IPSJ Transactions on Advanced Computing Systems Vol. 2 No. 3 153–162 (Sep. 2009)

Regular Paper

Aggregate Router: An Efficient Inter-Cluster MPI Communication Facility

Hiroya Matsuba[†1] and Yutaka Ishikawa[†2]

At a cluster of clusters used for parallel computing, it is important to fully utilize the inter-cluster network. Existing MPI implementations for cluster of clusters have two issues: 1) Single point-to-point communication cannot utilize the bandwidth of the high-bandwidth inter-cluster network because a Gigabit Ethernet interface is used at each node for inter-cluster communication, while more bandwidth is available between clusters. 2) Heavy packet loss and performance degradation occur on the TCP/IP protocol when many nodes generate short-term burst traffic. In order to overcome these issues, this paper proposes a novel method called the aggregate router method. In this method, multiple router nodes are set up in each cluster and inter-cluster communication is performed via these router nodes. By striping a single message to multiple routers, the bottleneck caused by network interfaces is reduced. The packet congestion issue is also avoided by using high-speed interconnects in a cluster, instead of the TCP/IP protocol. The aggregated router method is evaluated using the HPC Challenge Benchmarks and the NAS Parallel Benchmarks. The result shows that the proposed method outperforms the existing method by 24% in the best case.

1. Introduction

With the increasing demands of applications for computational power and memory space, a single computational cluster sometimes cannot satisfy the user's requirements. A "cluster of clusters" is one popular solution to obtain more resources than a single cluster can provide. A cluster of clusters is a parallel computer that consists of two or more computational clusters (sub-clusters). A typical cluster of clusters, for example, is configured such that each node is connected using Infiniband [12)] or Myrinet [4),20)] for intra-cluster communication, while a Gigabit Ethernet is used to connect to a 10 Gbps inter-cluster network.

†1 Information Technology Center, The University of Tokyo
†2 Graduate School of Information Science and Technology, The University of Tokyo

In order to use a cluster of clusters as a parallel computer, an MPI [18)] communication library must be capable of handling both the intra and inter cluster networks to provide a transparent all-to-all connectivity to MPI applications. Several MPI implementations have already been proposed with such a feature, including PACX-MPI [6)], MPICH-Madeleine [1)], MPICH-VMI [22)], GridMPI [16)], metaMPICH [3)], IMPI [7)], and MPICH-G2 [13)]. In these MPI implementations, however, there are several issues in utilizing the bandwidth of inter-cluster communication: an interface bottleneck issue and TCP-related issues.

Interface Bottleneck Issue
Even sub-clusters are inter-connected with a high-speed network, this interconnect usually does not have any connectivity to an external network. In such a case, a slower network interface is used for the external communication. This bandwidth limitation may become a performance bottleneck.

TCP-Related Issues
Two problems involving the use of the TCP protocol for the MPI communication are known [17)]: the slow start and the tail drop issues. When a TCP connection becomes idle and no packets are transmitted within a certain time, TCP drops into the slow start mode. This behavior is problematic on MPI because MPI traffic often becomes idle during the computation phase of applications. In such a case, the communication phase always starts with the slow start mode and the bandwidth utilization remains low, especially when some packets are lost during the slow start mode. The tail drop issue occurs when a packet is lost and there are no packets following the lost packet. In this case, TCP does not recognize the packet loss until the retransmit timeout occurs. Because this timeout is too long, hundreds of milliseconds in standard TCP implementations, this results in a low bandwidth utilization.

In this paper, we propose a new method called the *aggregate router method*. In this method, inter-cluster communication is performed by selected nodes, called router nodes. In order for other nodes to communicate across sub-clusters, messages are transmitted to the router nodes and forwarded to destination sub-clusters. In a sub-cluster, a high-speed interconnect is used to transmit messages between router nodes and other nodes. A key feature of the aggregate router method is that multiple routers are used in a single sub-cluster. Large messages are divided into several chunks and forwarded to destination clusters using multiple routers. Even if the router nodes have a slower network interface than the inter-cluster network, this interface bottleneck can be eliminated by using multiple routers. All the TCP related issues are avoided by not using the TCP/IP protocol in sub-clusters.

The aggregate router method is evaluated using the HPC Challenge Benchmarks and NAS Parallel Benchmarks. The results show that the proposed method can overcome the interface bottleneck and TCP-related issues. With the MPIFFT, HPL, and CG benchmarks, the aggregate router method outperforms the existing router method by 1.8%, 2.7%, and 24%, respectively. It also outperforms the TCP/IP-based method by 61%, 6.7% and 246%, respectively. The benchmark results with 5 ms, 10 ms, or 15 ms of additional delay in the inter-cluster network show that the aggregate router method is more tolerant to a network delay than TCP/IP-based methods.

