Effects of Packet Pacing for MPI Programs in a Grid Environment

Ryousei Takano [1,2], Motohiko Matsuda [1], Tomohiro Kudoh [1]
Yuetsu Kodama [1], Fumihiro Okazaki [1], Yutaka Ishikawa [3,1]

[1]*Grid Technology Research Center*
National Institute of Advanced Industrial Science and Technology (AIST), Japan
{takano-ryousei, m-matsuda, t.kudoh, y-kodama, f-okazaki}@aist.go.jp
[2]*AXE, Inc.*
[3]*University of Tokyo*
ishikawa@is.s.u-tokyo.ac.jp

Abstract— **Improving the performance of TCP communication is the key to the successful deployment of MPI programs in a Grid environment in which multiple clusters are connected through high performance dedicated networks. To efficiently utilize the inter-cluster bandwidth, a traffic control mechanism is required so as not to allow the aggregate transmission bandwidth to exceed the inter-cluster bandwidth when multiple nodes communicate at one time. In this paper, we propose a traffic control method for MPI programs, in which an application or the MPI runtime controls the transmission rate based on the communication pattern by using certain MPI attributes. Packet pacing is used at each node preventing microscopic burst transmission to thus avoid congestion. We confirm the effectiveness of the proposed method by experiments using a 10 Gbps emulated WAN environment. We show most of the NAS Parallel benchmarks improve the performance, since the proposed method reduces packet losses due to traffic congestion on the inter-cluster network. The results have indicated that it is feasible to connect multiple clusters and run large-scale scientific applications over distances up to 1000 kilometers, if an appropriate network is available.**

I. Introduction

Several Grid-enabled MPI systems designed for multiple clusters connected by a wide-area network have been proposed. Users can migrate their application programs from a local system to a Grid environment, and process a very large data set that is too large to run on a single system. Since most of these systems use TCP for inter-cluster communication, improving the performance of TCP communication is the key to successful deployment of MPI programs in a Grid environment.

We have shown in previous work that the NAS Parallel benchmarks (NPB) , which are not modified to tolerate latency, obtain the practical performance if one-way latency is smaller than 10 msec, based on an evaluation using an emulated WAN environment [4]. A 10 msec latency roughly corresponds to 1000 kilometers in actual networks, which covers the major cities in Japan as follows:

Route	Distance	1/2 round trip time
Tokyo – Tsukuba	60 km	0.8 msec
Osaka – Tsukuba	500 km	4.7 msec
Fukuoka – Tsukuba	900 km	9.4 msec

We have developed GridMPI [1] based on knowledge obtained from this work. GridMPI almost conforms to the MPI-2.0 specification. GridMPI is implemented based on the YAMPI [2] MPI system, and supports the Interoperable MPI (IMPI) protocol [3] to use multiple clusters over TCP.

Recently, optical network technology for wide-area networks has progressed rapidly, and clusters can be connected by 10 to 40 Gbps of bandwidth. In this paper, we assume a dedicated and fixed-bandwidth network. This assumption can be considered a realistic scenario based on Grid resource provisioning work like G-lambda [9], which provides co-allocated computing and optical network resources with advance reservation through Web services interfaces.

In such an environment, the efficient use of the inter-cluster network is important to achieve high performance in executing message passing programs. We may assume an environment in which the inter-cluster bandwidth (for example, 10 Gbps) is higher than the bandwidth of the network interface (for example, 1 Gbps) of each cluster node, and the aggregate bandwidth of nodes in a cluster is higher than the inter-cluster bandwidth. If the inter-cluster bandwidth is utilized efficiently, multiple nodes should be able to communicate at one time.

However, if multiple nodes communicate using the inter-cluster link, congestion may occur. In that case, the TCP congestion control mechanism limits the amount of data transmission. Since the congestion window size grows at each acknowledgement packet reception during the congestion avoidance phase, the window size slowly grows in a high bandwidth-delay product network. This causes performance degradation. Therefore, a traffic control mechanism is required to avoid this congestion and to utilize the inter-cluster bandwidth.

The basic idea of the traffic control method proposed in this paper is simple. First, we calculate the maximum allowable transmission bandwidth of each node (MATB) based on the communication pattern, so as not to allow the aggregate transmission bandwidth of the nodes to exceed the inter-cluster link bandwidth. Then we use a packet pacing software program called *PSPacer* [7], which we have developed, to regulate the transmission bandwidth of each node. PSPacer can precisely regulate the bandwidth of the outgoing traffic at a sender node by adjusting the space between packets.

The MPI runtime notifies PSPacer of the MATB by using the MPI attribute mechanism. For collective communication operations, such as *all-to-all*, the MPI runtime calculates the MATB and sets certain MPI attributes at the start and the end of each collective operation. For other communication patterns, knowledge of the communication pattern of the application is required to calculate the MATB. Therefore, applications explicitly control the transmission rate by setting MPI attributes.

The rest of the paper is organized as follows. Section II describes how to calculate MATB based on the communication patterns, and also illustrates the communication patterns of the NAS parallel benchmarks. In Section III, we present an overview of PSPacer. The implementation of the proposed method on GridMPI is shown in Section IV. Section V presents the experimental results obtained using an emulated WAN environment with 10 Gbps of bandwidth and a delay that ranged from 0 to 10 msec to demonstrate the effectiveness of the proposed method. We discuss an optimization of the proposed method in Section VI. In Section VII, we briefly mention related work on Grid-enabled MPI systems focused on inter-cluster communication. Finally, Section VIII summarizes the paper.

II. Transmission rate control based on communication patterns

We define the maximum allowable transmission bandwidth of a node (MATB) as the inter-cluster link bandwidth divided by the number of possible sender nodes to the link.

The use of the inter-cluster link varies depending on the communication pattern of the application. If a collective communication operation is used in an application program, its communication pattern is well fixed and thus the MATB can be calculated without specific knowledge about the application program. For example, for an *all-to-all* collective communication implementation, all the nodes participate in the inter-cluster communication, and therefore the MATB can be calculated by dividing the inter-link bandwidth by the number of nodes used in the cluster. On the other hand, if a point-to-point communication such as *send/receive* is used, MATB can not be calculated automatically. The application program should explicitly specify the MATB by calculating it based on the communication pattern used. The communication pattern can be obtained from a profiling run of the application and the environment, but the approach is not discussed in this paper.

