy the minimum support are determined. This iterative procedure terminates when large itemset or candidate itemset becomes empty. Association rules which satisfy user specified minimum confidence can be derived from these large itemsets.

HPA partitions the candidate itemsets among processors using a hash function, like the hash join in relational databases. HPA effectively utilizes the whole memory space of all the processors. Hence it works well for large scale data mining. In the detail of HPA, please refer to [8][9].

3.2 Implementation of HPA program on PC cluster

We have implemented HPA program on our PC cluster. Each node of the cluster has a transaction data file on its own hard disk. Solaris socket library is used for the inter-process communication. All processes are connected with each other by socket connections, thus forming mesh topology. In the ATM level, PVC (Permanent Virtual Channel) switching is used since the data is transferred continuously among all the processes.

Transaction data is produced using data generation program developed by Agrawal, designating some parameters such as the number of transaction, the

Table 1. The number of candidate and large itemsets

C	the number of candidate itemsets
L	the number of large itemsets
T	the execution time of each pass [sec]

pass	C	L	T
pass 1		1023	11.2
pass 2	522753	32	69.8
pass 3	19	19	3.2
pass 4	7	7	6.2
pass 5	1	0	12.1

number of different items, and so on. The produced data is divided by the number of nodes, and copied to each node's hard disk.

The parameters used in the evaluation is as follows: The number of transaction is 10,000,000, the number of different items is 5000 and minimum support is 0.7%. The size of the transaction data is about 800Mbytes in total. The message block size is set to be 8Kbytes and the disk I/O block size is 64Kbytes.

The numbers of candidate itemsets and large itemsets, and the execution time of each pass executed on 100 nodes PC cluster are shown in Table 1. Note that the number of candidate itemsets in pass 2 is extremely larger than other passes, which often happens in association rules mining.

4 Optimization of transport layer protocol parameters

4.1 Broadcasting on the cluster and TCP retransmission

The execution times of pass $3 - 5$ are relatively long in Table 1, although they do not have large number of itemsets. At the end of each pass, a barrier synchronization and exchange of data are needed among all nodes, that is, all-to-all broadcasting takes place. Even if the amount of broadcasting data is not large, cells must be discarded at the ATM switch if timing of the broadcasting is the same at all nodes. Since pass $3 - 5$ have little data to process, actual execution time is quite short, thus broadcasting is performed almost simultaneously in all nodes, which tend to cause network congestion and TCP retransmission as a result. We have executed several experiments to find the better retransmission parameters setting suitable for such cases.

We use TCP protocol implemented in Solaris OS, whose parameters can be changed with user level commands. Two parameters changed here are 'maximum interval of TCP retransmission' and 'minimum interval of TCP retransmission', which we call 'MAX' and 'MIN' respectively. The default setting is MAX = 60000 [msec] and MIN = 200 [msec] in the current version of Solaris. The interval of

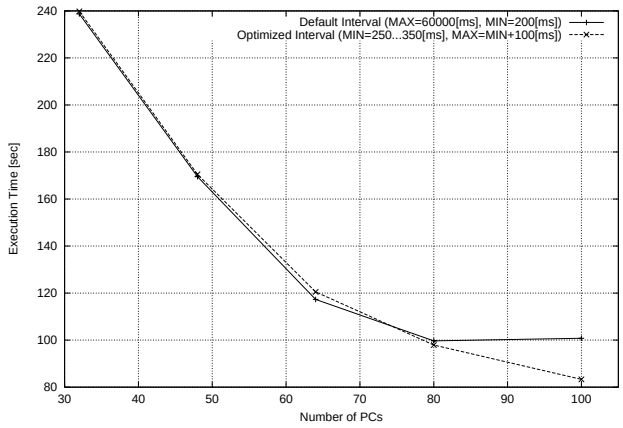

Fig. 2. Execution time of HPA program on PC cluster

TCP retransmission is dynamically changed in the protocol, within the limits between MAX and MIN.

As a result of experiments, we have found that the default value of MAX is not suitable for the cluster, which might cause the unnecessary long retransmission interval. MAX should be set to be smaller than the default value, such as MAX = MIN + 100[msec]. Moreover, MIN is better to be set as random value, which can prevent the collision of the cells at ATM switch.