2. Background

In order to realize the inter-cluster MPI communication, several implementations of MPI have been proposed. There are three approaches to inter-cluster communication: the all-to-all, the router, and the gateway approaches. **Figure 1** illustrates the all-to-all approach. In this approach, each node has connectivity to any other node in different sub-clusters. This approach is commonly used at the cluster of clusters where each sub-cluster has a high-speed interconnect, while each node also has an Ethernet network to communicate with peer sub-clusters. In this example, the TCP/IP protocol is used for an all-to-all inter-cluster connection. MPICH-Madeleine[1], MPICH-VMI[22], GridMPI[16], and metaMPICH[3] implement this approach, and IMPI[7] assumes this approach. **Figure 2** shows the router approach. Unlike the all-to-all approach, not every node has connectivity to other sub-clusters. Instead, each sub-cluster has router nodes that can communicate with the router nodes of other sub-clusters. Inter-cluster communication is performed via these router nodes. This approach has been used for functionality reasons. In parallel computers that do not have an Ethernet interface on each node, I/O nodes act as router nodes. PACX-MPI[6] and MPICH-G2[13] adopt this approach, and metaMPICH and IMPI adopt this

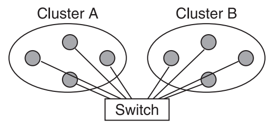


Fig. 1 All-to-all approach.

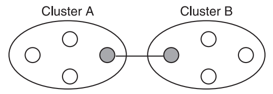


Fig. 2 Router approach.

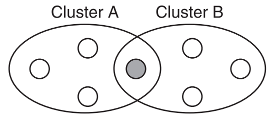


Fig. 3 Gateway approach.

approach as well as the all-to-all approach. The other approach for communicating with other clusters is the gateway approach, which is shown in **Fig. 3**. In this approach, a special node called a gateway node has connectivity to all the nodes. All the inter-cluster communication is performed using this gateway node.

Among these approaches, the all-to-all and router approaches suffer from the interface bottleneck issue because each node or router usually does not have a network interface fast enough to utilize the inter-cluster network. The all-to-all approach is affected also by TCP-related issues when many nodes send data to the bottleneck switch at the same time. We do not consider the gateway approach because the inter-cluster network is not used in this approach.

3. Aggregate Router Method

This section describes a new method to achieve an efficient inter-cluster communication. First, we describe an overview of the approach, which we call the *aggregate router method*. After this, we explain the design details we incorporated to realize the method.

3.1 Design Overview

In order to achieve efficient inter-cluster communication, we extend the router

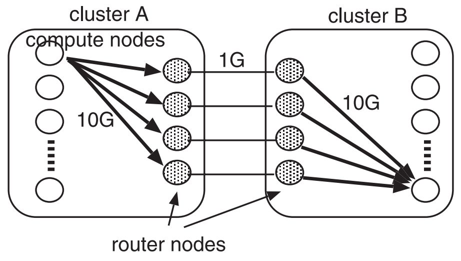


Fig. 4 Aggregate router method.

approach mentioned in the previous section. **Figure 4** shows an overview of the aggregate router method. With this method, multiple routers exist in a sub-cluster. Messages are distributed to these routers and forwarded to destination sub-clusters. In Fig. 4, a *compute node* means a node on which MPI processes run. A *router node* means a cluster node that has connectivity to the external network and forwards data to other sub-clusters. In the rest of this paper, we refer to an MPI process that executes a user's parallel program as a *compute process*. We also use the term *router process* to refer to a process or thread that performs forwarding of data at router nodes. Compute and router processes may reside on the same node, i.e., the compute and router nodes may actually be physically one a single computer.

The aggregate router method works as follows. When a compute node sends a message to other sub-clusters, this message is divided into chunks of arbitrary size. These chunks are forwarded to multiple router processes using a cluster-internal network, such as Infiniband or Myrinet. When the router node receives the chunks, it sends them to the peer router in the destination sub-cluster. The router in the destination sub-cluster sends them to the final destination process using its own cluster-internal network.

Because a single message is transmitted using multiple router nodes, the bandwidth of inter-cluster communication is not restricted to the capacity of the network interface on each compute or router node. Thus, the interface bottleneck issue is avoided. TCP-related issues are eliminated because the TCP/IP protocol is used only for connecting between router processes. Although the TCP protocol is still used for inter-cluster communication, the packet congestion and the tail drop issues are avoided by adjusting the number of router nodes. For example in Fig. 4, because four pairs of routers are connected with a 1 Gbps network each, congestion and packet loss never occur as long as more than 4 Gbps of bandwidth is available for the inter-cluster network.