Now we discuss on the communication patterns of the NPB benchmarks, a typical high performance computing benchmark program. Here, we assume two clusters with the same number of nodes (N nodes for each cluster) are connected by an inter-cluster link with bandwidth B.

NPB includes eight benchmarks: BT (Block Tridiagonal ADI), CG (Conjugate Gradient), EP (Embarrassingly Parallel), FT (Fourier Transform), IS (Integer Sort), LU (LU Factorization), MG (Multi-Grid Method), and SP (Scaler Pentadiagonal ADI). The NPB benchmarks, except for IS and FT, exchange most of the data by using a combination of point-to-point operations.

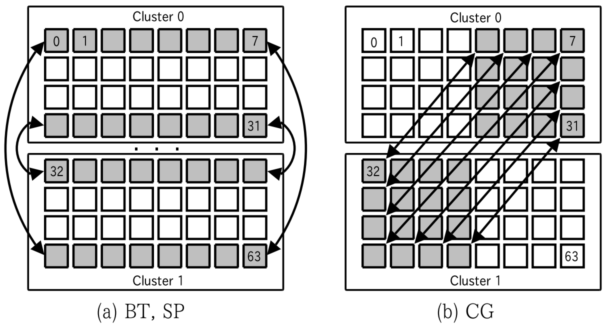


Fig. 1. Inter-cluster communication in the NPB benchmarks. The processor array is divided into 2 clusters. Only half nodes of each cluster (i.e., the grayed box) exchange data through inter-cluster communications.

TABLE I
Maximum allowable bandwidth (MATB) of each node for the NPB benchmarks. N is the number of nodes at each cluster; and B is the inter-cluster bandwidth.

Benchmark	MATB
BT, SP	$B/(2\sqrt{2N})$
CG	$B/(N/2)$
LU, MG	B/N
FT, IS (*all-to-all*)	B/N

The IS and the FT benchmarks use the `MPI_Alltoallv` and `MPI_Alltoall` collective communication operations, respectively. Therefore, the number of nodes which participate in the inter-link communication is N.

The communication pattern of the BT, SP, and CG benchmarks is shown in Figure 1, where N is 32 (i.e., the total number of nodes is 64). In the BT and SP benchmarks, two rows in each cluster participate in the inter-cluster communication. Therefore, the number of nodes participating in the inter-cluster communication can be calculated by $2\sqrt{2N}$. On the other hand, in the CG benchmark, half of the nodes in each cluster exchange data through the inter-cluster link. Therefore, $N/2$ nodes participate in the inter-cluster communication.

In the LU and the MG benchmarks, all the nodes transfer data through inter-cluster communications. Therefore, the number of nodes which participate in the inter-link communication is N.

The programmer can calculate the required MATB as the inter-cluster bandwidth (B) divided by the number of nodes which will transfer data simultaneously through inter-cluster communications. The MATB's are shown in Table I. For example, the MATB is 625 Mbps for the CG benchmark, given by the equation $B/(N/2)$, when $N = 32$ and $B = 10\ Gbps$. The calculation of MATB described above can be straightforward extended to more clusters.

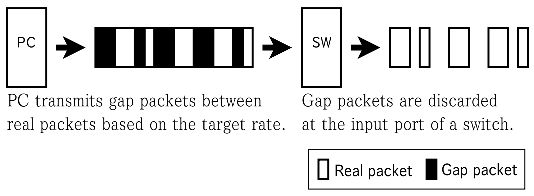

PC
SW
PC transmits gap packets between real packets based on the target rate.
Gap packets are discarded at the input port of a switch.
Real packet
Gap packet

Fig. 2. Precise pacing method using gap packets.

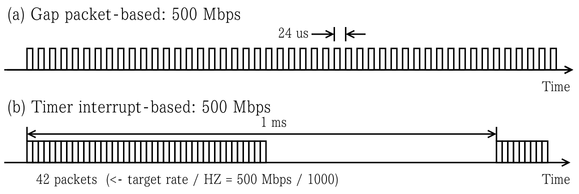

(a) Gap packet-based: 500 Mbps
24 us
Time
(b) Timer interrupt-based: 500 Mbps
1 ms
42 packets (<- target rate / HZ = 500 Mbps / 1000)
Time

Fig. 3. Implementation of a rate limitation method: (a) gap packet-based and (b) timer interrupt-based. The transmission rate is 500 Mbps, the timer frequency is 1000 HZ, and MTU is 1500 bytes.

III. PSPacer

We use a packet pacing software program called PSPacer [7] to regulate the transmission bandwidth of each node. Different from existing rate limitation methods such as the Token Bucket Filter (TBF) of Linux, which uses timer interruption, PSPacer achieves quite precise rate limitation results, and no microscopic burst transmission which exceeds the limit is generated.

PSPacer controls the packet transmission intervals by inserting additional packets, called *gap packets*, between adjacent packets. The transmission interval can be controlled accurately by adjusting the number and size of the gap packets. Figure 2 shows an overview of the method. For example, to limit the transmission rate to one half the link rate, a gap packet of the same size as the preceding real packet is inserted after every real packet.

PSPacer uses the 802.3x PAUSE frame as a gap packet. Therefore, the gap packets are discarded at the input ports of switches and routers, and only real packets go through the inter-cluster networks.

Figure 3 shows the outgoing packets from a sender node, when either a gap packet-based and a timer interrupt-based method are used to regulate the bandwidth to 500 Mbps. The former accurately adjusts intervals between packets to 24 usec. In contrast, the latter always produces a certain amount of burst transmission. The performance of TCP communication can degrade as the burstiness of traffic increases, because the number of packet losses at switches or routers increases as the burstiness increases [8].

PSPacer is implemented as a loadable kernel module on the Linux operating system. PSPacer is based on the iproute2 traffic control framework, and it is independent from both network protocols and network interface cards (NICs).