4.2 Total performance of HPA program using proposed method

HPA program is executed using the proposed parameter setting of TCP on the PC cluster pilot system. The execution time of HPA program is shown in Figure 2. In this Figure, one line indicates the case using default TCP retransmission parameters, i.e. MAX = 60000[msec] and MIN = 200[msec], and the other line indicates the case using random parameters (MIN = 250 ... 350[ms], MAX = MIN + 100[ms]).

Reasonably good speedup is achieved up to 100 PCs using proposed optimized parameters. Since the application itself is not changed, the difference only comes from TCP retransmission, occurred along with barrier synchronization and all-to-all data broadcasting.

5 Dynamic remote memory acquisition

5.1 Dynamic remote memory acquisition and its experiments

As shown in section 3, the number of candidate itemsets in pass 2 is very much larger than in other passes in association rule mining. The number of itemsets is

strongly dependent on user-specified conditions, such as the minimum support value, and it is difficult to predict how large the number will be before execution. Therefore, it may happen that the number of candidate itemsets increases dramatically in this step so that the memory requirement becomes extremely large. When the required memory is larger than the real memory size, part of the contents of memory must be swapped out. However, because the size of each data item is rather small and all the data is accessed almost at random, swapping out to a storage device is expected to degrade the total performance severely.

We have executed several experiments in which available memory in remote nodes is used as a swap area when huge memory is dynamically required. In the experiments, a limit value for memory usage of candidate itemsets is set at each node. When the amount of memory used exceeds this value during the execution of the HPA program(in Pass 2), part of the contents is swapped out to available memory in remote nodes, that is, application execution nodes acquire remote memory dynamically. Although such available remote nodes could be found dynamically in a real system, we selected them statically in these experiments. On the other hand, when an application execution node trys to access an item that had been swapped out, a pagefault occurs.

The basic behavior of this approach has something in common with distributed shared memory systems[10], memory management system in distributed operating systems[11], or cache mechanism in client-server database systems[12]. For example, if data structures inside applications are considered in distributed shared memory, almost the same effect can be expected. That is to say, it is possible to program almost the same mechanism using some types of distributed shared memory systems. Thus, our mechanism might be regarded as equivalent to a case of distributed shared memory optimized for a particular application.

We have executed experiments of the proposed mechanism on the PC cluster. The parameters used in the experiment are as follows. The number of transactions is 1,000,000, the number of different items is 5,000, and the minimum support is 0.1%. The number of application execution nodes is 8 in this evaluation. The number of memory available nodes is varied from 1 to 16. With these conditions, the total number of candidate itemsets in pass 2 is 4,871,881. Since each candidate itemset occupied 24 bytes in total(structure area + data area), approximately 14-15Mbytes of memory were filled with these candidate itemsets at each node.

5.2 Remote update method

When memory usage is limited, the execution time is much longer than when there is no memory limit. This is because the number of swapouts is extremely large. In Table 2 the numbers of pagefaults on each application execution node are shown. Because most of the memory contents are accessed repeatedly, a kind of thrashing seems to happen in these cases. In order to prevent this phenomenon, a method for restricting swapping operations is proposed.

When usage of memory reaches the limit value at a node, it acquires remote memory and swaps out part of its memory contents. The contents will be

Table 2. The numbers of pagefaults on each application node

Usage limit	node 1	node 2	node 3	node 4	node 5	node 6	node 7	node 8
12[MB]	1606258	2925254	1306521	2361756	1671840	1723410	2166277	2545003
13[MB]	885798	1896226	593000	1374688	932374	896150	1326941	1375398
14[MB]	254094	1003757		512984	286945	191102	601657	407628
15[MB]		268039						

swapped in again if this data is accessed later. Instead of swapping, it is sometimes better to send update information to the remote memory when a pagefault occurs. That is to say, once some contents are swapped out to memory in a distant node, they are fixed there and accessed only through a remote memory access interface provided by library functions. This remote update method has been applied only to the itemsets counting phase, for simplicity.