3.2 Message Striping

In the aggregate router method, a single message is fragmented into multiple chunks and sent to another sub-cluster using multiple routers. In striping a message into several chunks, we must consider two things: the message size and the message ordering.

Generally, the transmission overhead in smaller messages is relatively larger than that in larger messages. For example in the bandwidth benchmark shown in the next section, 8 KB messages are transmitted with 41 MB/s while 4 KB messages are transmitted with 25 MB/s. In this case, if an 8 KB message is fragmented and forwarded to two routers, the transmission rate is 50 MB/s, ignoring all the overhead to divide and reassemble the message. The performance gain is limited to only 18%.

Another issue is message ordering. If messages are divided into multiple chunks and are simply sent via different routers, the chunks might arrive at the destination host out of order. In order for the receiver to assemble the message in the correct order, a sequence number must be attached to each chunk and the chunks must be reordered in the receiver. Therefore, a reordering buffer is required in the receiver. It increases the communication overhead to copy data between the reordering and receive buffers.

Because of these issues, in the aggregated router method, a small size message is simply sent to the destination without fragmentation. A large size message is fragmented, and fragmented chunks are scattered to routers if each chunk is large enough to utilize the physical network bandwidth. In order to reduce the message reordering cost, the transmission mechanism for such a large message is integrated into the MPI rendezvous protocol. With the Rendezvous protocol, when a large message is transmitted, a handshake is performed before the data is actually sent. By this handshake, the address of the receive buffer is fixed. The body part of the data is sent to this receive buffer without any intermediate copy. In the aggregate router method, this protocol is extended. The sender process transmits fragmented chunks to the destination via multiple routers. Each chunk has the address of the receive buffer where the chunk should be stored. The receiver has only to store each chunk in the indicated address and to count the number of arrived chunks to check whether all the chunks have arrived. No reordering overhead is incurred. This mechanism is similar to that introduced for making use of multiple links of interconnect [15]. The threshold between small and large messages is chosen by users. The proper value depends on the performance of network interface cards and router nodes.

3.3 ICBC Communication Library

The aggregate router method is implemented in a new communication library called ICBC (Inter-Cluster Bandwidth Control). ICBC is designed to be used from an MPI communication library. All the information about the topology of a cluster of clusters is handled by ICBC. The MPI implementation and its user may treat the cluster of clusters as a single parallel computer.

Figure 5 shows the protocol stack of ICBC. As described in the previous section, the aggregate router method has compute and router processes. The ICBC communication library has a dedicated implementation for each process.

ICBC at Compute Process

The ICBC implementation on a compute process provides interfaces to be used from an MPI implementation. At the top of the protocol stack, a common ICBC layer exists. This layer is responsible for managing message queues, such as receive or unexpected message queues. Below this common layer, there are transport-specific layers. As the underlying transport protocol, various networks, such as TCP, PM [23], Infiniband (OFED [21]), or Myrinet Express [20], may be used.

In Fig. 5, TCP, InfiniBand, and PM are located under the byte transfer library

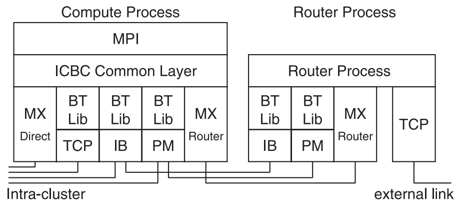


Fig. 5 ICBC protocol stack.

(shown as BT Lib). This library is for the transport layer which provides only reliable and ordered delivery functionality. For such devices, the byte transfer library provides a matching capability, which corresponds to the tag and communicator of MPI. The rendezvous protocol and the message striping capability are also implemented in this library. This byte transfer library is used both for intra and inter cluster communication. The Myrinet Express (MX) library provides its own matching capability and the implementation of the rendezvous protocol. "MX Direct" in Fig. 5 is implemented so that the features of MX are fully utilized for intra-cluster communication. For an inter-cluster communication where MX cannot communicate directly, the router-aware implementation is prepared, which is shown as "MX Router" in Fig. 5. The message striping capability of the aggregate router method is implemented here.

ICBC at Router Process

Instead of the common ICBC layer at compute processes, router processes directly access each intra-cluster communication network. As regards the external connection, any network protocol may be used to connect between router processes as long as it guarantees a reliable and orderly delivery. Typically, the TCP/IP protocol is used here.

Integration with MPI

ICBC is currently integrated with two MPI implementations: YAMPI [26] and MPICH2 [19]. YAMPI is an MPI implementation which is used as a core implementation of GridMPI [16]. In order to integrate the ICBC communication library with YAMPI, we added a new implementation of point-to-point communication to YAMPI. For MPICH2, we implemented the ADI3 interface [8] over the ICBC communication library.

