Evaluation of MPI Implementations on Grid-connected Clusters using an Emulated WAN Environment

Motohiko Matsuda Tomohiro Kudoh Yutaka Ishikawa
Grid Technology Research Center, AIST *University of Tokyo*

Abstract

The MPICH-SCore high performance communication library for cluster computing is integrated into the MPICH-G2 library in order to adapt PC clusters to a Grid environment. The integrated library is called MPICH-G2/SCore. In addition, for the purpose of comparison with other approaches, MPICH-SCore itself is extended to encapsulate its network packet into a UDP packet so that packets are delivered via L3 switches. This extension is called UDP-encapsulated MPICH-SCore. In this paper, three implementations of the MPI library, UDP-encapsulated MPICH-SCore, MPICH-G2/SCore, and MPICH-P4, are evaluated using an emulated WAN environment where two clusters, each consisting of sixteen hosts, are connected by a router PC. The router PC controls the latency of message delivery between clusters, and the added latency is varied from 1 millisecond to 4 milliseconds in round-trip time. Experiments are performed using the NAS Parallel Benchmarks, which show UDP-encapsulated MPICH-SCore most often performs better than other implementations. However, the differences are not critical for the benchmarks. The preliminary results show that the performance of the LU benchmark scales up linearly with under 4 millisecond round-trip latency. The CG and MG benchmarks show the scalability of 1.13 and 1.24 times with 4 millisecond round-trip latency, respectively.

1. Introduction

By using communication mechanisms for a Grid environment, such as MPICH-G2 [2], an application can now run across clusters spread over campus-area, metropolitan-area, and wide-area networks, that is, a meta-computing environment is ready. However, there have been few reports on the performance of such systems [2, 11, 15]. Some papers, such as [15], report that when an application runs across the Internet or dedicated long-distance networks, its performance does not scale. It is conjectured that performance degradation is due to several reasons, such as large latency, low bandwidth, TCP/IP problems, cost of securing messages, unpredictable network load, and so on.

In cluster computing, on the other hand, the MPICH-SCore MPI library and the PM/Ethernet low-level communication library on Ethernet have been successfully introduced [10, 13, 14]. PM/Ethernet overcomes the performance problems of TCP/IP and thus achieves better performance than TCP/IP. However, it cannot be employed in a Grid environment because its network protocol is not on top of the IP protocol. *UDP-encapsulated MPICH-SCore* is an extension to MPICH-SCore in which PM/Ethernet's network packet is encapsulated in a UDP packet so that the packets are delivered via L3 switches.

In order to pursue a high performance communication environment on Grid-connected clusters, MPICH-SCore is integrated with MPICH-G2 and it is presented in this paper. The resulting library is called *MPICH-G2/SCore*. MPICH-G2 assumes that communication between computers from different organizations is carried out by Globus I/O [5], while communication within a parallel computer is realized by a platform's MPI implementation, a so-called *vendor MPI*, supported on the parallel computer. MPICH-SCore is used as a vendor MPI in MPICH-G2/SCore.

As a first step to understanding existing drawbacks and to designing a new communication architecture for a Grid-connected clusters environment, network latency and its impact on MPI applications should be taken into account in MPI implementations. In this paper, three implementations of MPI are evaluated: UDP-encapsulated MPICH-SCore, MPICH-G2/SCore, and MPICH-P4. These MPI implementations are evaluated under the setting of an emulated WAN environment, in which two clusters, each consisting of 16 hosts, are connected by a router PC. The router PC controls the latency of message delivery using an Internet emulation tool NIST Net [9]. Experiments are performed on the NAS Parallel Benchmarks [12, 8], where the latency between clusters is varied from 1 millisecond to 4 milliseconds in round-trip time. In the experiments, UDP-encapsulated MPICH-SCore most often performs better than other MPI implementations because it avoids the use of the TCP/IP protocol stack and does not support a security mechanism unlike MPICH-G2.

This paper first introduces MPI implementations, UDP-encapsulated MPICH-SCore, MPICH-G2/SCore, and MPICH-P4, in the following section. Then, the experimental setting is described in Section 3. The results of communication bandwidth and experiments involving the NAS Parallel Benchmarks are shown and discussed in Section 4. Related work on MPI and their execution in a Grid environment is summarized in Section 5. This paper concludes with a discussion of future research in Section 6.

2. MPI Implementations

2.1. Base Implementation of MPICH

MPICH [6] is a portable MPI implementation in which an abstract network device is introduced so that a variety of communication facilities can be easily integrated. MPICH-SCore, MPICH-G2 and MPICH-P4 are all based on MPICH.