IV. Implementation

A. Overview

We have implemented the proposed traffic control method on GridMPI. As described in section II, for point-to-point communications, applications calculate the MATB based on communication patterns, as shown in Table I, and notify the GridMPI runtime using MPI attributes. For collective communications, the GridMPI runtime calculates the MATB itself. Then, the GridMPI runtime notifies PSPacer of the MATB, and PSPacer regulates the outgoing traffic.

B. Rate control attributes

MPI attributes provide a caching facility that allows an application to attach arbitrary pieces of information associated with a communicator. We use the MPI attribute mechanism to exchange information between the user applications and the MPI runtime. Applications can explicitly retrieve and pass transmission rate control parameters by using the attributes.

The proposed method provides two predefined attribute keys: `YAMPI_PSP_MAXRATE` and `YAMPI_PSP_MATB`. `YAMPI_PSP_MAXRATE` returns the shared bandwidth in Kbytes/sec, which typically means the available bandwidth between clusters (i.e., B). Currently, this value is given to the MPI runtime by an environment variable. `YAMPI_PSP_MATB` is the regulated transmission rate of the host in Kbytes/sec, which can be using "put" and "get" operations. The operation to put the attribute triggers that the attribute value is sent to PSPacer by the MPI runtime, which is different from the normal usage of attributes.

A typical program code segment is shown as below. `psp_set_limit` takes the number of nodes which transfer data simultaneously through inter-cluster communications as the parameter `n`. This function works as follows: (1) The inter-cluster bandwidth is obtained by `MPI_Attr_get` with `YAMPI_PSP_MAXRATE`. (2) The MATB is obtained by `MPI_Attr_get` with `YAMPI_PSP_MATB`. (3) The MATB is calculated. Then, finally, (4) MATB is set by `MPI_Attr_put` with `YAMPI_PSP_MATB`.

```
    int psp_set_limit(MPI_Comm *comm, int n)
    {
      int *rate, *matb, flag;

(1)   MPI_Attr_get(comm, YAMPI_PSP_MAXRATE,
                   (void *)&rate, &flag);
      if (flag) { /* if the attribute is given */
(2)     MPI_Attr_get(comm, YAMPI_PSP_MATB,
                     (void *)&matb, &flag);
(3)     *matb = *rate / n;
(4)     MPI_Attr_put(comm, YAMPI_PSP_MATB,
                     (void *)matb);
      }
    }
```

C. Rate-managed collective operations

For collective communication, we provide rate-managed collective operations, and no modification of application code is needed. When applications call a collective operation, the

MPI runtime calculates the MATB and sets attributes at the start and the end of the collective operation.

Collective operations are internally replaced by rate-managed operations using a runtime loading feature of a set of communication operations. Currently, only `MPI_Alltoall` and `MPI_Alltoallv` are supported. A rate-managed `MPI_Alltoall` is illustrated as below. Here, `_ImpiAlltoall` is the default internal function of `MPI_Alltoall`; and `nhosts_in_client` returns the number of nodes in the cluster to which the node belongs. The transmission rate is regulated during the communication process of `MPI_Alltoall`.

```
int _PspAlltoall(void *sbuf, ...)
{
  int ret;
  /* enter the rate regulation */
  psp_set_limit(comm, nhosts_in_client());
  ret = _ImpiAlltoall(sbuf, scnt, stype,
               rbuf, rcnt, rtype, comm);
  /* leave the rate regulation */
  psp_set_limit(comm, 1);
  return ret;
}
```

V. Evaluation

A. Experimental Setting

Figure 4 shows the experimental setting, and Table II shows the specifications of the node PC and the switches. Each cluster consisted of 32 nodes. Each node has both Gigabit Ethernet and Myrinet interfaces. Two clusters were connected through a 10 Gbps WAN emulator. Myrinet-2000 and MX 1.6 [12] were used for intra-cluster communications, and Gigabit Ethernet was used for inter-cluster communications. In the experiment, we used GtrcNET-10 [10][11] as the WAN emulator. GtrcNET-10 consists of a large-scale Field Programmable Gate Array (FPGA), three 10 Gbps Ethernet XENPAK ports, and three blocks of 1 GB DDR-SDRAM. The FPGA is a Xilinx XC2VP100, which includes three 10 Gbps Ethernet MAC and XAUI interfaces. GtrcNET-10 provides many functions, such as traffic monitoring in microsecond resolution, traffic shaping, and WAN emulation at 10 Gbps wire speed. GtrcNET-10 was used to add delay between clusters and to observe network traffic. GtrcNET-10 added a one-way delay, which varied from 0 msec to 10 msec. Note that the round trip delay is the value multiplied by two.

We used the NAS Parallel Benchmarks (NPB 3.2), and modified them to regulate the transmission rate using the proposed method. The problem sizes were classes B, C, and D [1].

Some TCP parameters were changed from the default value, as shown in Table III, because default parameters such as socket buffer sizes are not adequate for the experiment. tcp_no_metrics_save disables reuse of the parameters of the previous connection.

[1]The class D IS benchmark is not available. For the compilation of the class D FT benchmark, the code model was set to medium (i.e., "-mcmodel=medium"), because the data size is larger than 2 GB.

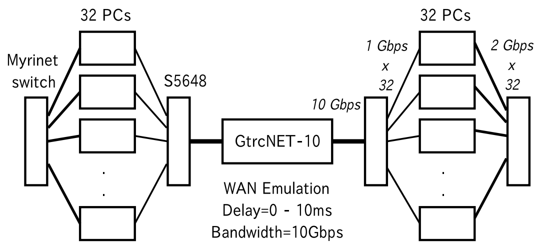


Fig. 4. Experimental Setting. Myrinet/MX (2 Gbps) is used for intra-cluster communications, and Gigabit Ethernet (1 Gbps) is used for inter-cluster communications.

TABLE II
PC Cluster Specifications.