4. Experimental Results

In this section, we evaluate the aggregate router method. First, the basic network performance is measured by using latency and bandwidth benchmarks. After this evaluation, the performance of benchmark applications is measured. The NAS Parallel Benchmarks [2] and the HPC Challenge Benchmarks [9] are used for benchmark applications.

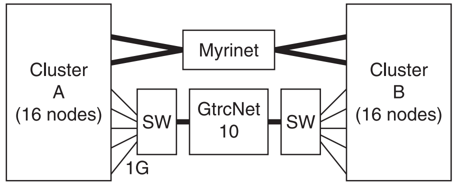


Fig. 6 Cluster configuration.

Table 1 Specifications of cluster nodes.

CPU	Dual-Core Opteron 2212 (2.2 GHz) × 2
Chipset	Broadcom HT1000 + HT2100
Memory	4 GB Dual-channel DDR2 667 MHz
Ethernet	Broadcom NetXtreme (1 Gbps)
Interconnect	Myrinet-10G (Clusters A, B)

4.1 Experiment Setup

The configuration of experimental clusters is shown in **Fig. 6**.

We use two sets of 16-node PC clusters, clusters A and B. Both clusters A and B have a Myrinet-10G interconnect. Although clusters A and B are originally a single 32-node cluster connected by the Myrinet-10G interconnect with full-bisection configuration, we use them as two sets of a 16-node cluster. The detailed specifications of each cluster node are shown in **Table 1**. Each node has a Gigabit Ethernet interface and is connected to Ethernet switches. Each Ethernet switch has a 10 Gbps uplink and this 10 Gbps network is used as the inter-cluster network. In the inter-cluster network, we put a special hardware equipment called GtrcNET-10 [10),14)]. During the application benchmarks, GtrcNET-10 introduces 0 ms (no additional delay), 5 ms, 10 ms, or 15 ms delay to the inter-cluster network to emulate WAN environments. It also measures the bandwidth of the inter-cluster communication.

The aggregate router method is compared with two existing methods for inter-cluster MPI communication; the all-to-all and router approaches as we described in Section 2. The existing approaches are realized in the ICBC communication library together with the aggregate router method. For the router approach, our implementation supports static load balancing. Using this feature, we can set up multiple routers in a single sub-cluster. Unlike the aggregate router method, each compute node always uses one fixed router. As an MPI implementation, MPICH2 over the ICBC communication library is used.

In this section, the following patterns of configuration are used to run each benchmark.

(1) **Aggregate Router with Eight Routers**
All nodes in each 16-node cluster perform as router nodes, and also as computation nodes. Eight router nodes are used for sending data, and the other eight are used for receiving data. In the rest of this section, this configuration is referred to as the "aggregate router configuration" or "AR". The threshold between the small and large messages, which is described in Section 3.2, is set to 32 KB. This is chosen by preliminarily running the ping-pong benchmark with various values.

(2) **Statically Assigned Eight Routers**
The role of each node is identical with that in the aggregate router configuration. The difference from the aggregate router configuration is that each compute process uses one fixed router and a single message is not striped. This configuration is referred to as the "static routers configuration" or "SR".

(3) **All-to-All**
Each process establishes TCP connections to all processes in peer sub-clusters. This configuration is referred to as the "all-to-all configuration" or "AA".

(4) **TCP Only**
All the communication, including intra-cluster communication, is performed using the TCP/IP protocol. This configuration is referred to as "TCP".

(5) **MX Native**
The two clusters are used as a single 32-node cluster connected by Myrinet-10G with a full bisection configuration. This configuration is referred to as "MX".

4.2 Network Performance

For the fundamental network performance, the point-to-point bandwidth and latency of inter-cluster communication, are measured. Among the configurations

Table 2 Round trip time/2 (μs).

AR,SR	AA	MX
64.20	47.21	3.90

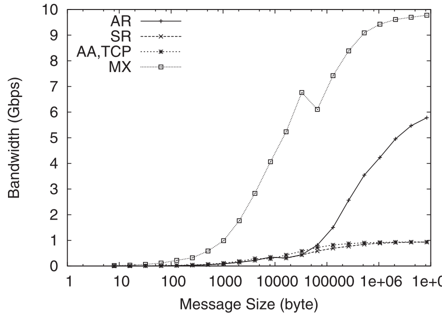


Fig. 7 Bandwidth.

described above, we do not measure the performance with the TCP configuration because the AA and TCP configurations are identical as long as we measure the performance of the inter-cluster communication.

