Implementation and Evaluation of Parallel Data Mining on PC Cluster and Optimization of its Execution Environments

Masato Oguchi, Masaru Kitsuregawa

Abstract— **Personal Computer/Workstation clusters have been studied intensively in the field of parallel and distributed computing. In the viewpoint of applications, data intensive applications such as data mining and ad-hoc query processing in databases are considered very important for high performance computing, as well as conventional scientific calculations. We have built and evaluated PC cluster pilot systems, especially SAN-connected PC cluster, and implemented parallel data mining on them. Several optimization, including dynamic data allocation, is discussed for the execution of this application.**

Keywords— **PC cluster, Data Mining, Storage Area Network, Optimization, Dynamic data allocation.**

I. Introduction

Recently personal computer/workstation (PC/WS) clusters have become a hot research topic in the field of parallel and distributed computing. They are considered to play an important role as large scale parallel computers in the next generation, for good scalability and cost performance ratio. The reason is as follows:

Components of today's high performance parallel computers are evolving from proprietary parts, e.g. CPUs, disks, and memories, into commodity parts. This is because technologies for such commodity parts have matured enough to be used for high-end computer systems. While an interconnection network has not yet been commoditized thus far, Some common-purpose networks, e.g. Fast/Gigabit Ethernet and ATM-LAN are the strong candidates as a de facto standard of high speed communication networks. With the progress of high performance networks, future parallel computer systems will undoubtedly employ commodity networks as well. Examining these technological trends, Fast Ethernet and/or ATM-connected PC clusters are considered promising platform for future high performance parallel computers.

In the viewpoint of application, we believe that data intensive applications, such as data mining and ad hoc query processing in databases, are extremely important for massively parallel processors in the near future. We previously developed a large scale ATM-connected PC cluster consists of 100 Pentium Pro PCs, and implemented several database applications, including parallel data mining, to evaluate their performance and the feasibility of such applications using PC clusters[1][2].

M. Oguchi is with the Research and Development Initiative, Chuo University, Tokyo, Japan. E-mail: oguchi@computer.org .

M. Oguchi and M. Kitsuregawa is with the Institute of Industrial Science, University of Tokyo, Tokyo, Japan.

Different from the conventional scientific calculations, association rule mining, one of the best known problems in data mining, has a peculiar usage of main memory. It allocates a lot of small data on main memory, and the number of those areas multiplies to be enormous during the execution. Thus, the requirement of memory space changes dynamically and becomes extremely large. Contents of memory must be swapped out if the requirement exceeds the real memory size. However, because the size of each data element is rather small and all the elements are accessed almost at random, swapping out to a secondary storage system is likely to cause severe performance degradation. We are investigating the feasibility of using available memory in remote nodes as a swap area, when application execution nodes need to swap out their memory contents. We report our experimental results in this paper, in which nodes executing an application acquire extra memory dynamically from several remote nodes in the ATM-connected PC cluster. Moreover, a method using distant node's memory with remote update operations, which is expected to prevent a thrashing problem, is proposed and evaluated.

LAN-connected PC clusters are employed as a system of large scale server sites and/or high performance parallel computers. In both cases, huge volume of data may be transferred frequently from one node's disk to another, for the execution of parallel computing, load distribution, maintenance of the system, and so on. A LAN cluster is a shared-nothing system, that is to say, all nodes of the cluster are connected only with a LAN and no disk is shared among them. Therefore, LANs may become almost always busy with data management of disks. The bandwidth of LANs in the cluster should not be flooded with these kinds of data transfer, because LANs should be used for other traffic, such as client-server request communication and parallel/distributed computing among several nodes.

In order to reduce LAN traffic and raise availability of nodes on the cluster, Storage Area Networks(SANs), e.g. Fibre Channel, has come to be adopted[3]. SANs link storage devices directly to all nodes of the cluster, therefore, SANs can prevent bandwidth congestion of LANs. In the case of SAN clusters, different from LAN clusters, each node does not have to communicate with each other through a LAN for reading data from other nodes' disks, because a pool of storage is shared among all nodes and can be accessed directly through a SAN without burden to the other nodes nor LANs.

In this paper, we have also built another PC cluster

which has a SAN-connection as well as a LAN-connection, and examined its performance features. Basic characteristics of data transfer on the cluster are evaluated. Performance of parallel data mining application on the SAN cluster is examined. In the case of SAN cluster, each node can access all shared disks directly. However, if a lot of nodes access the same shared disk simultaneously, performance of application must degrade due to I/O-bottleneck. A dynamic data declustering method, in which data is declustered to several other disks through a SAN during the execution of application, is proposed and evaluated.

The rest of paper is organized as follows: In Section II, related works on PC/WS clusters are introduced. Data mining application and its parallelization are related in Section III. Our PC cluster pilot systems are shown and implementation of parallelized association rule mining on them are mentioned in Section IV. In Section V, the method of dynamic remote memory utilization for parallel data mining is explained. Performance results of the evaluation of proposed mechanisms are shown and analyzed in Section VI. In Section VII, a dynamic data declustering method, which is expected to prevent I/O-bottleneck situation in use of shared disks, is proposed and evaluated. Final remarks are made in Section VIII.