There are two protocols used according to the message size: the *eager* and *rendezvous* protocols. In the eager protocol, as soon as a send message is posted, all of the message data is sent to the receiver. On the receiver side, if a corresponding receive has not yet been posted, the data must first be buffered in memory. Later on, when a receive is posted, the buffered data is copied to the user buffer. It works with a single copy from a buffer of the network interface when the message is expected, otherwise it needs an extra copy when the message is unexpected.

In the rendezvous protocol, on the other hand, when a send is posted, only a message envelope is sent and buffered on the receiver. When a corresponding receive is posted, the receiver informs the sender that it can handle the message. Because the receiver is aware of the location of the user buffer, it can always copy the data directly to the user buffer without any intermediate copying. Although the copying cost is eliminated, it incurs the cost from round-trip latency. If the round-trip latency is large like in a Grid environment, it will degrade performance.

2.2. UDP-encapsulated MPICH-SCore

MPICH-SCore is implemented on top of the PM low-level communication library for cluster computing. PM implements such devices as Myrinet, Ethernet, and shared memory systems, and it virtualizes the devices and provides a single programming interface. In this paper, an implementation on Ethernet, called PM/Ethernet, is focused on.

PM/Ethernet is a kernel-level driver designed on top of Ethernet drivers [13]. Reliable communication is realized by the PM/Ethernet driver. To coexist with the PM and IP protocols, an extra layer is introduced on top of the Ethernet drivers so that IP packets are passed to the IP handler, while PM packets are passed to the PM/Ethernet driver. PM introduces a light-weight reliability protocol based on the go-back N protocol. To reduce the overhead of interrupt handling, the *interrupt reaping* technique [13] is introduced, which eliminates interrupts when an application waits for a message. Polling routine of the PM driver invokes an interrupt handler of an underlying Ethernet device, which in effect polls the device and consumes in-coming messages without interrupt handling.

UDP-encapsulated MPICH-SCore is an extension to MPICH-SCore such that a PM packet is encapsulated in a UDP packet. This encapsulation is needed because the PM protocol is not designed to work on top of the TCP/IP protocol stack, but to work on the Ethernet MAC frame. MPICH-SCore has an option to control the selection of a protocol, either eager or rendezvous. Message size at which the protocols are switched is a tunable parameter of the runtime system.

In the following, all experimental results on MPICH-SCore are actually for the UDP-encapsulated MPICH-SCore, and thus it may be referred to as MPICH-SCore for short.

2.3. MPICH-G2/SCore

MPICH-G2 is a version of MPICH for a Grid environment, which employs the Globus I/O [5] which, in turn, is based on the Nexus communication layer [4] and is implemented on top of TCP/IP and the SSL (Secure Socket Layer). Communication primitives of MPICH-G2 almost all call ones in Globus I/O directly. In addition, MPICH-G2 can be configured to use another MPI implementation available on the platform, which establishes a layered configuration as a cluster of parallel computers or a cluster of clusters. Messaging libraries are switched with regard to the destination of a message: messages targeted inside a local system are sent by a subordinate platform's MPI (a so-called *vendor MPI*), or messages targeted outside a local system are sent by Globus I/O. MPICH-G2 has an overhead because Globus I/O secures messages by executing SSL's message digest facility.

MPICH-G2/SCore is an implementation using MPICH-SCore as a vendor MPI implementation. Thus, it is implemented so that inter-cluster communication is handled by Globus I/O, but communication inside a cluster is handled by MPICH-SCore.

In the following, all experiments using MPICH-G2 are done using MPICH-G2/SCore, and thus, it may be referred to as MPICH-G2 for short.

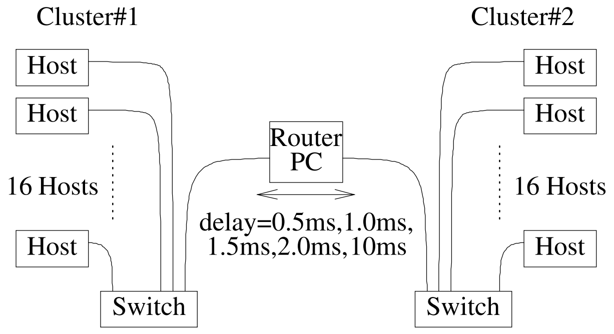

Figure 1. Experimental Setting

Table 1. Cluster Host Spec.

Processor	Pentium III 933 MHz
Memory	1 GByte
I/O Bus	66 MHz PCI-64
Network	3Com Gigabit Ethernet NIC
OS	Linux 2.4.18

Table 2. Router PC Spec.