The access interface function has been developed to realize the remote update operations. The execution time using this method is shown in Figure 3. This figure shows the execution time of pass 2 of the HPA program, when the number of memory available nodes is 16. The execution times for dynamic remote memory acquisition, according to this method and the previous simple swapping case, are compared in the Figure. The execution time using hard disks as a swapping device is also shown, for comparison. Seagate Barracuda 7,200[rpm] SCSI hard disks have been used for this purpose. Other conditions are the same as the case of dynamic remote memory acquisition.

The execution time using hard disks as swapping devices is very long, especially when the memory usage limit is small, because each access time to a hard disk is much longer than that for remote memory through the network. The execution time of dynamic remote memory acquisition with simple swapping is better than for swapping out to hard disks. It increases, however, when the memory usage limit is small, since the number of pagefaults becomes extremely large in such a case.

Compared to these results, the execution time of dynamic remote memory acquisition with remote update operations is quite short, even when the memory usage limit is small. It seems to be effective to provide a simple remote access interface for the itemsets counting phase, because the number of swapping operations during this phase is very large. These results indicate that, performance of the proposed remote memory acquisition with remote update operations is considerably better than other methods.

References

1. C. Huang and P. K. McKinley: "Communication Issues in Parallel Computing Across ATM Networks", *IEEE Parallel and Distributed Technology*, Vol.2, No.4, pp.73-86, 1994.

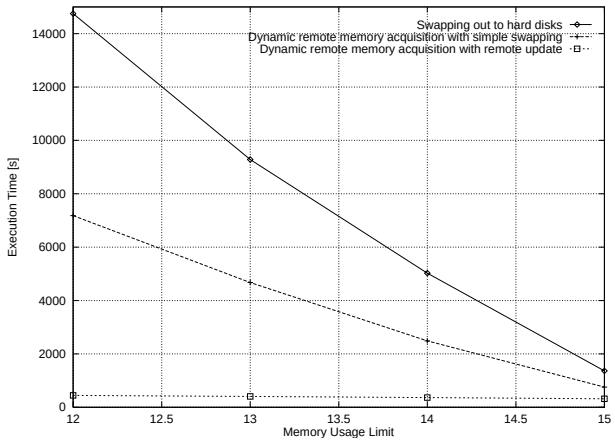

Fig. 3. Comparison of proposed methods

2. R. Carter and J. Laroco: "Commodity Clusters: Performance Comparison Between PC's and Workstations", *Proceedings of the Fifth IEEE International Symposium on High Performance Distributed Computing*, pp.292-304, August 1996.
3. D. E. Culler et al.: "Parallel Computing on the Berkeley NOW", *Proceedings of the 1997 Joint Symposium on Parallel Processing(JSPP '97)*, pp.237-247, May 1997.
4. T. Tamura, M. Oguchi, and M. Kitsuregawa: "Parallel Database Processing on a 100 Node PC Cluster: Cases for Decision Support Query Processing and Data Mining", *Proceedings of SuperComputing '97*, November 1997.
5. U. M. Fayyad et al.: "Advances in Knowledge Discovery and Data Mining", *The MIT Press*, 1996.
6. V. Ganti, J. Gehrke, and R. Ramakrishnan: "Mining Very Large Databases", *IEEE Computer*, Vol.32, No.8, pp.38-45, August 1999.
7. R. Agrawal, T. Imielinski, and A. Swami: "Mining Association Rules between Sets of Items in Large Databases", *Proceedings of the ACM International Conference on Management of Data*, pp.207-216, May 1993.
8. T. Shintani and M. Kitsuregawa: "Hash Based Parallel Algorithms for Mining Association Rules", *Proceedings of the Fourth IEEE International Conference on Parallel and Distributed Information Systems*, pp.19-30, December 1996.
9. M. J. Zaki: "Parallel and Distributed Association Mining: A Survey", *IEEE Concurrency*, Vol.7, No.4, pp.14-25, 1999.
10. C. Amza et al.: "TreadMarks: Shared Memory Computing on Networks of Workstations", *IEEE Computer*, Vol.29, No.2, pp.18-28, February 1996.
11. M. J. Feeley et al.: "Implementing Global Memory Management in a Workstation Cluster", *Proceedings of the ACM Symposium on Operating Systems Principles*, pp.201-212, December 1995.
12. S. Dar et al.: "Semantic Data Caching and Replacement", *Proceedings of 22nd VLDB Conference*, September 1996.