II. Related works

Initially, the processing nodes and/or networks were built from customized designs, since it was difficult to achieve good performance using only off-the-shelf products[4][5]. Such systems were interesting as research prototypes, but most failed to be accepted as a common platform. However, because of advances in workstation and network technologies, reasonably high performance WS clusters can be built using off-the-shelf workstations and high speed LANs[6][7]. Several researches had been made on PC clusters[8][9], in which some scientific calculation benchmarks were executed on the cluster. Because performance of PCs and networks used in those projects was not good enough, absolute performance of such clusters was not attractive compared with high-end massively parallel processors.

Workstations were overwhelmingly superior to personal computers until recently, in terms of performance as well as sophisticated software environments. However, recent PC technology has dramatically increased its CPU, main memory, and cache memory performance. While RISC processors used in today's WSs provide better floating point performance than microprocessors used in PCs, some applications, such as database processing, primarily require good integer performance. Since integer performance of latest PCs is better than that of WSs (e.g. SPECint95 of 800MHz PentiumIII is 38.3, while SPECint95 of 450MHz UltraSPARC-II is 19.7), PCs have better cost performance ratios than WSs for database operations. High-speed bus architectures, such as the PCI bus, have also improved I/O performance of PCs. Moreover, sophisticated UNIX-based operating systems have been implemented on personal computers, which should greatly assist the realization of PC clusters. Because the size of the PC market is much larger than the WS market, further increase in the cost performance ratio is expected for PC clusters. As the performance of PCs has increased dramatically, variety of research projects on PC clusters have been reported until now[10][11][12][13][14].

III. Data mining application and its parallelization

A. Mining of association rules

Data mining has attracted a lot of attention from both research and commercial community, for finding interesting trends hidden in large transaction logs. Data mining is a method for the efficient discovery of useful information, such as rules and previously unknown patterns existing among data items in large databases, thus allowing for more effective utilization of existing data.

Large transaction processing system logs have been accumulated because of the progress of bar-code technology. Such data was just archived and not used efficiently until recently. The advance of microprocessor and secondary storage technologies allows us to analyze vast amount of transaction log data to extract interesting customer behaviors. For very large mining operations, however, parallel processing is required to supply the necessary computational power.

One of the best known problems in data mining is mining of association rules from a database, so called "basket analysis"[15][16][17]. Basket type transactions typically consist of a transaction identification 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 the Apriori algorithm proposed by R. Agrawal of IBM Almaden Research[18][19][20].

Apriori first generates so-called candidate itemsets (groups consisting of one or more items), then scans the transaction database to determine whether the candidates have the user-specified minimum support. In the first pass (pass 1), support for each item is counted by scanning the transaction database, and all items that achieve the minimum support are picked out. These items are called large 1-itemsets. In the second pass (pass 2), 2-itemsets (pairs of two items) are generated using the large 1-itemsets. These 2-itemsets are called the candidate 2-itemsets. Support for the candidate 2-itemsets is then counted by scanning the transaction database. The large 2-itemsets that achieve the minimum support are determined. The algorithm goes on to find the large 3-itemsets, the large 4-itemsets, and so on. This iterative procedure terminates when a large itemset or a candidate itemset becomes empty. Association rules that satisfy user-specified minimum confidence can be derived from these large itemsets.

B. Parallelization of association rule mining

In order to improve the quality of the rule, we have to analyze very large amounts of transaction data, which re-

TABLE I
EACH NODE OF THE ATM-CONNECTED PC CLUSTER

CPU	Intel 200MHz Pentium Pro
Chipset	Intel 440FX
Main memory	64Mbytes
IDE hard disk	WesternDigital Caviar32500 2.5Gbytes
SCSI hard disk	Seagate Barracuda 4.3Gbytes
OS	Solaris 2.5.1 for x86
ATM NIC	Interphase 5515 PCI Adapter

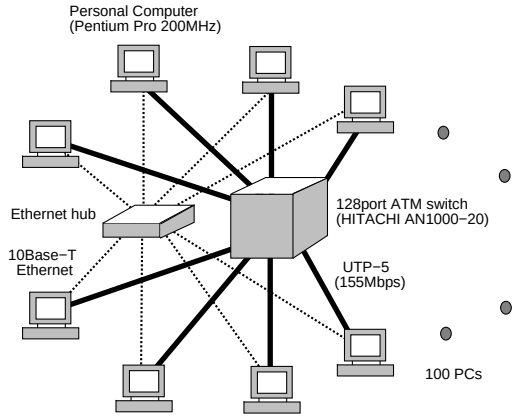

Fig. 1. An overview of ATM-connected PC cluster pilot system

quires considerable computation time. We have previously studied several parallel algorithms for mining association rules[21], based on Apriori. One of these algorithms, called Hash Partitioned Apriori (HPA), is implemented and evaluated on the PC cluster. 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. The steps of the algorithm are as follows.

1. Generate candidate k-itemsets:
   All processors have all the large $(k-1)$-itemsets in memory when pass k starts. Each processor generates candidate k-itemsets using large $(k-1)$-itemsets, applies a hash function, and determines a destination processor ID. If the ID is the processor's own, the itemset is inserted into the hash table, otherwise it is discarded.
2. Scan the transaction database and count the support value:
   Each processor reads the transaction database from its local disk. It generates k-itemsets from those transactions and applies the same hash function used in phase 1. The processor then determines the destination processor ID and sends the k-itemsets to it.
   When a processor receives these itemsets, it searches the hash table for a match, and increments the match count.
3. Determine large k-itemsets:
   Each processor checks all the itemsets it has and determines large itemsets locally, then broadcasts them to the other processors. When this phase is finished at all processors, large itemsets are determined globally. The algorithm terminates if no large itemset is obtained.