For the latency measurement, a sender transmits an eight-byte message to a receiver in a different sub-cluster and waits for its reply. The time taken by this ping-pong is measured and divided by two. For this benchmark, the behavior of the aggregate router (AR) configuration is identical to that of the static router (SR) configuration because the aggregate router method does not use multiple routers for sending messages of this size. Results are shown in **Table 2**. Because of the overhead required to forward data from a computation node to a router node, and the overhead required to schedule router processes, both the aggregate and static router configurations take more time than the all-to-all configuration.

The point-to-point bandwidth is shown in **Fig. 7**. Because each node has only a Gigabit Ethernet interface for inter-cluster communication, the bandwidth with the static router (SR) and all-to-all (AA) configurations is limited to 1 Gbps, which is the issue we mentioned before as the interface bottleneck. The aggregate router configuration (AR) achieves 5.78 Gbps when the message size is 8 MB. This is 6.2 times faster than the all-to-all configuration.

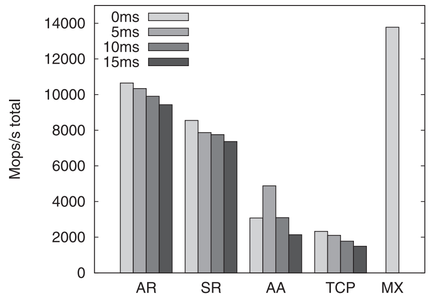


Fig. 8 CG results.

4.3 Application Benchmarks

In order to measure the performance at applications level, we use the CG benchmark in the NAS Parallel Benchmarks [2] and the MPIFFT and HPL benchmarks in the HPC Challenge Benchmarks [9]. For all the benchmarks, we run four MPI processes on each node. Processes with rank 0 to 63 run on cluster A and those with rank 64 to 127 run on cluster B. GtrcNET-10 inserts 0 ms (no additional delay), 5 ms, 10 ms, or 15 ms delay, in a round-trip. GtrcNET-10 also calculates the average bandwidth of inter-cluster communication every 1 ms.

CG

The CG benchmark in NAS Parallel Benchmarks is used with class D. During this benchmark, about 180 MB to 240 MB is used by each process. The results are shown in **Fig. 8**. Comparing four configurations among the no delay cases, the aggregate router (AR) outperforms the static router (SR), all-to-all (AA), and TCP configurations by 24%, 246%, and 358%, respectively. As for the AA configuration, the score with no delay is worse than that with a 5 ms delay. The reason is not clear for now.

Figure 9 shows the traffic sampling taken from the beginning of the benchmark at the no delay cases of each configuration. We can see in the figure that the aggregate router method (AR) fully utilizes the inter-cluster network while the utilization with the static router configuration (SR) is limited to 4 Gbps. This is because only four routers are used in the SR configuration because of the communication pattern in this benchmark. As for the all-to-all configuration (AA), the bandwidth remains below 2 Gbps in most of the time.

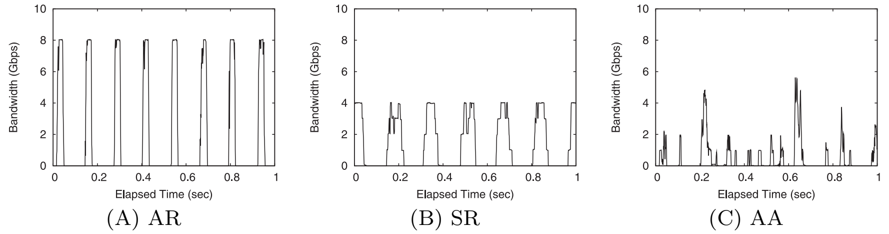


Fig. 9 Traffic sampling of CG.

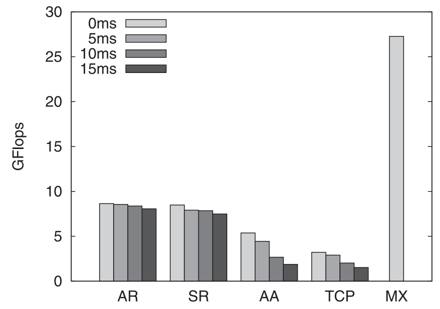


Fig. 10 MPIFFT results.

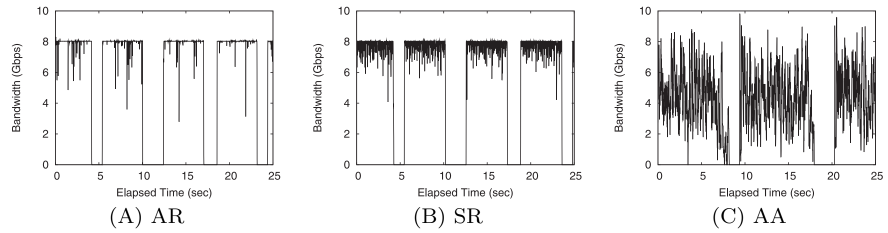


Fig. 11 Traffic sampling of MPIFFT.