Processor	Dual Xeon 2.4 GHz
Memory	1 GByte
I/O Bus	66 MHz PCI-64
Network	3Com Gigabit Ethernet NIC
OS	Linux 2.4.20

2.4. MPICH-P4

MPICH-P4 is an implementation using TCP/IP. It is layered over the p4 communication library [1], thus it is called MPICH-P4.

3. Experimental Setting

A Grid environment was emulated using two clusters, each of which consists of 16 hosts, connected by a router PC. The router PC emulates a WAN-like environment. Figure 1 shows the setting, and Table 1 shows the specification of the hosts in the clusters. The network is built with Gigabit Ethernet. The switches are 3Com's SuperStack 4924.

Table 2 shows the specification of the router PC, which has two Gigabit Ethernet NICs to bridge the two clusters. The router PC runs the NIST Net network emulator [9], which controls communication between the two clusters. The NIST Net is a general-purpose tool for emulating dynamics in the Internet. It is implemented as a kernel module extension to the Linux kernel, and is capable of controlling network behavior such as latency, bandwidth, and packet loss. In the experiments, only network latency was considered. The router PC uses a very recent version of the Linux kernel because it has much better performance in networking than its predecessors. It attains almost the peak performance level of the Gigabit Ethernet hardware.

The delay added by the network emulator was varied from 0 milliseconds to 2 milliseconds in the experiments, specifically, 0ms, 0.5ms, 1.0ms, 1.5ms, 2ms, and optionally, 10ms. Note that the delay values are shown as the round-trip latency later in this paper, those values are displayed as multiplied by two.

The TCP buffer size is set to 2 MBytes instead of the default 128 KBytes in all measurements using TCP/IP. This is because TCP performance is sensitive to buffer size in networks with large latency, and the default value is too small for some milliseconds. From the point of bandwidth-latency product, 2 MBytes is sufficient for the range of latency used in experiments.

Similarly, PM/Ethernet has a tunable parameter corresponding to the buffer size of TCP/IP. It is the count of messages sent in burst before receiving acknowledgements. The value is set large enough to tolerate latency in the experiments, too. Other parameters are kept as default except explicitly mentioned.

As mentioned, MPICH-G2's Globus I/O uses SSL for securing messages and has overhead. However, it does not fully utilize the security facility. In the default setting of the current distribution of Globus (Globus 2.0), the security layer does not encrypt messages, but only protects the integrity of messages by using message digests. In the following experiments, this default setting is used.

Versions of systems used are: SCore 5.2 and MPICH-1.2.0 for MPICH-SCore (included in the SCore 5.2 distribution); MPICH-1.2.4 for MPICH-G2 and MPICH-P4; Globus is Globus 2.0; The Fortran compiler is GNU GCC 2.96, and the benchmarks are compiled with option -O3; The network emulator is NIST Net 2.0.

Some benchmarks in the NAS Parallel Benchmarks are omitted from the result. The FT benchmark is omitted because GCC fails to compile it. The BT and SP benchmarks are also omitted because they need the number of processors to be square and thus they are not appropriate for benchmarking in this paper.

4. Evaluation

4.1. Low-Level Communication Bandwidth

Bandwidth depends on latency even in burst-mode transfer, because it is assumed that the communication hardware is unreliable and thus a communication library implements