IV. OUR PC CLUSTER PILOT SYSTEM

A. An overview of our ATM-connected PC cluster

In our first PC cluster pilot system, 100 nodes of 200MHz Pentium Pro PCs has been connected with an ATM switch Each node consists of the components shown in Table I.

HITACHI's AN1000-20, which has 128 port 155Mbps UTP-5, is used as an ATM switch. Since this switch has 128 port, all nodes can be connected directly with each other, forming a star topology rather than a cascade configuration. All nodes of the cluster are connected by a 155Mbps ATM LAN as well as by an Ethernet. An RFC-1483 PVC driver, which supports LLC/SNAP encapsulation for IP over ATM[22][23], is used. Only UBR traffic class is supported in this driver. TCP/IP is used as a communication protocol. An overview of the ATM-connected PC cluster is shown in Figure 1.

B. Implementation of parallelized association rule mining

The HPA program described in Section III was implemented on the PC cluster pilot system. Each node of the cluster has a transaction data file on its own hard disk. Transaction data was produced using a data generation program developed by Agrawal, designating some parameters, such as the number of the transaction, the number of different items, and so on[20]. The produced data was divided by the number of nodes, and copied to each node's hard disk. WesternDigital Caviar32500 IDE disks are used for this purpose.

At each node, two processes are created and executed. One process makes candidate itemsets from previous large itemsets, and sends them to the other process, which puts the data into a hash table. In the data counting phase, one process generates itemsets by scanning the transaction data file, and sends them to the other processes on the nodes selected by the hash function. The target processes check and increment their hash table values accordingly.

Solaris Transport Layer Interface (TLI) system calls are used for the inter-process communication. All processes are connected with each other by TLI transport endpoints, thus forming a mesh topology. /dev/tcp is used as a transport layer protocol. It is a two-way connection based byte stream. On the ATM level, Permanent Virtual Channel (PVC) switching is used, since data is transferred continuously between all processes.

During the execution of HPA, itemsets are kept in memory as linked structures that are classified by a hash function. That is to say, all itemsets having the same hash value are assigned to the same hash line on the same node, and their structures are connected with each other to form

TABLE II

The number of candidate and large itemsets at each step

C	Number of candidate itemsets
L	Number of large itemsets

pass	C	L
pass 1		1023
pass 2	522753	32
pass 3	19	19
pass 4	7	7
pass 5	1	0

TABLE III

Each node of the SAN-connected PC cluster

CPU	Intel 800MHz Pentium III
Main memory	128Mbytes
IDE hard disk	Quantum Fireball 20Gbytes
SCSI hard disk	Seagate Cheetah 18.2Gbytes
OS	Solaris 7 for x86
Fast Ethernet NIC	3Com 3C905B-TX
Fibre Channel NIC	Emulex LP8000 Host Bus Adapter

a list.

As an example, a result of HPA is shown in Table II. In this execution, the number of transactions is 10,000,000, the number of different items is 5,000, and the minimum support is 0.7%. When the PC cluster using 100 PCs is employed for this problem, reasonably good performance improvement is achieved[2]. Several characteristics such as CPU usage and network performance of the cluster during the execution of HPA are analyzed and discussed in [24].

It is known that the number of candidate itemsets in pass 2 is very much larger than in other passes, as can be seen in the table. This often happens in association rule mining.

C. SAN-connected PC cluster pilot system

Recently, we have also built a SAN-connected PC cluster pilot system. 32 nodes of 800MHz Pentium III PCs are connected with Fast Ethernet as well as Fibre Channel. Each node consists of the components shown in Table III.

All 32 PCs of the cluster and 32 SCSI hard disks are connected with a Fibre Channel. Seagate Cheetah 18.2Gbytes is used as SCSI hard disks, and Brocade SilkWorm 2800 is employed as a Fibre Channel switch. Switching ability of this device is 200MB/s per port. Hitachi Black Diamond 6800, which has 64Gbps switching ability, is used as a Fast Ethernet Switch. This switch has more than enough capacity to connect 32 nodes through Fast Ethernet with no blocking. In Figure 2, an overview of the SAN-connected PC cluster is shown.

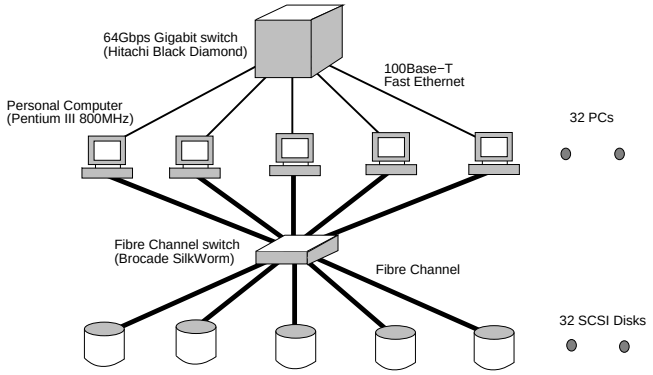

Fig. 2. An overview of SAN-connected PC cluster pilot system

V. Dynamic remote memory utilization during the execution of data mining

A. Swapping operation on the PC cluster

The number of candidate itemsets in pass 2 is very much larger than other passes in association rule mining, as mentioned in Section IV. The number of itemsets is strongly dependent on user-specified conditions, such as minimum support value, and it is difficult to predict before execution 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 requirement of memory becomes extremely large. When the required memory is larger than the real memory size, part of the memory contents must be swapped out. However, because the size of each data is rather small and all the data is accessed almost at random, swapping out to a storage device is expected to degrade performance severely in this case. In large scale clusters, a large fraction of the memory of total nodes is not in active use on average[25].