Comparing cases with 5 ms and 15 ms delay in each configuration, the performance with AR and SR configurations degrades by 8.8%, 6.4% respectively. With TCP-based methods such as AA and TCP configurations, the performance degrades by 56% and 29%, respectively.

MPIFFT

As a parameter of the MPIFFT benchmark, we set the vector size to 1073741824. With this size, each process uses about 517 MB of memory. The result is shown is **Fig. 10**. Comparing the performance with no additional delay (shown as 0 ms), the aggregate router (AR) outperforms the static router (SR), all-to-all (AA), and TCP configurations with 1.8%, 61%, and 168% respectively. With a 15 ms delay, performance of AR configuration degrades 6.6% compared with the no delay case, while performance of SR, AA, and TCP configurations degrades 12%, 65% and 53%, respectively.

The bandwidth of the inter-cluster communication is shown in **Fig. 11**. These traffic samplings are taken when the benchmarks run with no additional delay. As for the aggregate router (AR) and static router (SR) configurations, the maximum bandwidth is 8 Gbps because we use only eight router processes.

As shown in Fig. 11 (A), the aggregate router method almost fully utilizes the bandwidth of inter-cluster network. As for the static router configuration, Fig. 11 (B) shows that the traffic keeps between 6 Gbps and 8 Gbps. With the all-to-all configuration (AA), as shown in Fig. 11 (C), the traffic is not stable over time and the overall performance is limited. This irregular traffic shows that many packet losses occur during the benchmark. Because the performance penalty caused by packet losses becomes larger as the network delay becomes longer, the performance with the AA and TCP configurations degrades greatly with the long network delay.

HPL

As the parameter of HPL, we set HPL_N to 110000. With this configuration, about 820 MB of memory is used in each process. The results are shown in **Fig. 12**. With no additional delay, the aggregate router configuration (AR) outperforms the static router (SR), all-to-all (AA) and TCP ones by 2.7%, 6.7%, and 20%, respectively. The impacts of network delay are almost same across all configurations.

The traffic sampling is shown in **Fig. 13**. These samplings are taken after 19.5% of the benchmark, i.e., 21,504 columns, is completed. In the figures, three communication phases are marked as (1), (2), and (3). As shown in this table, the AR configuration achieves 8 Gbps in every communication phase while the

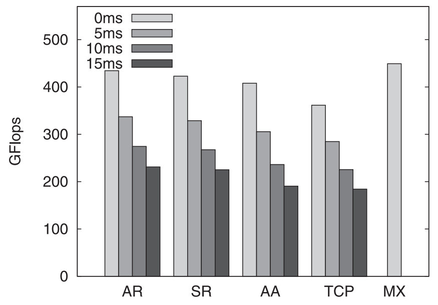
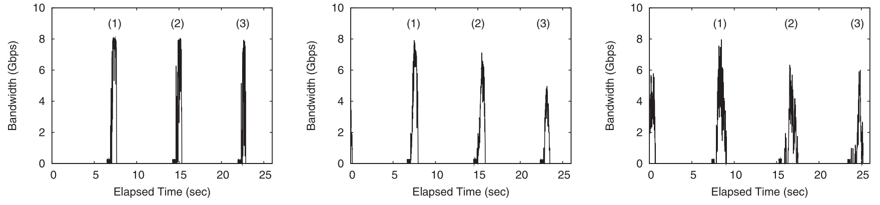


Fig. 12 HPL results.



(A) AR (B) SR (C) AA

Fig. 13 Traffic sampling of HPL.

Table 3 Amount of transmitted data (SR configuration, megabyte).

Router	0	1	2	3	4	5	6	7
Phase(1)	46	46	82	81	83	81	33	32
Phase(2)	29	29	70	70	59	59	40	40
Phase(3)	29	29	0	0	29	29	52	59

bandwidth with other configurations is limited below 8 Gbps. In order to look into the reason why the SR configuration cannot achieve 8 Gbps, we count the amount of data transmitted by each router. **Table 3** shows the result. As shown in the table, the amount of transmitted data varies across eight routers. Because of the interface bottleneck of each router, the utilization of the inter-cluster network remains below 8 Gbps while only heavy loaded routers send data. Especially in the communication phase (3), because only six routers are used, the bandwidth of inter-cluster communication is limited to at most 6 Gbps. On the other hand, because the AR configuration always uses all of the eight routers, it fully utilizes the inter-cluster network throughout the communication phase. The traffic with the AA configuration is not stable as we have seen at the MPIFFT benchmark.