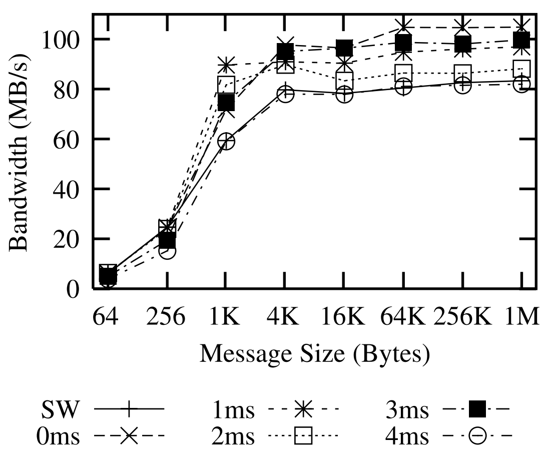

Figure 2. PM/Ethernet Bandwidth

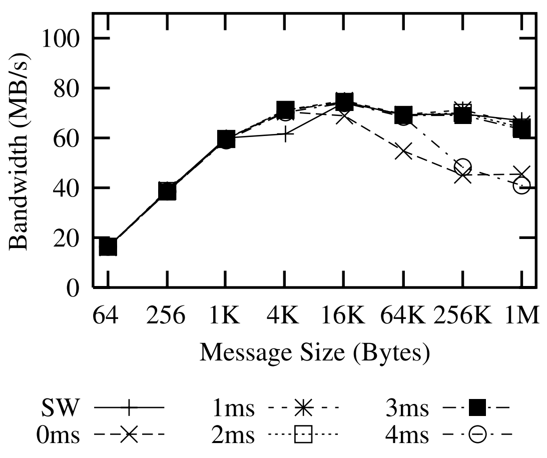

Figure 3. TCP/IP Bandwidth

a retransmission protocol in one layer. PM/Ethernet ensures reliability in MPICH-SCore, and TCP/IP ensures reliability in MPICH-G2 and MPICH-P4.

First, the bandwidth of the low-level communication layers is shown. Figures 2 and 3 show the point to point communication bandwidth in PM/Ethernet and TCP/IP. In this measurement, round-trip latency was varied from 0 milliseconds to 4 milliseconds. Measurement was performed by sending messages of a given size in bursts. In the figure, data points are labeled by round-trip latency in milliseconds. The label *SW* shows the case of two clusters directly connected by an Ethernet switch and without the router PC. In TCP/IP, the *no delay* option is set because MPICH-G2 and MPICH-P4 both use that option.

The maximum bandwidth of PM/Ethernet is 110 MB/s, while the maximum bandwidth of TCP/IP is 80 MB/s. Comparing Figures 2 and 3, TCP/IP is faster than PM/Ethernet for small messages. However, TCP/IP degrades performance. In the case of the 4 millisecond round-trip latency, TCP/IP bandwidth is only 40 MB/s at a 1 MByte message size.

The bandwidth of PM/Ethernet in the switch-connected case is the lowest in Figure 2. It is a consequence of the behavior of interrupt reaping affected by the network emulator. As previously described, interrupt reaping reduces the number of interrupts by polling the Ethernet device, but its effect is sensitive to the timing of polling and arrival of messages. Statistics reveals that more interrupts are generated for the switch-connected case.

Strangely, it is observed that the bandwidth of TCP/IP drops on large messages for the zero-latency case, and gets below the latency-added cases in Figure 3. Reasons are unclear, but the behavior was stable and repeated.

4.2. MPI-Level Communication Bandwidth

Figures 4, 5, and 6 show the point to point communication bandwidth of the MPI implementations, MPICH-G2/SCore, MPICH-P4, and (UDP-encapsulated) MPICH-SCore, respectively. These results reveal the basic characteristics of the implementations. Labels show the round-trip latency and the label *SW* means an Ethernet switch, as before. It should be noted that both of MPICH-SCore and MPICH-G2/SCore use PM/Ethernet in local communication, and thus the data points labeled *SW* show similar curves in Figures 4 and Figure 6. Other cases are via TCP/IP in MPICH-G2/SCore.

Both MPICH-G2/SCore and MPICH-P4 are layered on top of TCP/IP, the performance of which is bounded by the underlying Linux's TCP/IP implementation. However, both show relatively lower bandwidth than the raw TCP/IP, and also show the influence of the latency. The performance of MPICH-P4 shows a drop at 256 KBytes as shown in Figure 5. This drop does not exist in the raw TCP/IP. It is because the rendezvous protocol is used for messages larger than 128 KBytes, and the latency affects the bandwidth. Unlike MPICH-SCore, MPICH-P4 is not capable of controlling the switching point of the protocols at runtime.

Comparing the performance of MPICH-G2/SCore with MPICH-P4 reveals the overhead for security and adaptation to a Grid environment. Although the performance is slightly disturbed in MPICH-P4, mostly it performs better than MPICH-G2/SCore.

MPICH-SCore scales up to 16 KByte messages. However, the bandwidth drastically drops at 16 KBytes as shown in Figure 6. This is because at 16 KBytes, the MPICH-SCore communication protocol changes from the eager protocol to the rendezvous protocol, by default. This result shows the fact that the rendezvous protocol yields extra