B. Dynamic remote memory acquisition

Remote nodes, whose memories are available for application execution nodes, are found dynamically during the execution. We call them "memory available nodes". The mechanism to decide the availability of remote nodes is shown in Figure 3. On memory available nodes, a process is running to monitor the amount of available memory periodically. "netstat -k" command provided by Solaris operating system is used to get memory information from the kernel statistics structure. [1] Each time the process gets the information, the process broadcasts it to all application execution nodes.

On application execution nodes, a client process is running and waiting for the information sent from the memory monitoring processes running on memory available nodes. The client process has a memory area which can be shared with application processes, and the received information about the amount of memory at each node is written on the shared memory. The application processes can read

[1]The "-k" option to netstat command is not documented on the manual pages. See [26] for more information.

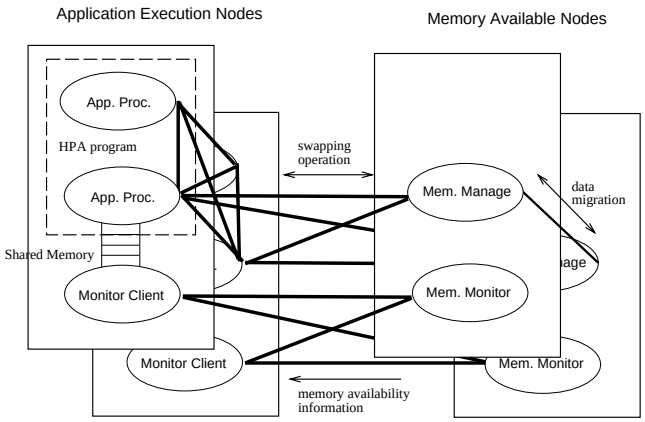

Fig. 3. Dynamic decision mechanism for remote memory availability

this information at anytime, to decide the policy of swapping operations. For example, when a memory available node does not have enough memory space, the shortage of memory is detected by application processes, so that another node is chosen as a swapping destination afterward. On the other hand, if some other processes begin their execution on a memory available node which already accepted swapping operations, the swapped out data must be migrated to other memory available nodes to make space on its memory for the new processes.

As a method of experiments, a limit value for memory usage of candidate itemsets is set at each node. When the amount of memory usage exceeds this value during the execution of HPA program, part of contents is swapped out to available remote nodes' memory. That is to say, the application execution node acquires memory area dynamically from one of memory available nodes when it is needed. When the number of candidate itemsets becomes extremely large in pass 2 and the amount of memory usage exceeds a specified value, the node sends some of its memory contents to destination memory available nodes. The unit of swapping operation is a hash line, which is a listed structures. The hash line swapped out is selected using a LRU algorithm. At the memory available node, the received contents are allocated and written in its main memory. Each memory available node may receive swapped out data from several application execution nodes.

On the other hand, a pagefault occurs when an application execution node accesses an item that had been swapped out. It sends a request to a memory available node which holds the data, and the memory available node sends back the requested hash line. After the application execution node receives the contents, they are allocated and written on main memory again, then the execution of application resumes. Replacements of data are decided by LRU manner.

C. Remote memory update option

Because most of memory contents are accessed repeatedly, the number of pagefaults is considered to become very high when the memory usage limit is small. A kind of thrashing may happen in such a case. In order to prevent this phenomenon, we investigate a method to restrict swapping operations.

When usage of memory reaches to the limit value at an application execution node, it acquires remote memory and swaps out part of its memory contents. The contents will be 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 remote node, they are fixed at there and accessed only through a remote memory access interface provided by library functions. We apply this policy to the itemsets counting phase, in which the memory access operation is simple – to compare itemsets with the contents of the hash table and update the table.

VI. Evaluation of dynamic remote memory utilization on PC cluster

A. Implementation of the proposed mechanism

The parameters used in the experiment are as follows: The number of transactions is 1,000,000, the number of different items is 5000, and the minimum support is 0.1%. The size of the transaction data is about 80Mbytes in total. The message block size is set to be 4Kbytes, and the disk I/O block size is 64Kbytes.

The number of application execution node is eight in this evaluation. The number of hash line for candidate itemsets is 800,000 in total, hence about 100,000 hash lines are assigned to each node during the execution. The unit of swapping operation is a hash line, which could be contained in one message block in this experiment.

With the above conditions, the number of candidate itemsets in pass 2 was 4,871,881 in total. These candidate 2-itemsets are assigned to each node using a hash function. Since each candidate itemset occupies 24bytes in total(structure area + data area), approximately 14-15Mbytes of memory are filled with these candidate itemsets at each node.