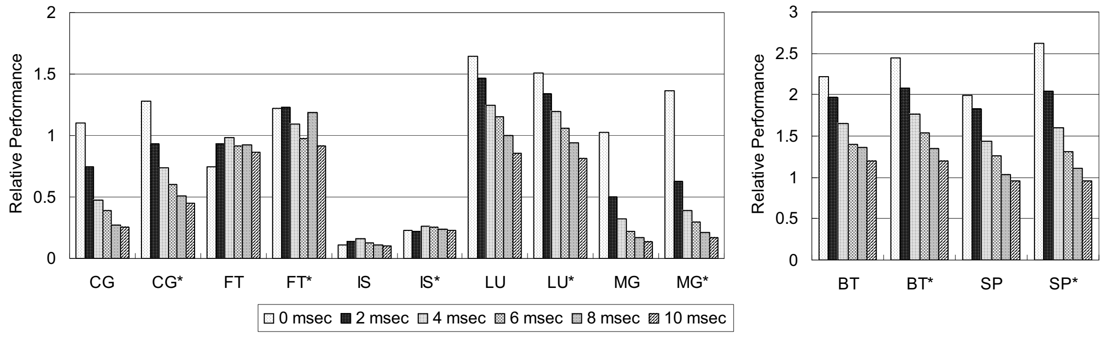
Node PC	
CPU	Opteron/2.0 GHz dual
Memory	6 GB DDR333
Ethernet	Broadcom BCM5704
Myrinet	Myricom M3F-PCIXD-2
OS	SuSE Enterprise Server 9 (Linux-2.6.17)
Compiler	Intel Compiler 9.1
Switch	
Huawei-3Com Quidway S5648 + optional 10 Gbps port	
Myricom M3-SW16-8F + M3-SPINE-8F	

IMPI [3] sends messages as chunks (IMPI packets). IMPI_C_DATALEN is the maximum packet size in bytes. IMPI uses a rendezvous protocol if the size of an MPI message is larger than this value. However, when delay is large, use of the rendezvous protocol will degrade performance. Therefore, in the experiment, we avoided use of the rendezvous protocol by setting IMPI_C_DATALEN to a very large value (21474836 = 0x7fffffff). IMPI also uses flow control of IMPI packets. IMPI_H_HIWATER specifies the maximum number of packets sent without receiving ACKs from the other end of the connection, the default value of which is 20. Although this value does not have much effect when IMPI_C_DATALEN is set to a very large value, here, it is set to 4000.

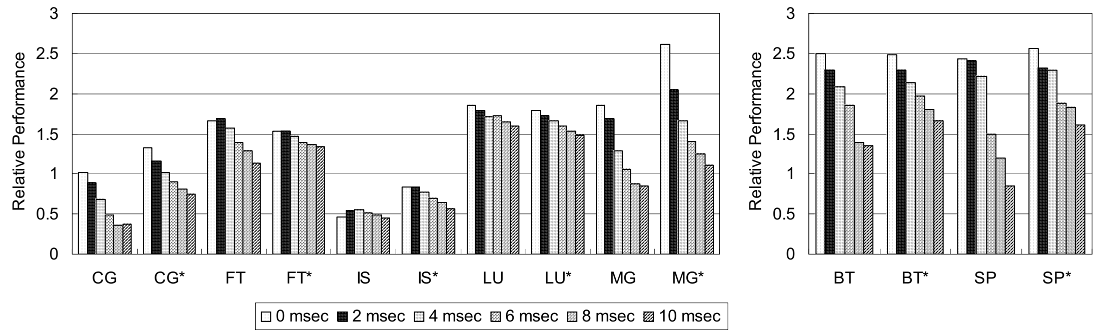
We ran each benchmark three times and took the highest performance value among the results.

TABLE III
Parameters.

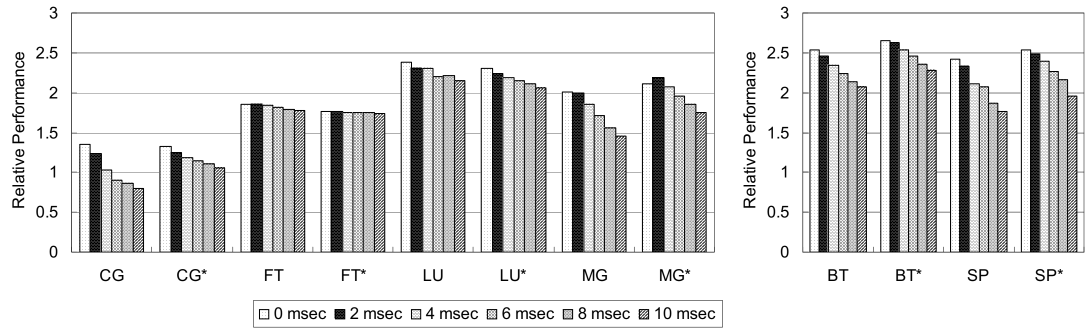
sysctl parameters	
net.core.rmem_max	3000000
net.core.wmem_max	3000000
net.ipv4.tcp_rmem	3000000 3000000 3000000
net.ipv4.tcp_wmem	3000000 3000000 3000000
net.ipv4.tcp_no_metrics_save	1
Environment variables for GridMPI	
IMPI_SOCBUF	3000000
IMPI_C_DATALEN	2147483647
IMPI_C_HIWATER	4000



(a) class B



(b) class C



(c) class D

Fig. 5. Relative performance normalized to the single 32-node or 25-node cluster cases. For each benchmark, the left set of bars (e.g., CG) shows the case without the proposed method, and the right set of bars (e.g., CG*) shows the case with the proposed method.

TABLE IV

PERFORMANCE OF NPB BENCHMARKS (MOP/S TOTAL). THE UPPER ONE SHOWS THE RESULTS OF THE SINGLE 32-NODE CLUSTER CASE; THE BOTTOM ONE SHOWS THOSE OF THE SINGLE 25-NODE CLUSTER CASE.

class	CG	EP	FT	IS	LU	MG
B	4052.79	286.09	9231.33	429.68	21845.65	16343.43
C	4463.16	286.33	8817.88	377.75	22362.18	12821.56
D	3154.22	286.10	8354.19	-	14247.11	14530.26

class	BT	SP
B	26632.09	12092.8
C	18855.64	7364.76
D	19607.83	7607.38

B. Results

Figure 5 shows the relative performance of the cases using both clusters (a total of 64 nodes) with various inter-cluster delays normalized to the single 32-node or 25-node cluster cases. For the BT and SP benchmarks, the total number of nodes should be a square of an integer. Therefore, they are normalized to the 25-node cluster case. For other benchmarks, they are normalized to the 32-node case. For each benchmark, the left set of bars (e.g., CG) shows the case without the proposed method, and the right set of bars (e.g., CG*) shows the case with the proposed method. Table IV shows the absolute performance in Mop/s with the single 32-node or 25-node cluster cases.

