In this paper, an ATM connected PC cluster consists of 100 PCs is reported, and characteristics of a transport layer protocol for the PC cluster are evaluated. Point-to-point communication performance is measured and discussed, when a TCP window size parameter is changed. Retransmission caused by cell loss at the ATM switch is analyzed, and parameters of retransmission mechanism suitable for parallel processing on the large scale PC cluster are clarified.

In the viewpoint of applications, data intensive applications such as data mining and ad-hoc query processing in databases are considered very important for massively parallel processors, in addition to the conventional scientific calculation. Thus investigating the feasibility of such applications on an ATM connected PC cluster is quite meaningful.

Parallel data mining is implemented and evaluated on the cluster. Default TCP protocol cannot provide good performance, since a lot of collisions happen during all-to-all multicasting executed on the large scale PC cluster. Using TCP parameters according to the proposed optimization, sufficient performance improvement is achieved for parallel data mining on 100 PCs.
2. Background of this research work

2.1. Studies of PC/WS clusters

Several discussions investigating PC/WS clusters can be found in the literature. 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[1][2]. Such systems are interesting as a research prototypes, but most of them failed to be accepted as a common platform. However, because of advances in workstation and network technologies, we can build reasonably high performance WS clusters using off-the-shelf workstations and high speed LANs[3].

Until recently, workstations were overwhelmingly superior to personal computers, in terms of performance as well as sophisticated software environments. Recent PC technology however, has dramatically increased its CPU, main memory, and cache memory performance. While RISC processors used in todays WSs provide much higher floating point performance than microprocessors used in PCs, some applications such as database processing primarily require good integer performance. Since todays PCs and WSs have almost comparable integer performance, PCs have better cost performance ratio than do WSs for database operations. High speed bus architecture such as the PCI bus has also improved I/O performance of PCs. Since the size of PC market is much larger than the WS market, further increase in the cost performance ratio is expected for PC clusters.

Several projects on PC clusters have been reported until now[4][5], 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. Preferably good cost/performance has been achieved however, in these PC clusters[5].

2.2. Features of our project

Our studies on PC cluster have several features, different from other research works, as follows:

First, we have constructed a large scale PC cluster. While other reported PC clusters have several or several tens nodes at most, we realized a cluster consists of 100 Pentium Pro PCs. As far as the authors know, no research has been reported on large scale PC clusters, in which over 100 nodes are connected with a high speed network. The only reported cluster consists of over 100 nodes is UCB’s NOW project in which 105 SPARC WSs are connected with Myrinet[6]. Since the amount of data processed and transferred simultaneously is quite different between the cases of large scale and small scale clusters, it is difficult to discuss the behavior of a large scale system unless we construct over 100 nodes clusters actually.

Second feature, ATM is used as a communication network in our cluster. Since other high speed networks such as Fast Ethernet are also widely used, some cluster experiments employ those media. Moreover, a cluster-oriented network like Myrinet has come to be commercially available[6][7][8], which provides better network performance, although several restrictions exist, including a limited distance between nodes. Compared with those networks, ATM is extensively used from local area to widely distributed environments. This seamless structure and its quality control mechanisms are among the merits of ATM technology. Although it has been said ATM may not be suitable for pure data transmission purposes and/or does not fit with traditional computer communication protocols such as TCP/IP, recent dramatical improvements of computer and NIC technology are solving these problems.

Because ATM is developed as a general network rather than a dedicated network, we must investigate if any problem exists when it is used as a connection network of large scale PC clusters. Especially in this paper, we are focusing on a transport layer protocol on ATM networks, that is, how to make TCP/IP over ATM work well on the cluster. TCP is not only a very popular reliable protocol for computer communication, but also having quite general functions as a transport layer. Thus the results of our experiments must be valid even if other transport protocols are used, for investigating connection-oriented communication protocols on large scale clusters.

As a third feature of our project, we used data intensive applications for the evaluation of the PC cluster. Various research projects to develop PC/WS clusters have been reported until now. Most of them however, only measured basic characteristics of PCs and networks, and/or some small benchmark programs were examined. In NOW project also, the WS cluster is evaluated using scientific applications and simple sorting programs[6]. Data intensive
applications such as data mining and ad-hoc query processing in databases are considered very important applications for parallel processors, in addition to the conventional scientific calculations[9]. In this paper, we employ data mining as an example of data intensive applications.