B. Dynamic remote memory acquisition

First, a method using available remote nodes' memory with simple swapping is examined. The maximum number of nodes used as memory available nodes is changed from 1 to 16. In this experiment, all memory available nodes are assumed to have enough memory space to accept requests of swapping operations. In such a case, all the memory available nodes are used for swapping operations throughout the execution of the program and therefore the number of memory available nodes is constant during the experiment. The execution time of pass 2 in HPA program, when the number of memory available nodes changes from 1 to 16, is shown in Figure 4. In this figure, the result of 5 different cases are shown. The upper 4 lines are the cases of memory usage for candidate itemsets being limited as 12Mbytes, 13Mbytes, 14Mbytes, and 15Mbytes, respectively. The lowest line is the case with no memory usage

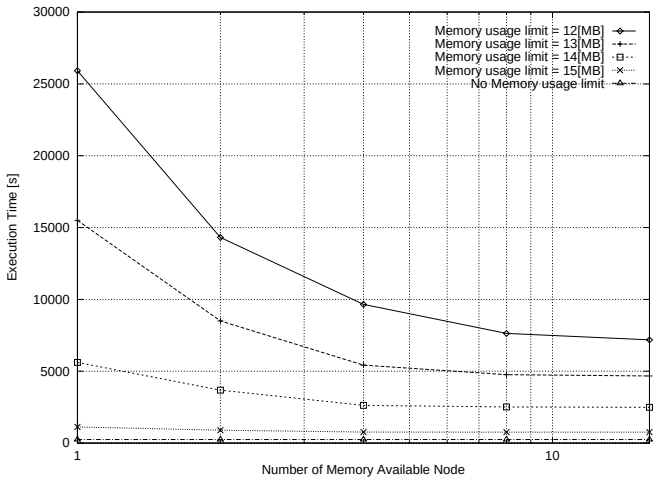

Fig. 4. Execution time of HPA program (pass 2)

TABLE IV
THE EXECUTION TIME FOR EACH PAGEFAULT (THE NUMBER OF MEMORY AVAILABLE NODES IS 16)

$Exec$	execution time of pass 2 in HPA [sec]
$Diff$	difference in execution time between this and the no memory limit case [sec]
Max	maximum number of pagefaults [times]
PF	execution time of each pagefault [msec]

Usage limit	$Exec$	$Diff$	Max	PF
12MB	7183.1	6936.1	2925243	2.37
13MB	4674.0	4427.0	1896226	2.33
14MB	2489.7	2242.7	1003757	2.22
15MB	757.3	510.3	268093	1.90

limit, in which application execution nodes are assumed to have enough memory for candidate itemsets so that no swapping occurs. Other mechanisms and conditions are the same with memory limited cases. Memory monitor mechanism is running in this case also, for comparison.

When the number of memory available nodes is small, the execution time is quite long especially when the memory usage limit size is smaller. Apparently memory available node(s) become bottleneck in these cases. This bottleneck is resolved when the number of memory available nodes is 8 - 16 in this experiment.

The execution time becomes longer as the memory usage limit size becomes small, since the number of swap out increases in such cases. When memory usage is limited, the execution time is quite longer than that of the no memory limit case. This is because the number of swap out is extremely large.

We can calculate the execution time of each pagefault as follows. We will focus on the case when the number of memory available nodes is 16, in which memory available nodes are not considered to be bottleneck. In the case of memory usage limit being 13Mbytes, for example, the execution time of the program is 4674.0sec, and the difference of the execution time between this and the no memory limit case is 4427.0sec. The total execution time is decided by the busiest node that does the most swapping operations. In this case, the maximum number of pagefaults in the busiest node was 1,896,226. Thus, the execution time of each pagefault can be obtained by dividing the difference in execution times by the maximum number of pagefaults, 2.33msec in this example. The other cases are also calculated in Table IV.

We can compare the pagefault execution time with the access time of hard disks. According to a state-of-art specification of SCSI hard disks, Seagate Barracuda 7,200rpm disks for example, the average seek time for read is about 8.8msec and the average rotation waiting time is about 4.2msec. In the case of latest fast hard disks such as HITACHI DK3E1T 12,000rpm disks, the average seek time for read is about 5msec and the average rotation waiting time is about 2.5msec. Therefore, it takes at least 13.0msec in average to read data from 7,200rpm hard disks, and 7.5msec even with the fastest 12,000rpm hard disks.

C. Using remote memory update option

The execution time using this method is shown in Figure 5. This figure shows the execution time of pass 2 of HPA program, when the number of memory available nodes is 16. The execution times of dynamic remote memory acquisition using remote update operations and using simple swapping are compared in the figure. The execution time using hard disks as a swapping device is also shown, for comparison. Seagate Barracuda 7,200rpm SCSI hard disk is used for this purpose. In this case, memory contents are swapped out to hard disks when the memory usage of candidate itemsets exceeds the limit value. Other conditions and implementations are the same with 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 of 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.

The execution time of dynamic remote memory acquisition with remote update operations is quite short compared to these results, 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 extremely large. According to these results, performance of the proposed remote memory utilization with remote update operations is considerably better than other methods.