In most of the results, performance becomes worse linearly

TABLE V

RELATIONSHIP BETWEEN THE RELATIVE PERFORMANCE AND THE PROBLEM SIZE, WHERE THE DELAY IS 10 MSEC.

class	CG	EP	FT	IS	LU	MG	BT	SP
B	0.45	1.96	0.92	0.23	0.81	0.17	1.20	0.95
C	0.75	1.99	1.34	0.56	1.48	1.11	1.66	1.61
D	1.06	2.00	1.74	-	2.06	1.75	2.27	1.96

as the delay increases. A further remarkable point is that the influence of the delay decreases as the problem size increases. This is because both the cost of computation and the message size become relatively larger as the problem size increases.

Table V focuses on the relationship between the performance and the problem size, where the delay is 10 msec. All benchmarks of class D outperform the single 32-node cluster, even though the delay is 10 msec. The results have indicated that it is feasible to connect clusters and run large-scale scientific applications over distances up to 1000 kilometers, if an appropriate network is available.

A detailed examination of the obtained results is described below. Shown in Figure 5 (a) are the results of class B. The performance of IS and FT seems to indicate very little relationship with the delay. Especially, the performance of IS is significantly lower when two clusters are used, although the performance of the proposed method is almost two times larger than the standard GridMPI. This is because TCP on Ethernet is used for inter-cluster communication, and the burstiness of the traffic caused a lot of packet losses. The reason for these results is discussed in Section V-C. Figure 5 (b) shows the results of class C. For MG, super-linear speedup was observed. The effect of cache memory can be considered as the reason for this super-linear speedup. The number of L2 cache misses is 3.1 times lower when the number of processors is 64 compared with the case with 32 processors, since line conflict misses and capacity misses increase.

In Figure 6, the performance improvement by the proposed method is shown by comparing with the standard GridMPI. The performance improves as the problem size decreases, since the communication cost is relatively large when the problem size is small.

Focusing on the results in which the delay is 10 msec, in class B, CG, IS, and MG show improvement from 35 to 130 percent, FT shows a slight improvement, and BT and SP show no improvement in performance. In class C, BT, CG, FT, IS, MG, and SP show improvement from 20 to 100 percent in performance. In class D, BT, CG, MG, and SP show improvement from 10 to 30 percent, and FT shows a slight degradation in performance. In all classes, LU shows a slight degradation in performance.

When the problem size is large, the effectiveness of the proposed method increases as the delay increases, as shown in Figure 6 (c). The result indicates that the proposed method is effective for large bandwidth delay product networks.

However, when the problem size is small (i.e., in the cases of class B and class C), we observe significant performance improvement when the delay is small (0 to 2 msec). This phenomenon can be explained as follows. When the data size is small, the size of data transferred at one time is small. In such a situation, the TCP congestion control mechanism used to adjust the transmission rate does not work well. Therefore, many packet losses occur if the proposed method is not used.

C. Analysis of the effects of packet pacing

To analyze effects of packet pacing, we observed the detailed behavior of inter-cluster communications using the traffic monitoring feature of GtrcNET-10. Figures 7, 8, 9, and 10 show the bandwidth sampled in 1 msec resolution. Here, the problem size is class C, and the delay is 0 msec. The results for the CG and the IS benchmarks show the advantage of packet pacing. On the other hand, performance is slightly degraded by using PSPacer in the FT and the LU cases.

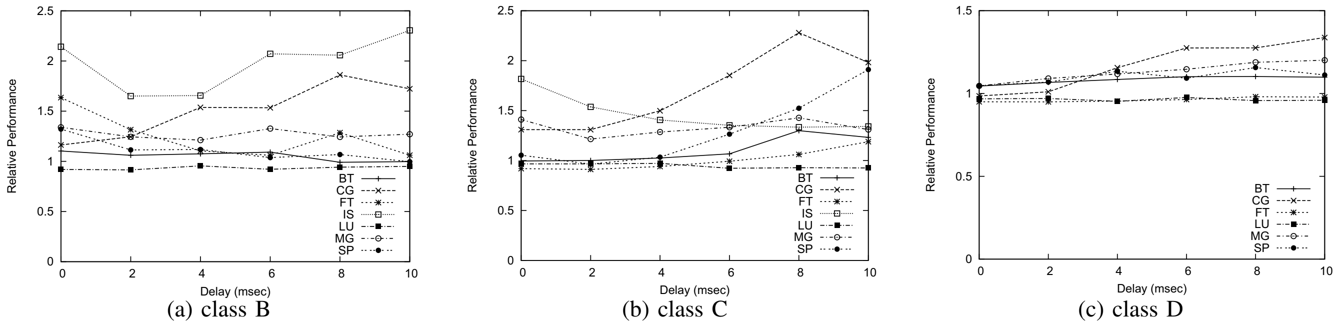
In the CG benchmark, there are some spike-shaped traffics (bursts at small intervals) as shown in Figure 7. Without pacing, traffic stops for a period of about 200 msec twice during the observation. These traffic stops are caused by Retransmit-Timeouts (RTO), which are the results of bursty packet losses due to microscopic bursts in some spikes. When pacing is used, there are no such microscopic bursts, and no RTO occurred.

In TCP congestion control algorithms, bursty packet losses are much more serious than a single packet loss. Sometimes these losses are not recovered by NewReno or Selective ACK (SACK) algorithms, and cause RTO.