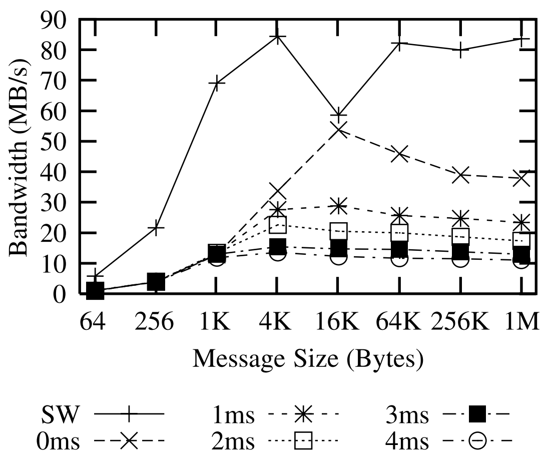

Figure 4. MPICH-G2/SCore Bandwidth

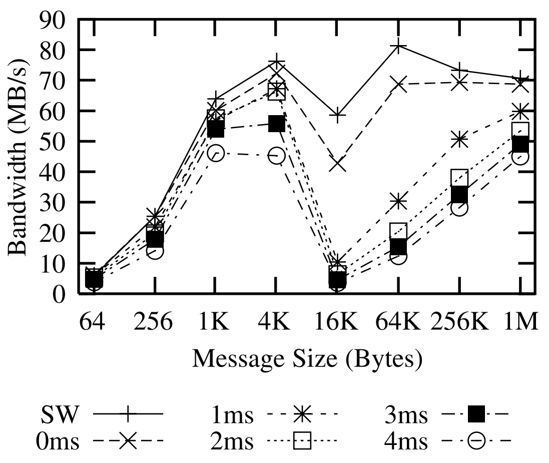

Figure 6. MPICH-SCore Bandwidth

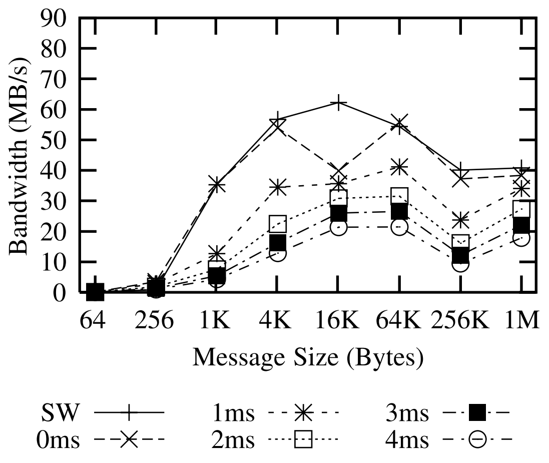

Figure 5. MPICH-P4 Bandwidth

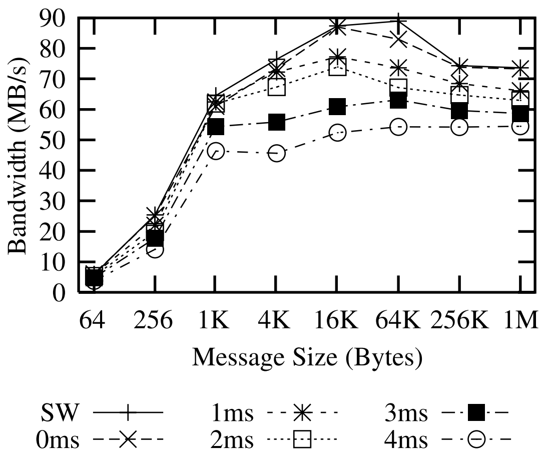

Figure 7. MPICH-SCore(Eager) Bandwidth

communication delay as described in Section 2.1. Figure 7 shows the result when all message are delivered using the eager protocol.

4.3. NAS Parallel Benchmarks

To evaluate the latency impact on applications, the NAS Parallel Benchmarks Suite (NPB2.3) [8] was run on (UDP-encapsulated) MPICH-SCore, MPICH-G2/SCore, and MPICH-P4. The problem size of the benchmarks is class B. Round-trip latency was varied from 0 milliseconds to 4 milliseconds. The results include cases of 20 milliseconds for reference.

The benchmark results are shown by scalability, which is a ratio to the performance using MPICH-SCore with 16 hosts. It normalizes the performance and makes it easy to compare different benchmarks. Since the experiments were performed by 32 hosts, the value 2 is the perfect case. If scalability is below one, there is no merit in connecting two clusters. Note that the scalability values in a graph reflect absolute performance, because the ratio bases on the common value. It means scalability for MPICH-P4 per se is shown slightly lower than the actual value. However, there is no such mismatch for MPICH-G2/SCore, because the actual communication of MPICH-G2/SCore is performed by MPICH-SCore within a cluster. The absolute values for 16 hosts are shown in Table 3. The legend *MPICH-SCore* in the table and the graphs actually means UDP-encapsulated MPICH-SCore.