5. Related Work

As mentioned in Section 2, there are several MPI implementations for a cluster of clusters, such as PACX-MPI[6)], MPICH-Madeleine[1)], MPICH-VMI[22)], metaMPICH[3)], and MPICH-G2[13)]. IMPI[7)] is a specification that makes MPI implementations interoperable for a cluster of clusters. LAM/MPI[5)], HP-MPI[11)], and GridMPI[16)] implement IMPI. Among these MPI implementations, the aggregate router method is similar to metaMPICH and GridMPI[25)] in that the router method is adopted and multiple router nodes can be used in each cluster. The difference is that multiple routers are used only for static load balancing with metaMPICH and GridMPI.

GridMPI has its own traffic pacing software[24)] in order to avoid packet congestion at bottleneck switches. However, the bandwidth that may be consumed by each node is different at any one time. For example, when only two nodes communicate with each other, they should use as much bandwidth as possible, whereas pacing is required when many nodes communicate at the same time. In order to set up the traffic pacing software of GridMPI, application programmers must give hints to the pacing software, which is difficult for programmers.

Both Infiniband and Myrinet have switches that can convert the data link protocol; that is, these switches are able to transmit IP over Infiniband or IP over Myrinet packets into an Ethernet network. By using these switches, TCP connections can be established across clusters using high-speed interconnects. This method overcomes the interface bottleneck issue. However, the slow start issue of the TCP protocol still remains.

6. Conclusion

In this paper, we have proposed a new method for an efficient MPI communication on a cluster of clusters. The proposed method is called the aggregate router method. This method overcomes the existing issues on inter-cluster MPI communication: the interface bottleneck and TCP-related issues. The key feature of the proposed method is that a single message is fragmented and forwarded with multiple router nodes. By this method, the load of the routers is equally balanced and the inter-cluster network is efficiently used even if each router node has only a low-bandwidth network interface for inter-cluster communication.

We have developed a new software called the ICBC communication library in order to realize the aggregate router method proposed in this paper. This library is integrated with the two MPI implementations: YAMPI and MPICH2. In an evaluation using the CG benchmark in the NAS Parallel Benchmarks, it has been shown that the proposed method performs better than the the existing router method and the TCP/IP based method by 24% and 246% respectively. At the MPIFFT and HPL benchmarks in the HPC Challenge Benchmarks, the proposed method also outperforms the existing one. In an evaluation with the emulated WAN environment, it has been shown that the aggregate router method is more tolerant of a large network delay than the TCP/IP based method.

The most important contribution of this paper is that we have proposed a new method to fully utilize multiple router nodes regardless of the communication pattern of parallel applications. This paper has also revealed the advantage of the router based method against the method with end-to-end TCP connections. The advantage is significant especially in a long-delay WAN environment.

Although we have evaluated the aggregate router method using MPI parallel applications, this method may be used for other types of inter-cluster communication such as file transfer. Because we have designed the ICBC communication library to be independent from MPI, any application can make use of the ICBC communication library and leverage the advantage of the aggregate router method. Our future work is to make such software and evaluate the performance. Further evaluation of the aggregate router method is also our future work, which includes the evaluation of processor cycles consumed by router processes.

Acknowledgments The authors would like to express their thanks to Dr. Yuetsu Kodama at National Institute of Advanced Industrial Science and Technology (AIST), who kindly let the authors use his equipment, GtrcNET-10.