The performance impact on a message passing programs is more serious. When the tail of the last message within an iteration is dropped, the receiver can not recognize packet losses because there are no successive packets within the iteration. Therefore, the sender can not retransmit the lost packet until it recognizes the loss by detecting RTO. While waiting for the RTO, the execution of the program stops, and it can not proceed to the next iteration. In the Linux TCP implementation, the RTO is approximately $RTT + 200\ msec$. Thus, the communication stops 200 msec or longer, and causes low bandwidth utilization.

In the IS benchmark, the bandwidth reaches nearly 10 Gbps, as shown in Figure 8. Without pacing, in the 1.4 msec period shown in Figure 8 (a), two iterations are executed. Bursty packet losses occur and result in RTOs, and the traffic in an iteration is divided into four major blocks. On the other hand, with pacing, four iterations are executed in the same period, as shown in Figure 8 (b). Here, no RTO occurs because the traffic is paced.

The FT benchmark shown in Figure 9 also fully utilizes the 10 Gbps inter-cluster link. Both (a) and (b) show traffic during two iterations. We observed packet losses in the both cases. The bandwidth with pacing changes more gently than the without-pacing case, while the number of packet losses is only one-tenth of without-pacing case. When pacing is used, the maximum transmission rate of each node is regulated, and it takes more time to transmit a given amount of data than without pacing. In the *all-to-all* communication, each node



(a) class B (b) class C (c) class D

Fig. 6. Performance improvement ratio obtained with the proposed method.

TABLE VI
OPTIMIZED RATE CONTROL WITH THE CLASS B IS BENCHMARK (MOP/S TOTAL).

Delay (msec)	Standard	B/N	$B/2N$
0	46.16	98.88	348.27
2	58.17	96.00	250.36
4	68.42	113.25	243.34
6	53.06	109.93	199.18
8	48.94	100.70	196.29
10	43.12	99.42	170.88

send data both to nodes inside its cluster and nodes at the other cluster. Therefore, not all the node send data using the inter-cluster link, and the inter-cluster link is not fully utilized in the with-pacing case. However, we cannot tell how many nodes send data using inter-cluster nodes, since the intra-cluster and the inter-cluster bandwidth are different, and the communication phases are not synchronized.

We observed packet losses in the IS and the FT benchmarks even with pacing. These losses occur not at the inter-cluster link, but at the switch of the receiver side cluster. If multiple nodes send data to a single destination node, the aggregated bandwidth can exceed the interface bandwidth of the receiver node. As a result, the buffer of the switch may overflow, and cause packet losses.

In such cases, smaller MATB will reduce congestion at the receiver switch. Table VI shows the results with the class B IS benchmark with smaller MATB. By changing the regulated rate from the original B/N to $B/2N$, we got 2 to 3.5 times the performance improvement.

The traffic of the LU benchmark is very low, as shown in Figure 10. Both (a) and (b) show traffic during five iterations. The inter-cluster link is not fully utilized in both cases, and no packet losses occur. Since there are no packet losses even in the without-pacing case, the performance becomes slightly worse when pacing is used, in which case the packet transmission intervals are stretched, and then the time period of the communication phase increases.

VI. DISCUSSION

In the current implementation, an application program sets MATB. However, if the MPI runtime maintains information about network topology, the application only have to notify communication pattern of the application to the MPI runtime. The MPI runtime may be able to calculate MATB based on the network topology information and the communication pattern.

Currently, the MATB is calculated assuming all the possible sender nodes transmit at as high a rate as possible. However, not all the nodes may send at the MATB. Therefore, by relaxing the regulation rate, performance may be improved. On the other hand, as discussed in the previous section, using smaller MATB sometimes yields better performance, since the congestion at the receiver switch buffer can be avoided. However, there is currently no established way to find the optimal regulation rate.

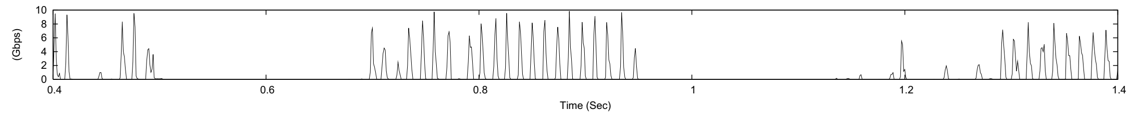
For collective communication other than *all-to-all*, only one node from each cluster participates in the inter-cluster communication in most existing MPI implementations for a Grid environment. Therefore, the bandwidth of the fast inter-cluster link can not be fully utilized. We have proposed efficient *bcast* and *allreduce* collective operation algorithms for fast wide-area inter-cluster networks, in which multiple nodes can participate in the inter-cluster communication [6]. That is, the number of nodes allowed to communicate at one time is algorithmically limited to avoid congestion. Inside a cluster, data are forwarded to the nodes which participate in the inter-cluster communication.

The communication pattern of collective operations depends on the algorithms. Our future work includes application of pacing to the proposed reduction algorithms.

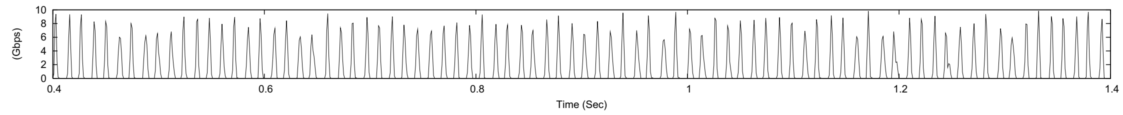
VII. RELATED WORK

Several Grid-enabled MPI systems, including PACX-MPI [13], MPICH-G2 [14], MPICH-Madeleine [16], MPICH-VMI [17], and MC-MPI [19] have been proposed in the past. However, these studies give a few consideration to the impact of the delay on the performance in large bandwidth delay product networks.