3. PC cluster pilot system and its communication performance

3.1. Components of the cluster

In our pilot system, 100 PCs are connected with an ATM switch. 200MHz Pentium Pro PCs are used as a node of the cluster. Each node consists of components shown in Table 1.

Table 1. Each node of PC cluster

<table>
<thead>
<tr>
<th>Component</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>Intel 200MHz Pentium Pro</td>
</tr>
<tr>
<td>Chipset</td>
<td>Intel 440FX</td>
</tr>
<tr>
<td>Main memory</td>
<td>64Mbytes</td>
</tr>
<tr>
<td>Disk drive</td>
<td>2.5Gbytes IDE hard disk</td>
</tr>
<tr>
<td>OS</td>
<td>Solaris2.5.1 for x86</td>
</tr>
<tr>
<td>ATM NIC</td>
<td>Interphase 5515 PCI ATM Adapter</td>
</tr>
</tbody>
</table>

All nodes of the cluster are connected with a 155Mbps ATM LAN as well as an Ethernet. We use RFC-1483 PVC driver, which supports LLC/SNAP encapsulation for IP over ATM[10][11]. Only UBR traffic class is supported in this driver. TCP/IP over ATM is used as communication protocols.

HITACHI’s AN1000-20 is used as an ATM switch. Since this switch has 128 port 155Mbps UTP-5, all nodes can be connected directly with each other, composing a star topology rather than a cascade configuration. A photograph of the pilot system is shown in Figure 1, and an overview of the PC cluster is shown in Figure 2.

3.2. Performance of point-to-point communication

Effective throughput between nodes through the ATM network is evaluated on the PC cluster. As a tool for measurements, a version of ttcp program, with slight modification to support ATM and to work on Solaris x86 OS, is used. Throughput is severely influenced by flow control mechanisms implemented in a transport layer protocol. We use TCP protocol in Solaris OS, whose parameters can be changed with user-level command. In this experiment, point-to-point throughput is measured when a maximum TCP window size is changed on the cluster.

Basically, the window size is decided by a receiver according to a buffer size and data processing rate at the node, and notified to a sender. However, TCP also enable senders to control data flow by setting a window size, called
congestion window. While the window size is dynamically changed by TCP protocol during the communication, the maximum size of congestion window is modified explicitly in this evaluation. Two suites of experiments are performed: MSS(Maximum Segment Size) is set to be 8192 bytes and 1024 bytes, respectively. The block size is 8Kbyte, and the total amount of transmitted data is 64Mbytes, in both cases.

Point-to-point throughput is shown in Figure 3, and the numbers of Send and Ack packets transmitted during the execution are shown in Figure 4 and Figure 5.

![Figure 3. Point-to-point throughput](image1)

![Figure 4. Number of transmission packet (MSS=8192[bytes])](image2)

When a maximum window size is large, the throughput is considerably high, in the case of MSS=8192[bytes]. It almost reaches 120Mbps when the maximum window size is larger than about 50Kbytes. Since default parameters in current OS are set within this range, point-to-point throughput of the cluster is quite satisfactory even if the default parameter settings are not tuned up. Throughput becomes lower as a maximum window size becomes smaller.

When MSS is small, i.e. MSS=1024[bytes], throughput is only about 80Mbps at most, due to the segments processing overhead in the transport layer.

![Figure 5. Number of transmission packet (MSS=1024[bytes])](image3)

Average numbers of Send/Ack packets can be seen in Figure 4 and Figure 5. According to these figures, for example, one Ack packet is sent back on every three Send packets in average, when MSS=8192[bytes] and the maximum window size is larger than about 20–30Kbytes. If the maximum window size is smaller than this value, Ack packets are sent back more frequently, which degrade throughput of the communication dramatically as shown in Figure 3. It may be difficult to explain why the throughput is not a smooth curve when MSS=8192[bytes] in Figure 3, though the average ratio of Send/Ack numbers is constant when the maximum window size is larger than 20–30Kbytes. Since the maximum window size (congestion window) is a limit value at a sender side, an actual window size must be frequently changed during the communication depending on buffer status at a receiver side, even if the average Send/Ack ratio is constant. Thus the feature of the curve in this range must be decided by the condition of receivers, which is quite sensitive to packet receive timing.