References

1) Aumage, O. and Mercier, G.: MPICH/MADIII: A Cluster of Clusters Enabled MPI Implementation, *CCGRID '03: Proc. 3st International Symposium on Cluster Computing and the Grid*, Washington, DC, USA, pp.26–33, IEEE Computer Society (2003).
2) Bailey, D., Barszcz, E., Barton, J., Browning, D., Carter, R., Dagum, L., Fatoohi, R., Fineberg, S., Frederickson, P., Lasinski, T., Schreiber, R., Simon, H., Venkatakrishnan, V. and Weeratunga, S.: *The NAS Parallel Benchmarks* (1994).
3) Bierbaum, B., Clauss, C., Poeppe, M., Lankes, S. and Bemmerl, T.: The New Multidevice Architecture of MetaMPICH in the Context of Other Approaches to Grid-Enabled MPI, *PVM/MPI, Lecture Notes in Computer Science*, Vol.4192, pp.184–193 (2006).
4) Boden, N.J., Cohen, D., Felderman, R.E., Kulawik, A.E., Seitz, C.L., Seizovic, J.N. and Su, W.-K.: Myrinet: A Gigabit-per-Second Local Area Network, *IEEE Micro*, Vol.15, No.1, pp.29–36 (1995).
5) Burns, G., Daoud, R. and Vaigl, J.: LAM: An Open Cluster Environment for MPI, *Proc. Supercomputing Symposium*, pp.379–386 (1994).
6) Gabriel, E., Resch, M., Beisel, T. and Keller, R.: Distributed Computing in a Heterogeneous Computing Environment, *Proc. 5th European PVM/MPI Users' Group Meeting on Recent Advances in Parallel Virtual Machine and Message Passing Interface*, London, UK, Springer-Verlag, pp.180–187 (1998).
7) George, W.L., Hagedorn, J.G. and Devaney, J.E.: IMPI: Making MPI Interoperable, *Journal of Research of the National Institute of Standards and Technology*, Vol.105, No.3, pp.343–428 (2000).
8) Gropp, W., Lusk, E., Ashton, D., Ross, R., Thakur, R. and Toonen, B.: MPICH Abstract Device Interface Version 3.4 Reference Manual: Draft of May 20 (2003).
9) HPCC. http://icl.cs.utk.edu/hpcc/
10) GtrcNET-10. http://www.gtrc.aist.go.jp/gnet/
11) HP-MPI. http://www.hp.com/
12) InfiniBand. http://www.infinibandta.org
13) Karonis, N.T., Toonen, B. and Foster, I.: MPICH-G2: a Grid-enabled implementation of the Message Passing Interface, *J. Parallel Distrib. Comput.*, Vol.63, No.5, pp.551–563 (2003).
14) Kodama, Y., Kudoh, T., Takano, R., Sato, H., Tatebe, O. and Sekiguchi, S.: GNET-1: gigabit Ethernet network testbed, *CLUSTER '04: Proc. 2004 IEEE International Conference on Cluster Computing*, Washington, DC, USA, pp.185–192, IEEE Computer Society (2004).
15) Liu, J., Vishnu, A. and Panda, D.K.: Building Multirail InfiniBand Clusters: MPI-Level Design and Performance Evaluation, *SC '04: Proc. 2004 ACM/IEEE Conference on Supercomputing*, Washington, DC, USA, p.33, IEEE Computer Society (2004).
16) Matsuda, M., Kudoh, T. and Ishikawa, Y.: Evaluation of MPI Implementations on Grid-connected Clusters using an Emulated WAN Environment, *CCGRID '03: Proc. 3rd International Symposium on Cluster Computing and the Grid*, Washing-

ton, DC, USA, IEEE Computer Society (2003).
17) Matsuda, M., Kudoh, T., Kodama, Y., Takano, R. and Ishikawa, Y.: TCP Adaptation for MPI on Long-and-Fat Networks, *Proc. IEEE International Conference on Cluster Computing* (2005).
18) The Message Passing Interface (MPI) standard. http://www-unix.mcs.anl.gov/mpi/
19) MPICH2 Design Document, Technical report, Argonne National Laboratory (2002). Draft.
20) Myrinet. http://www.myri.com
21) The OpenFabrics Alliance. http://www.openfabrics.org/
22) Pant, A. and Jafri, H.: Communicating efficiently on cluster based grids with MPICH-VMI, *CLUSTER '04: Proc. 2004 IEEE International Conference on Cluster Computing*, Washington, DC, USA, pp.23–33, IEEE Computer Society (2004).
23) Takahashi, T., Sumimoto, S., Hori, A., Harada, H. and Ishikawa, Y.: PM2: High Performance Communication Middleware for Heterogeneous Network Environments, *SC2000: Proc. 2000 ACM/IEEE Conference on Supercomputing* (2000).
24) Takano, R., Kudoh, T., Kodama, Y., Matsuda, M., Tezuka, H. and Ishikawa, Y.: Design and Evaluation of Precise Software Pacing Mechanisms for Fast Long-Distance Networks, *Proc. PFLDnet 2005* (2005).
25) Takano, R., Matsuda, M., Kudoh, T., Kodama, Y., Okazaki, F., Ishikawa, Y. and Yoshizawa, Y.: High Performance Relay Mechanism for MPI Communication Libraries Run on Multiple Private IP Address Cluster, *Proc. International Symposium on Cluster Computing and the Grid* (*CCGrid2008*) (2008).
26) YAMPI Official Home Page. http://www.il.is.s.u-tokyo.ac.jp/yampi/

(Received January 27, 2009)
(Accepted April 10, 2009)

Hiroya Matsuba received his Master of Computer Science degree in 2005 from the University of Tokyo. He is an Assistant Professor at Information Technology Center of the University of Tokyo. His research interests include high-performance interconnections, communication libraries, and operating systems.

Yutaka Ishikawa received his Doctor of Engineering degree in 1987 from Keio University. He worked at the Electrotechnical Laboratory from 1987 and worked on Real World Computing Project from 1993. He is a Professor of Department of Computer Science, the University of Tokyo. His research interests include cluster system software, dependable software, real-time distributed system, and next-generation supercomputers.