MC-MPI provides a locality-aware connection management that limits the number of inter-cluster connections by forward-

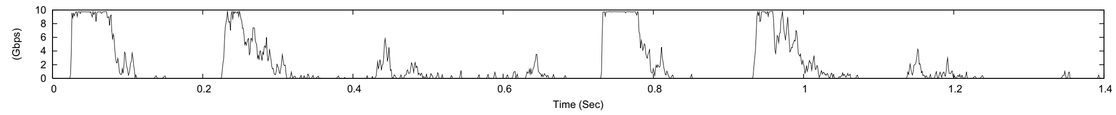


(a) without pacing

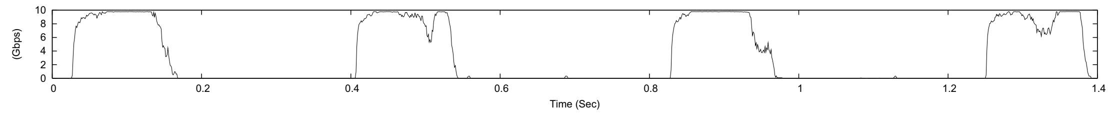


(b) with pacing

Fig. 7. Inter-cluster traffic of class C CG, 0 msec delay.

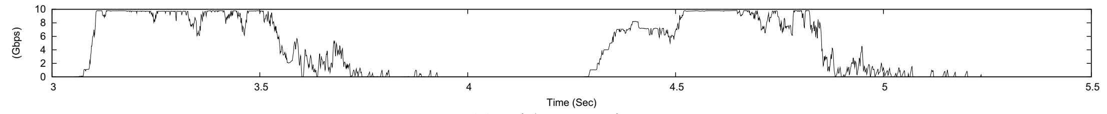


(a) without pacing

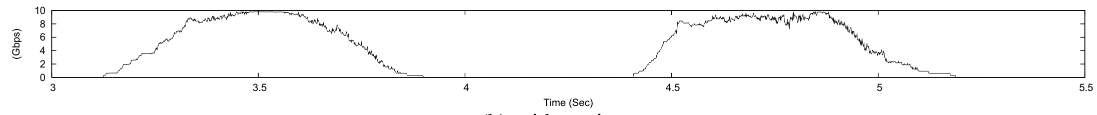


(b) with pacing

Fig. 8. Inter-cluster traffic of class C IS, 0 msec delay.

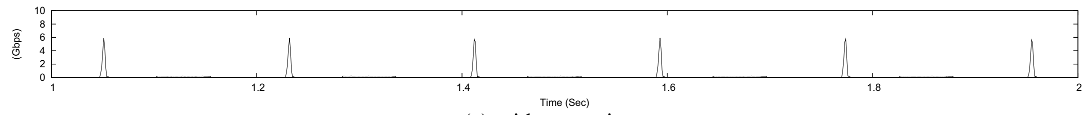


(a) without pacing

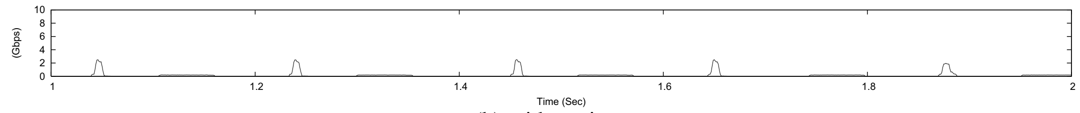


(b) with pacing

Fig. 9. Inter-cluster traffic of class C FT, 0 msec delay.



(a) without pacing



(b) with pacing

Fig. 10. Inter-cluster traffic of class C LU, 0 msec delay.

ing messages of processes that can not communicate directly to the communication nodes. H. Saito, et al., [19] reported that the performance of IS (i.e., *all-to-all* communication pattern) significantly improved on 256 nodes distributed across 4 clusters by decreasing the number of connections.

TCP is widely used as the transport protocol for inter-cluster communications in these systems. However, the performance of TCP communication can be markedly degraded on large bandwidth delay product networks. To address this TCP performance issue, our previous work proposed three improvements in the Linux TCP stack: pacing at start-up, reducing RTO time, and TCP parameter switching at the transition of computation phases in an MPI application [5]. Alternatively, some MPI systems have employed alternative transport protocols, such as the Stream Control Transmission Protocol (SCTP) and UDP based protocols.

H. Kamal, et al., [18] reported that SCTP is well-suited for MPI due to its message-oriented nature, and because a set of congestion control parameters is used by multiple streams. LAM-SCTP provides a direct mapping from SCTP streams to MPI message tags. Under loss conditions, LAM-SCTP outperforms LAM-TCP for latency tolerant programs, because relaxing the ordering of messages can make it robust to network delay and loss. However, while the performance of LAM-SCTP is comparable to that of LAM-TCP in the NPB benchmarks when larger problem sizes are used, it is lower when the problem size is small. MPICH-RBUDP [15] uses Reliable-Blast UDP to achieve high bandwidth utilization over high performance dedicated networks. However, UDP is not friendly to TCP, and thus users may have to use a different communication library when a dedicated network is not available. The method proposed in this paper can be applied to those implementations which use transport protocols different from TCP.

MPICH-GQ [20] provides quality of service mechanisms to manage contention at shared networks and CPUs, and hence improves performance of MPI applications. MPICH-GQ cooperates with edge routers on a Diffserv network, and the aggregate transmission bandwidth of nodes is regulated using a token bucket traffic shaper on the router. To address the TCP performance issue due to bursty traffic of MPI communications, they reported that both a large token bucket buffer and an accurate reservation value is required. In the method we have proposed, communication patterns of application programs are used to decide the regulated transmission rate. By using the PSPacer, transmission rate can be accurately regulated at each node with no microscopic burst transmission, and no hardware support is required.

A. Gulati, et al., [21] proposed NIC-based rate control for proportional bandwidth allocation in Myrinet clusters. Since no packet loss occurs in a Myrinet network, they used rate control for fairness among connections.

VIII. Conclusion

Several Grid-enabled MPI systems designed for multiple clusters connected by a wide-area network have been proposed. Since most of these systems use TCP for inter-cluster communication, improving the performance of TCP communication is the key to successful deployment of MPI programs in a Grid environment. In this paper, to tackle this issue, we have proposed a traffic control method for MPI programs, in which an application or the MPI runtime calculates the maximum allowable transmission bandwidth of a node (MATB) based on the communication pattern, so as not to allow the aggregate transmission bandwidth of the nodes to exceed the inter-cluster bandwidth. To precisely control the transmission rate according to the MATB, we used a packet pacing software program called PSPacer, which achieves precise rate regulation with no microscopic burst transmission.