EP Figure 8 shows the result of experiments run using the EP benchmark. The result is obvious. EP is an Em-

Table 3. NAS Parallel Benchmarks Results (Mops/s total) for 16 Hosts

	MPICH-SCore	MPICH-P4
EP	32.91	32.67
CG	503.01	486.80
LU	57.41	46.44
MG	1709.27	1705.33
IS	1185.43	1176.51

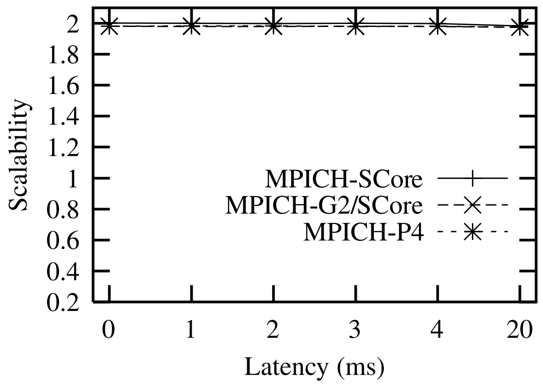

Figure 8. EP

barrassingly Parallel problem and latency does not affect its performance in all MPI implementations.

CG Figure 9 shows the result of experiments run using the CG benchmark. The performance of MPICH-SCore fairly scales at small latency and gradually degrades with a slight gain up to a latency of 4 milliseconds (1.13 times faster at 4ms). While MPICH-SCore shows consistent performance degradation, MPICH-G2/SCore and MPICH-P4 have a performance drop at zero latency. The reason is unclear but the results were repeatedly observed.

MG Figure 10 shows the result of experiments run using the MG benchmark. The performance of MPICH-SCore fairly scales at a small latency like CG. In MPICH-SCore, the MG performance has a gain up to a latency of 4 milliseconds (1.24 times faster at 4ms). MPICH-P4 also shows similar behavior. However, MPICH-G2/SCore does not scale in this benchmark.

LU Figure 11 shows the result of experiments run using the LU benchmark. LU shows good scalability in all implementations. Latency has little impact on the performance. It still has gain even at 20 milliseconds (1.76 at 20ms for MPICH-SCore). Furthermore, it shows super-linearity at

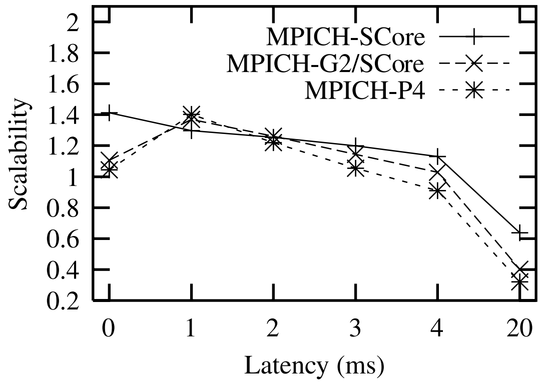

Figure 9. CG

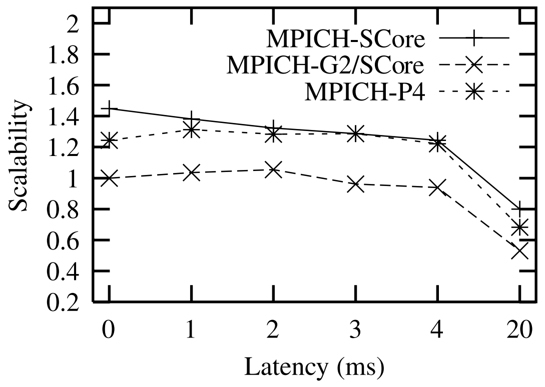

Figure 10. MG

low latency. This result shows that the latency hiding is very effective for LU [16].

IS Figure 12 shows the result of experiments run using the IS benchmark. IS's low performance is due to lack of bandwidth. Although IS is known to heavily use all-to-all communication, the connection between clusters was a single Gigabit link in the experiment, which is unbalanced compared to a connection inside a cluster using a switch.