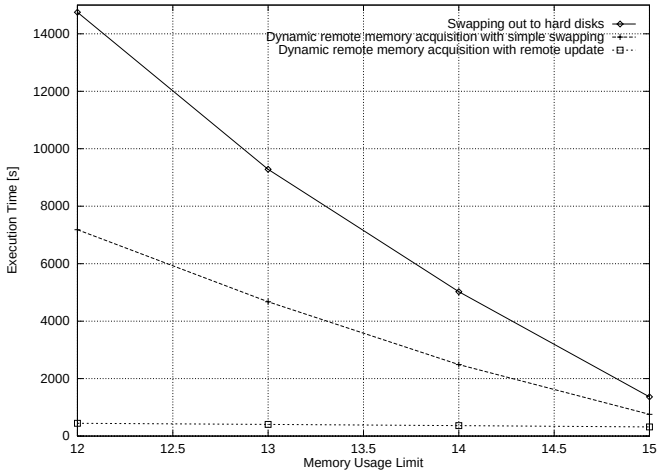

Fig. 5. Comparison of proposed methods

VII. Dynamic data declustering on SAN-connected PC cluster

A. Dynamic data declustering

Different from the case of LAN-connected cluster, each node can access all shared disks connected with SAN in the case of SAN-connected cluster. Applications on each node do not have to care where data is stored. However, when a lot of nodes access to shared disks simultaneously, performance of application must degrade due to I/O-bottleneck.

If stored data is accessed repeatedly during the execution of application, the probability of access conflict on the shared disk may increase. In such a case, it is better to decluster the data from one disk to many during/after first access. Each node copies portion of the data which may be accessed again afterward, from one shared disk to another which can be used exclusively. The copied portion of the data, instead of the original one, is used after the declustering is completed. We call this method dynamic data declustering in the rest of this paper. In some applications, it is difficult to decluster the data before execution of program. Using this method, it is possible to decide which node needs which portion of the data dynamically.

B. Execution of HPA program on SAN-connected PC cluster

The HPA program explained in Section III is implemented on our SAN-connected PC cluster pilot system. Solaris socket library is used for the inter-process communication. As a type of socket connection, SOCK_STREAM is used, which is two-way connection based byte stream.

In this experiment, the number of transaction is 10,000,000, the number of different items is 5,000, and the minimum support is 0.6%. The size of the transaction data is about 800Mbytes in total. The message block size is 8Kbytes, and the disk I/O block size is 64Kbytes in this experiment.

The produced transaction data is stored at one of SCSI hard disks, which is shared by all PCs through Fibre Channel. During execution of the application, this data is accessed by all processes concurrently. The contents of the data are divided by the number of nodes almost equally, and each node of the cluster reads its own portion during the execution. The number of nodes used in this application is eight. The result of HPA is shown in Table V.

TABLE V
The number of itemsets and the execution time at each pass

C	Number of candidate itemsets
L	Number of large itemsets
T	Execution time of each pass [sec]

pass	C	L	T
pass 1	–	1219	65.4
pass 2	742371	126	367.8
pass 3	92	52	63.6
pass 4	27	26	63.4
pass 5	8	8	63.1
pass 6	1	0	63.6

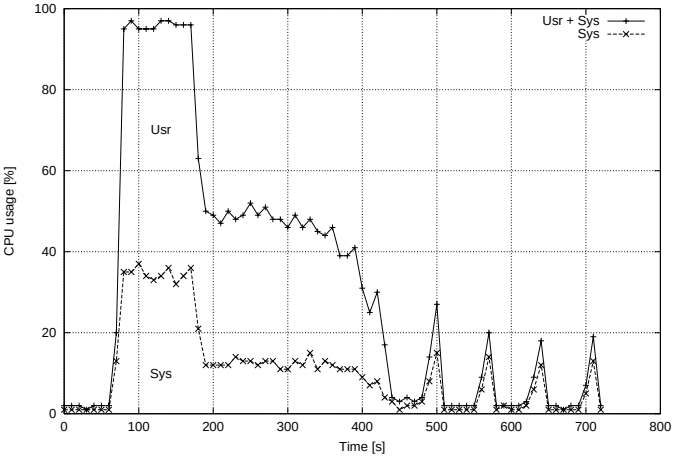

Fig. 6. CPU usage (using 1 Disk)

The details of system behavior are investigated when HPA program is executed on the SAN-connected PC cluster. CPU usage of system during the execution of HPA program is shown in Figure 6. This figures shows performance is bounded by CPU in pass 2, provided that network operation accounts a considerable part of CPU usage. In other passes, on the other hand, performance is bounded by I/O operation rather than CPU.

C. An evaluation of dynamic data declustering method

As mentioned in previous section, We have found pass 2 of HPA using one shared disk is a CPU-bound condition, but the CPU load is not high in other passes. In those passes, the accessed shared disks become bottleneck when the number of disks is small. Dynamic data declustering method proposed in Section VII-A can be applied in this situation.

Because each node is able to access all disks in the stor-

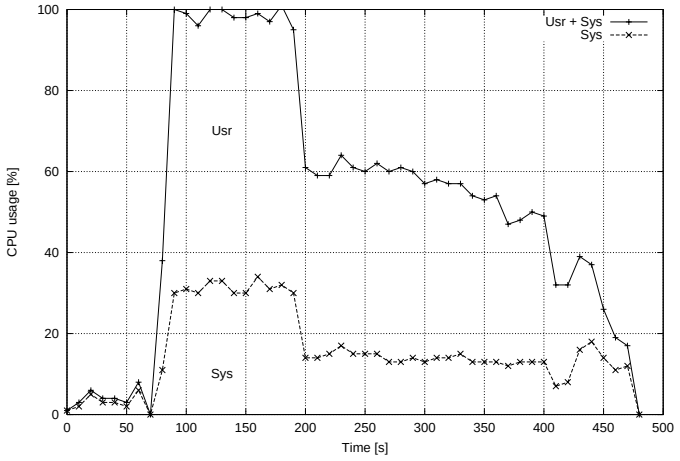

Fig. 7. CPU usage (Dynamic declustering from 1 to 8 disks)