We have confirmed the effectiveness of the proposed method by experiments with the NPB benchmarks using an emulated WAN environment with 10 Gbps bandwidth and a delay that ranged 0 to 10 msec. Focusing on the results in which the delay is 10 msec, in class B, CG, IS, and MG improved from 35 to 130 percent, FT improved slightly in performance; in class C, BT, CG, FT, IS, MG, and SP improved from 20 to 100 percent in performance; in class D, BT, CG, MG, and SP improved from 10 to 30 percent, and FT showed slightly worse performance; and LU showed slightly worse performance in all classes. This performance improvement is obtained because the proposed method reduces the performance degradation due to traffic congestion. In addition, we have shown that the influence of the delay decreases as the message size increases. The results have indicated that it is feasible to connect clusters and run large-scale scientific applications over distances of up to 1000 kilometers, if an appropriate network is available.

We also discussed whether it is possible to use the MATB to optimize for collective communication operations. For future work, we intend to see that the proposed method is optimized for collective operations designed for fast wide-area networks.

Acknowledgment

Part of this research was supported by a grant from the Ministry of Education, Sports, Culture, Science and Technology (MEXT) of Japan through the NAREGI (National Research Grid Initiative) Project.

References

[1] GridMPI, http://www.gridmpi.org/
[2] YAMPI, http://www.il.is.s.u-tokyo.ac.jp/yampi/
[3] W. L. George, J. G. Hagedorn, and J. E. Devaney, "IMPI: Making MPI Interoperable," Journal of Research of the National Institute of Standards and Technology, Vol.105, N.3, 2000.
[4] M. Matsuda, Y. Ishikawa, and T. Kudoh, "Evaluation of MPI Implementations on Grid-connected Clusters using an Emulated WAN Environment," 3rd Intl. Symp. on Cluster Computing and the Grid (CCGrid2003), pp.10-17, 2003.
[5] M. Matsuda, Y. Ishikawa, T. Kudoh, Y. Kodama, and R. Takano, "TCP Adaptation for MPI on Long-and-Fat Networks," IEEE Intl. Conf. on Cluster Computing (Cluster2005), pp.1-10, 2005.
[6] M. Matsuda, T. Kudoh, Y. Kodama, R. Takano, and Y. Ishikawa, "Efficient MPI Collective Operations for Clusters in Long-and-Fast Networks," IEEE Intl. Conf. on Cluster Computing (Cluster2006), 2006.
[7] R. Takano, T. Kudoh, Y. Kodama, M. Matsuda, H. Tezuka, and Y. Ishikawa, "Design and Evaluation of Precise Software Pacing Mechanisms for Fast Long-Distance Networks," 3rd Intl. Workshop on Protocols for Fast Long-Distance Networks (PFLDnet2005), 2005.

[8] R. Takano, Y. Kodama, T. Kudoh, M. Matsuda, F. Okazaki, and Y. Ishikawa, “Realtime Burstiness Measurement,” 4th Intl. Workshop on Protocols for Fast Long-Distance Networks (PFLDnet2006), 2006.

[9] A. Takefusa, M. Hayashi, N. Nagatsu, H. Nakada, T. Kudoh, T. Miyamoto, T. Otani, H. Tanaka, M. Suzuki, Y. Sameshima, W. Imajuku, M. Jinno, Y. Takigawa, S. Okamoto, Y. Tanaka, and S. Sekiguchi, “G-lambda: coordination of a grid scheduler and lambda path service over GMPLS,” iGrid2005, pp.868-875, 2005.

[10] GtrcNET-10, http://www.gtrc.aist.go.jp/gnet/

[11] Y. Kodama, T. Kudoh, R. Takano, H. Sato, O. Tatebe, and S. Sekiguchi, “GNET-1: Gigabit Ethernet Network Testbed,” IEEE Intl. Conf. on Cluster Computing (Cluster2004), pp.185-192, 2004.

[12] Myricom, Inc., “Myrinet Express (MX): A High-Performance, Low-Level, Message-Passing Interface for Myrinet,” 2003.

[13] E. Gabriel, M. Resch, and R. Rühle, “Implementing MPI with Optimized Algorithms for Metacomputing,” Third MPI Developer’s and User’s Conference (MPIDC’99), 1999.

[14] N. T. Karonis, B. Toonen, and I. Foster, “MPICH-G2: A Grid-Enabled implementation of the Message Passing Interface,” Technical Report ANL/MCS-P942-0402, 2002.

[15] N. T. Karonis, M. E. Papka, J. Binns, J. Bresnahan, and J. M. Link, “Effective Use of Dedicated Wide-Area Networks for High-Performance Distributed Computing,” Technical Report ANL/MCS-P1151-0404, 2004.

[16] O. Aumage and G. Mercier, “MPICH/MadIII: a Cluster of Clusters Enabled MPI Implementation,” 3rd Intl. Symp. on Cluster Computing and the Grid (CCGrid2003), pp.26-36, 2003.

[17] A. Pant and H. Jafri, “Communicating Efficiently on Cluster Based Grids with MPICH-VMI,” IEEE Intl. Conf. on Cluster Computing (Cluster2004), 2004.

[18] H. Kamal, B. Penoff, and A. Wagner, “SCTP versus TCP for MPI,” IEEE/ACM Supercomputing Conference 2005, 2005.

[19] H. Saito and K. Taura, “Locality-aware Connection Management and Rank Assignment for Wide-area MPI,” IEEE CCGrid 2007, 2007.

[20] A. Roy, I. Foster, W. Gropp, N. Karonis, V. Sander, and B. Toonen, “MPICH-GQ: Quality-of-Service for Message Passing Programs,” IEEE/ACM Supercomputing Conference 2000, 2000.

[21] A. Gulati, D. K. Panda, P. Sadayappan, and P. Wyckoff, “NIC-Based Rate Control for Proportional Bandwidth Allocation in Myrinet Clusters,” IEEE Intl. Conf. on Parallel Processing (ICPP2001), 2001.