Unlike the experiments of other benchmarks, the performance of MPICH-SCore is worse than that of other MPI implementations. The IS benchmark uses all-to-all communication, and MPICH-SCore's all-to-all communication is not proved as optimal. In the MPICH-SCore implementation, each data item is sent to each destination, host by host. In other words, as soon as data transfer to one destination has completed, next data transfer to another destination starts. That is, the operations proceed sequentially. If data transfer gets stuck, the following transfers will be delayed. In contrast, both MPICH-G2/SCore and MPICH-P4 use the TCP/IP protocol. The protocol layer may buffer the trans-

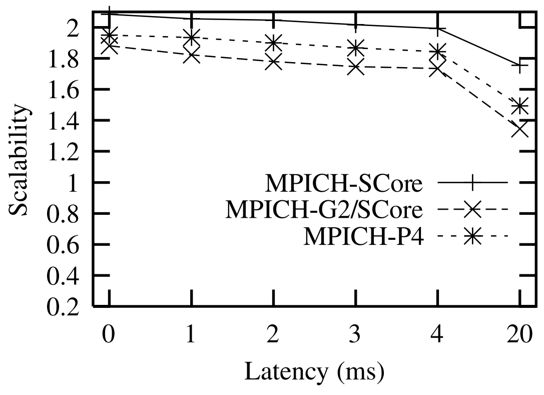

Figure 11. LU

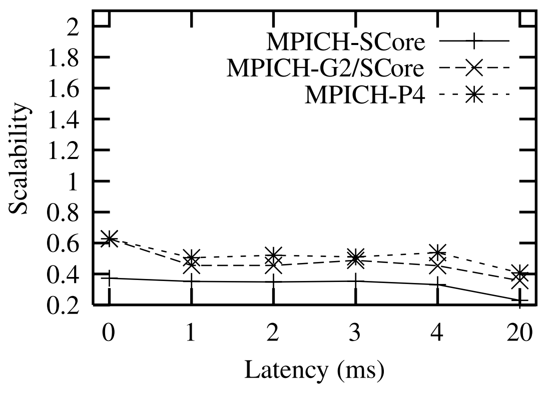

Figure 12. IS

fered data, and TCP/IP can function asynchronously. Thus, even if data transfer to one destination gets stuck, other on-going transfers may proceed.

Overall Result These results show that the benchmarks fairly tolerate a latency of this range. On the other hand, implementations show large differences, and overhead for securing messages and adapting to wide-area networks affects much. These experiments assume the heterogeneously configured network where only a link between clusters is slow. The benchmarks would show different behavior if they are run on homogeneously high-latency networks. That is, for example, the LU benchmark would be slower on such networks because it depends on neighbor communications. Similarly, since the limited bandwidth makes the performance of the IS benchmark poor, it will show different behavior if it is run on networks with higher bandwidth such as 10 Gbps which is getting available in a WAN.

5. Related Work

An evaluation of MPICH-G, the predecessor of MPICH-G2, is given in the paper [2]. However, the evaluation is not in a Grid environment, but on IBM SP2.

The paper on Ninf [15] extensively evaluates benchmark performance in metropolitan-area settings. Since the infrastructure of that time was very poor compared to today's standards, it does not meet our assumptions on latency and bandwidth. Also, it naturally used applications of a client-server type.

Karonis et al [7] reported MPI performance in wide-area settings. They developed a high performance topology-aware communication library, but it only shows performance for a broadcast operation.

6. Conclusions

In this paper, in order to pursue a high performance communication environment on Grid-connected clusters, MPICH-G2/SCore has been introduced. For the purpose of comparison with other approaches, MPICH-SCore has been extended to encapsulate its private network packet into a UDP packet so that packets are transfered via L3 switches. This extension is called UDP-encapsulated MPICH-SCore.

As a first step to understanding existing drawbacks and to designing a new communication architecture in a Grid-connected clusters environment, UDP-encapsulated MPICH-SCore, MPICH-G2/SCore, and MPICH-P4 have been evaluated in an emulated WAN environment using a network emulator.

UDP-encapsulated MPICH-SCore has better performance than other MPI implementations. However, differences are not critical. The results of experiments using the NAS Parallel Benchmarks, the problem size of which is class B, have shown that good performance is achieved under a 4 millisecond round-trip latency in EP and LU. The performance using the CG and MG benchmarks under 4 milliseconds show acceptable scalability as compared to the zero-latency case. Even though the scalabilities of applications may relatively be low, users will benefit from the use of larger resources, such as memory, that are available in multiple clusters.

The 4 millisecond round-trip latency covers roughly 400 kilo-meters in theory. The actual coverage is less than that, because several repeaters and routers exists in the line. For example, a round-trip takes about two milliseconds for 60 kilo-meter distance between Tokyo and Tsukuba, in the real case. It implies that a 4 millisecond round-trip latency will cover whole metropolitan-area.

Of course, the impact of latency depends on many aspects not considered in this paper, such as CPU performance, cluster size, problem size, communication granularity, and so on. If we would have used faster machines, some millisecond latency might become a total penalty. The permissible range of latency depends on machine performance and communication characteristics. Thus, in other words, according to the results, 4 millisecond round-trip latency is permissible under the current technology.