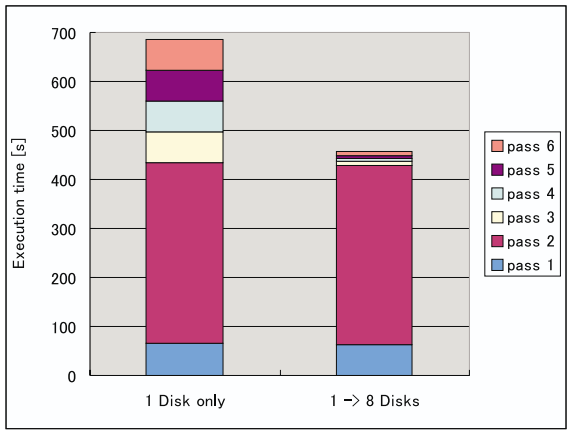

Fig. 8. Execution time of HPA program (Dynamic declustering from 1 to 8 disks)

age pool at any time, it is possible to copy its own portion of the transaction data to another disk, which can be used exclusively. In this method, portion of the data is copied to their own disks when the data is read at first in pass 1, then the copied data, instead of the original one, is used afterward. In Figure 8 − 10, the execution times of the proposed dynamic data declustering method are shown. The number of disks originally store the data is one − four, and the data is declustered to each node's own disk (eight disks in total) in pass 1 dynamically. The proposed method is compared with the original one in which data is read only from the same shared disk(s) repeatedly. CPU usage of the system, when data is declustered from one disk to eight disks, are also shown in Figure 7

The execution times in pass 1 and pass 2 are almost equal in both methods. In pass 1, this is because data must be read from a shared disk in both methods. After pass 1, the data is declustered to other disks which can be accessed exclusively. However, because pass 2 is not a I/O-bound condition, the execution times are almost the same in both methods. In pass 3 − pass 6, on the other hand, the execution time in the dynamic data declustering method is shorter than that of original method. This is because I/O-bottleneck is resolved by dynamic data declustering.

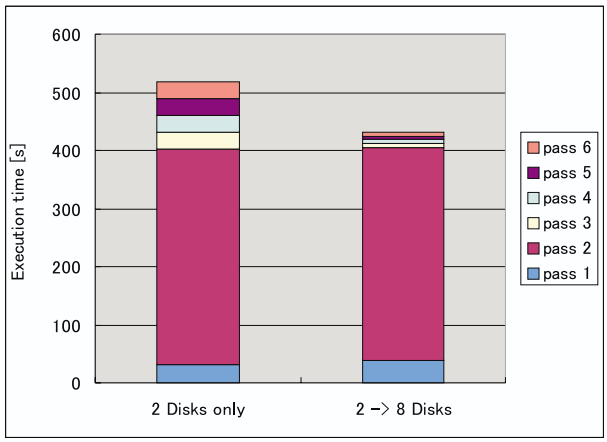

Fig. 9. Execution time of HPA program (Dynamic declustering from 2 to 8 disks)

VIII. Conclusions

In this paper, PC cluster pilot systems are constructed and evaluated. First, we show and discuss experimental results in which application execution nodes acquire extra memory dynamically from available remote nodes in an ATM-connected PC cluster. The experimental results show this method is considerably better than using hard disks as a swapping device. A method of updating remote memory in order to prevent thrashing was proposed and examined. This method achieves the best performance.

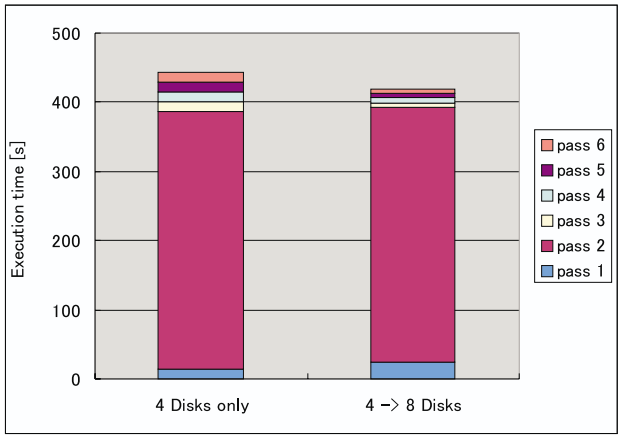

Fig. 10. Execution time of HPA program (Dynamic declustering from 4 to 8 disks)

SAN-connected PC clusters are suitable for a large scale server site because data transfer between disks does not have to depend on a LAN, thus the bandwidth of a network as well as the load of CPUs can be saved. We have also implemented and evaluated a data mining application on our SAN-connected PC cluster pilot system. In this application, transaction data is scanned repeatedly in iterative passes. A dynamic data declustering method, in which data is declustered from shared disks to other disks during the execution, is proposed and evaluated. As a result of the experiment implemented on the SAN cluster, the execution time of each pass becomes shorter, after the declustering are completed.

Acknowledgments

This project is partly supported by the Japan Society for the Promotion of Science (JSPS) RFTF Program and New Energy and Industrial Technology Development Organization (NEDO). We would like to thank Tokyo Electron Ltd. for technical help with Fibre Channel-related issues. Hitachi, Ltd. gave us extensive technical help with Gigabit switch.