According to these experiments, throughput becomes higher when MSS and the maximum window size is large. Although the window size cannot be set larger than 65535 bytes due to TCP header’s window size bits limitation, the default parameter settings seem to be appropriate as far as point-to-point throughput is concerned. Thus we use the default value for the maximum window size in the rest of this paper.

1 Window scale option is proposed and adopted in some latest BSD operating systems in which more than 16 bits window size can be used. The current version of Solaris does not support this option.
4. Optimization of transport layer protocol parameters for large scale PC cluster

4.1. Broadcasting on the cluster and TCP retransmission

In parallel and distributed applications, barrier synchronization and exchange of data are executed frequently. In such a case, all-to-all broadcasting takes place. Even if the amount of broadcast data is not large, a lot of collisions happen in a large scale ATM connected PC cluster, if timing of the broadcasting is the same at all nodes. This is a serious problem. When broadcasting is performed almost simultaneously at all nodes, and a network becomes heavily congested, cells are discarded at the ATM switch and TCP retransmission should happen as a result.

Several experiments are executed on 100 nodes of the PC cluster, in order to investigate retransmission characteristics. Two parameters used for tuning here are ‘maximum interval of TCP retransmission’ and ‘minimum interval of TCP retransmission’. We call them ‘MAX’ and ‘MIN’ respectively in the rest of the paper. The default settings are \( \text{MAX} = 60000\text{[msec]} \) and \( \text{MIN} = 200\text{[msec]} \) in the current version of Solaris. The interval of retransmission is dynamically changed according to the mechanism of TCP, within the limits of MAX and MIN.

4.2. Changing maximum interval of TCP retransmission

A simple all-to-all broadcasting program is executed on the cluster using 100 nodes. In this program, each node executes barrier synchronization at first, then sends 50Kbytes data to all the other nodes, and executes barrier synchronization again.

First, the broadcasting program is executed when MAX is changed while MIN is fixed at the default value (200[msec]). The execution time of the program is shown in Figure 6. In each case, the program is executed ten times respectively, and all results are indicated by different marks on the Figure.

As shown in the figure, the marks are dispersed but not completely at random, because the execution time in this evaluation is decided mainly by the TCP retransmission interval, which takes an exponential back-off value such as 6sec, 12sec, 24sec, and so on. The execution time tends to be smaller when MAX is short. Since the application itself is not changed, these differences must come from TCP retransmission waiting time only.

The most right points in the figure are the cases when \( \text{MAX} = 60000\text{[msec]} \), that is, the default value. Obviously the default value of MAX is not suitable for this cluster, due to the unnecessary long retransmission interval. Since general communication protocols assume to be used in a wide area distributed environment, the maximum interval of retransmission is set to be quite long. Such a long retransmission interval is meaningless for local clusters, thus we should prevent it from becoming longer.

![Figure 6. Execution time of the broadcasting program (MIN = 200[msec])](image)

4.3. Optimization of retransmission interval

Next, the broadcasting program is executed when MIN is changed. MAX is set to be smaller than the default value, such as \( \text{MAX} = \text{MIN} + 100\text{[msec]} \). The execution time is shown in Figure 7, and the amount of TCP retransmission during the execution is shown in Figure 8. The amount of TCP retransmission is represented per each node, which is the average value of all nodes. In this experiment also, the program is executed ten times respectively, and all results are indicated by different marks on the Figures.

The execution time of the program is reasonably short, and not so much changed when MIN varies. On the other
hand, the amount of retransmission is dispersed randomly as MIN is changed.

Figure 8. Amount of retransmission (MAX = MIN + 100 [msec])

Figure 9. Amount of retransmission (MAX = MIN + 100 [msec])

TCP mechanism dynamically changes the interval of retransmission, in order to set the most suitable value at each moment. Different from the case of communication among a small number of nodes, however, this method is not sufficient for a large scale PC cluster, because a great number of nodes may use the same value for the interval of retransmission, which causes a collision and heavy traffic congestion.