Only latency is considered in this paper. In the Internet, congestion with other users and packet loss may have to be considered. However, we can ignore such behavior in metropolitan-area networks, because bandwidth reservation services, such as WDM and TDM, are available in such networks. Thus, we may conclude that meta-computing using Grid-connected clusters is realistic in metropolitan-area networks using the current technology.

Towards actual use of this work, two further research projects will be carried out. First, a scheduler among distant clusters shall be developed to co-schedule applications. Second, a latency-aware implementation of the MPI library shall be designed, where the variance of latency is taken into account. As shown, MPICH-G2 and MPICH-P4 give less considerations to latency issues, or at least, the assumptions concerning environments mismatch. MPICH-SCore provides higher performance than others, whereas it was designed with very short latency in mind. It will be addressed in further study such issues as security and heterogeneity of computers as well as heterogeneity of networks.

Acknowledgements

This work is supported by the METI network computing project.

References

[1] Ralph Butler and Ewing Lusk, "User's guide to the p4 parallel programming system," Tech. Rep. ANL-92/17, Argonne National Laboratory, 1992.

[2] Ian Foster and Nicholas T. Karonis, "A Grid-Enabled MPI: Message Passing in Heterogeneous Distributed Computing Systems," SC'01, IEEE, 2001.

[3] Ian Foster and Carl Kesselman eds, *The Grid: Blueprint for a Future Computing Infrastructure*, Morgan Kaufmann Publishers, 1999.

[4] Ian Foster, Carl Kesselman, and Steven Tuecke, "The Nexus Approach to Integrating Multithreading and Communication," Journal of Parallel and Distributed Computing, Vol.37 pp.70–82, 1996.

[5] Ian Foster and Carl Kesselman, "Globus: A Toolkit-based Gird Architecture," in [3] pp.259–278, 1999.

[6] William Gropp, Ewing Lusk, Nathan Doss, and Anthony Skjellum, "A high-performance, portable implementation of the MPI message passing interface standard," Parallel Computing, Vol.22, No.6, pp.789–828, 1996.

[7] Nicholas T. Karonis, Bronis R. de Supinski, Ian Foster, William Gropp, Ewing Lusk, and John Bresnahan, "Exploiting Hierarchy in Parallel Computer Networks to Optimize Collective Operation Performance," Proc. of the 14th International Parallel Distributed Processing Symposium (IPDPS '00), May 2000.

[8] NAS Parallel Benchmarks.
http://science.nas.nasa.gov/Software/NPB

[9] NIST Net. http://snad.ncsl.nist.gov/itg/nistnet/

[10] SCore. http://www.pccluster.org

[11] Alain J. Roy, Ian T. Foster, William Gropp, Nicholas T. Karonis, Volker Sander, and Brian R. Toonen, "MPICH-GQ: Quality-of-Service for Message Passing Programs," SC'00, IEEE, 2000.

[12] William Saphir, Rob Van der Wijngaart, Alex Woo, and Maurice Yarrow, "New Implementations and Results for the NAS Parallel Benchmarks 2," 8th SIAM Conference on Parallel Processing for Scientific Computing, March 1997.

[13] Shinji Sumimoto, Hiroshi Tezuka, Atsushi Hori, Hiroshi Harada, Toshiyuki Takahashi, and Yutaka Ishikawa, "High Performance Communication using a Commodity Network for Cluster Systems," The Ninth International Symposium on High Performance Distributed Computing (HPDC-9), IEEE, pp.139–146, August 2000.

[14] Toshiyuki Takahashi, Shinji Sumimoto, Atsushi Hori, Hiroshi Harada, and Yutaka Ishikawa, "PM2: High Performance Communication Middleware for Heterogeneous Network Environments," SC'00, IEEE, 2000.

[15] Atsuko Takefusa, Satoshi Matsuoka, Hirotaka Ogawa, Hidemoto Nakada, Hiromitsu Takagi, Mitsuhisa Sato, Satoshi Sekiguchi, and Umpei Nagashima, "Multi-client LAN/WAN Performance Analysis of Ninf: a High-Performance Global Computing," SC'97, IEEE, 1997.

[16] Maurice Yarrow and Rob Van der Wijngaart, "Communication Improvement for the LU NAS Parallel Benchmark: A Model for Efficient Parallel Relaxation Schemes," Tech. Rep. NAS-97-032, NASA Ames Research Center, November 1997.