References

[1] 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 SC97: High Performance Networking and Computing (SuperComputing '97)*, November 1997.

[2] M. Oguchi, T. Shintani, T. Tamura, and M. Kitsuregawa: "Optimizing Protocol Parameters to Large Scale PC Cluster and Evaluation of its Effectiveness with Parallel Data Mining", *Proceedings of the Seventh IEEE International Symposium on High Performance Distributed Computing*, pp.34-41, July 1998.

[3] B. Phillips: "Have Storage Area Networks Come of Age?", *IEEE Computer*, Vol.31, No.7, pp.10-12, July 1998.

[4] R. S. Nikhil, G. M. Papadopoulos, and Arvind: "*T: A Multithreaded Massively Parallel Architecture", *Nineteenth International Symposium on Computer Architecture*, pp.156-167, May 1992.

[5] M. Blumrich, K. Li, R. Alpert, C. Dubnicki, E. Felten, and J. Sandberg: "Virtual Memory Mapped Network Interface for the SHRIMP Multicomputer", *Proceedings of the Twenty-First International Symposium on Computer Architecture*, pp.142-153, April 1994.

[6] 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, Winter 1994.

[7] D. E. Culler, A. A. Dusseau, R. A. Dusseau, B. Chun, S. Lumetta, A. Mainwaring, R. Martin, C. Yoshikawa, and F. Wong: "Parallel Computing on the Berkeley NOW", *Proceedings of the 1997 Joint Symposium on Parallel Processing(JSPP '97)*, pp.237-247, May 1997.

[8] T. Sterling, D. Saverese, D. J. Becker, B. Fryxell, and K. Olson: "Communication Overhead for Space Science Applications on the Beowulf Parallel Workstation", *Proceedings of the Fourth IEEE International Symposium on High Performance Distributed Computing*, pp.23-30, August 1995.

[9] 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.

[10] A. Barak and O. La'adan: "Performance of the MOSIX Parallel System for a Cluster of PC's", *Proceedings of the HPCN Europe 1997*, pp.624-635, April 1997.

[11] H. Tezuka, A. Hori, Y. Ishikawa, and M. Sato: "PM: An Operating System Coordinated High Performance Communication Library", *Proceedings of the HPCN Europe 1997*, pp.708-717, April 1997.

[12] M. Oguchi, T. Shintani, T. Tamura, and Masaru Kitsuregawa: "Characteristics of a Parallel Data Mining Application Implemented on an ATM Connected PC Cluster", *Proceedings of the HPCN Europe 1997*, pp.303-317, April 1997.

[13] Y. Ishikawa, A. Hori, H. Tezuka, S. Sumimoto, T. Takahashi, F. O'Carroll, and H. Harada: "RWC PC Cluster II and SCore Cluster System Software − High Performance Linux Cluster", *Proceedings of the Fifth Annual Linux Expo*, pp.55-62, 1999.

[14] M. Banikazemi, V. Moorthy, L. Herger, D. K. Panda, and B. Abali: "Efficient Virtual Interface Architecture (VIA) Support for the IBM SP Switch-Connected NT Clusters", *Proceedings of the International Parallel and Distributed Processing Symposium*, pp.33-42, May 2000.

[15] U. M. Fayyad, G. P. Shapiro, P. Smyth, and R. Uthurusamy: "Advances in Knowledge Discovery and Data Mining", *The MIT Press*, 1996.

[16] V. Ganti, J. Gehrke, and R. Ramakrishnan: "Mining Very Large Databases", *IEEE Computer*, Vol.32, No.8, pp.38-45, August 1999.

[17] M. J. Zaki: "Parallel and Distributed Association Mining: A Survey", *IEEE Concurrency*, Vol.7, No.4, pp.14-25, 1999.

[18] R. Agrawal, T. Imielinski, and A. Swami: "Mining Association Rules between Sets of Items in Large Databases", *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data*, pp.207-216, May 1993.

[19] R. Agrawal, T. Imielinski, and A. Swami: "Database Mining: A Performance Perspective", *IEEE Transactions on Knowledge and Data Engineering*, Vol.5, No.6, pp.914-925, December 1993.

[20] R. Agrawal and R. Srikant: "Fast Algorithms for Mining Association Rules", *Proceedings of the Twentieth International Conference on Very Large Data Bases*, pp.487-499, September 1994.

[21] 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.

[22] J. Heinanen: "Multiprotocol Encapsulation over ATM Adaptation Layer 5", *RFC1483*, July 1993.

[23] M. Laubach: "Classical IP and ARP over ATM", *RFC1577*, January 1994.

[24] M. Oguchi, T. Tamura, T. Shintani, and M. Kitsuregawa: "Implementation of Parallel Data Mining on an ATM Connected PC Cluster and Performance Analysis of TCP Retransmission Mechanisms", *The Transactions of the Institute of Electronics, Information and Communication Engineers*, Vol.J81-B-I, No.8, pp.461-472, August 1998.

[25] A. Acharya and S. Setia: "Availability and Utility of Idle Memory in Workstation Clusters", *Proceedings of the 1999 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems*, pp.207-216, May 1993.

[26] A. Cockcroft: "How Much RAM is Enough?", Sun World Online, http://www.sunworld.com/sunworldonline/swol-05-1996/swol-05-perf.html, May 1996.