Thus we used randomly different value from node to node, as the interval of TCP retransmission. In this experiment, random value between X ... X + 100[msec] is used as MIN, and MAX = MIN + 100[msec] like the previous experiment. Note that MAX and MIN are randomly different from node to node, but they are not changed dynamically during the execution. The amount of retransmission during the execution is shown in Figure 9. The execution time of the program has become almost the same with Figure 7. According to these results, changing the interval of TCP retransmission dynamically may not be enough for an application which includes simultaneous multicasting, in the case of a large scale PC cluster. Using randomly distributed value among nodes provides better performance in such a case.

5. Data mining application and its parallelization

Data mining is the 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 most well known problems in data mining is mining of the association rules from a database, so called “basket analysis”[12]. 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 most well known algorithm for association rule mining is Apriori algorithm proposed by R. Agrawal of IBM Almaden Research[13].

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[14], 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), the 2-itemsets (length 2) are generated using the large 1-itemsets. These 2-itemsets are called the candidate 2-itemsets. Then supports for the candidate 2-itemsets are counted by scanning the transaction database, the large 2-itemsets which satisfy the minimum support are determined. This repeating 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 as in the hash join. HPA effectively utilizes the whole memory space of all the processors. Hence it works well for large scale data mining. Detailed explanation of HPA is shown in [15].
6. Execution of the parallel data mining application on PC cluster

6.1. Implementation of HPA program

HPA program has been implemented on the PC cluster pilot system. 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. As a type of socket connection, SOCK_STREAM is used, which is two-way connection based byte stream. In the ATM level, PVC (Permanent Virtual Channel) switching is used since data is transferred continuously among all processes.

Transaction data is produced using data generation program developed by Agrawal, designating some parameters such as the number of transaction, the 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 transactions is 10,000,000, the number of different items is 5000, and the 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 in this experiment.

6.2. Execution of HPA program and evaluating effectiveness of the proposed optimization

The execution time of HPA program is shown in Figure 10. A solid line indicates the case using default TCP retransmission parameters, i.e. MAX = 60000[msec] and MIN = 200[msec], and a dotted line indicates the case using optimized parameters proposed in Section 4 (MIN = 250 ... 350[ms] and MAX = MIN + 100[ms], as one of the most appropriate parameter sets). The Speedup ratio calculated from the execution time is shown in Figure 11.

Reasonably good speedup is achieved as the number of PCs increases. Using default parameters, however, speedup is saturated and even become worse when the number of PCs is 100. As we traced the number of data transmission during the execution of HPA program, TCP retransmission was observed in last several passes. Because little data is processed in these last passes, the program execution time itself must be quite short. At the end of each pass, barrier synchronization and exchange of data are needed among all nodes, that is, all-to-all broadcasting takes place. Thus broadcasting is performed almost simultaneously in all nodes at these passes, which causes network congestion and TCP retransmission when the number of PCs is large.

TCP retransmission was observed only when a large number of PCs are used for the execution. The threshold of the number of PCs for this retransmission is about 80–90 in our pilot system, as we examine the data transmission in the application. This threshold should be decided as follows: In the all-to-all broadcasting phase, each node sends and receives several hundred bytes of data, i.e. a few tens ATM cells, to and from all the other joined nodes in order to update the current status of the execution. Since the buffer size assigned for each port in our ATM switch is 2000 cells, these buffers must overflow and cells are discarded when the number of PCs joined for the application is larger than 80–90.
barrier synchronization and data broadcasting is frequently used in parallel and/or distributed applications. According to the result of this experiment, the proposed method to optimize retransmission parameters must be quite effective, especially in the case of numbers of nodes being quite large in PC clusters.

7. Conclusion

In this paper, optimization of a transport layer protocol parameters for the large scale PC cluster was discussed. Retransmission caused by cell loss at the ATM switch was analyzed, and parameters of retransmission mechanism suitable for parallel processing on the large scale PC cluster were clarified. This method was evaluated using data mining application. Default TCP protocol could not provide good performance, since a lot of collisions occur in all-to-all broadcasting executed on the large scale PC cluster. Using a TCP parameters according to the proposed optimization, sufficient performance improvement has been achieved with parallel data mining on 100 PCs.

Acknowledgment

This project is partly supported by NEDO (New Energy and Industrial Technology Development Organization) and JSPS (Japan Society for the Promotion of Science). HITACHI, Ltd. technically helped us extensively ATM related issues.

References


