

MPICH-GP: A Private-IP-enabled MPI over Grid Environments

Kumrye Park¹, Sungyong Park^{1*}, Ohyoung Kwon², and Hyoungwoo Park³

¹ Dept. of Computer Science, Sogang University, Seoul, Korea
{namul, parksy}@sogang.ac.kr

² Korea University of Technology and Education, Chonan, Korea

³ Korea Institute of Science and Technology Information, Daejeon, Korea

Abstract. This paper presents an overview of MPICH-GP, which extends the MPICH-G2 functionality to support private IP clusters. To support the communication among private IP clusters, the MPICH-GP uses a communication relay scheme combining the NAT and a user-level proxy. The benchmarking results show that MPICH-GP outperforms the user-level two-proxy scheme used in PACX-MPI and Firewall-enabled MPICH-G, and also show comparable application performance to that of original MPICH-G2, especially in large message (or problem) size.

1 Introduction

As cluster systems become more widely available, it becomes feasible to run parallel applications across multiple private clusters at different geographic locations as a Grid environment. In the MPICH-G2 library [1], an implementation of the Message Passing Interface standard over Globus-based Grid environment, it is impossible for any two nodes located in different private clusters to communicate with each other directly across the public network.

In PACX-MPI [2], another implementation of MPI aiming to support the coupling of high performance computing systems distributed in a Grid, the communications among multiple private IP clusters are handled by *two user-level* daemons that allow the library to bundle communications and avoid having thousands of open connections between systems. However, since these daemons are implemented as proxies running in user space, the total bandwidth is only about half of the bandwidth obtained from kernel-level solutions [3]. It also suffers from higher latency due to the additional overhead of TCP/IP stack traversal and switching between kernel and user mode.

This paper presents the design and implementation of MPICH-GP, which is a private-IP-enabled MPI solution over Grid environments. To support private IP clusters, the MPICH-GP uses a communication relay scheme combining the NAT service with a user level proxy. In this approach, only incoming messages are handled by a user-level proxy to relay them into proper nodes inside the cluster, while the

* Corresponding author

outgoing messages are handled by the NAT service at the front-end node of the cluster. We have benchmarked our message relay scheme and compared it with the user-level two-proxy scheme used in PACX-MPI [2] and Firewall-enabled MPICH-G [4]. The performance results show that our NAT-based scheme outperforms the user-level two-proxy scheme. We have also benchmarked the performance of NAS Parallel Benchmark suite over MPICH-GP and compared it with those of MPICH-G2 and PACX-MPI. The results indicate that the overhead incurred by using a user-level proxy is minimal, especially in large message (problem) size, and the performance is at least better than that of PACX-MPI.

The rest of the paper is organized as follows. Section 2 presents an overview of the MPICH-GP architecture. The experimental results of MPICH-GP are presented in section 3. Section 4 concludes the paper.

2 Overview of MPICH-GP Architecture

In this section we present the design and implementation issues of the MPICH-GP architecture such as how the proxy process is designed including the communication relay scheme and protocol conversion, and the global rank management scheme used in MPICH-GP.

2.1 NAT-Based Communication Relay Scheme

In private IP clusters where each node within the cluster has a private IP address and thereby cannot directly communicate with public networks, a proxy that forwards incoming and outgoing messages is needed. The proxy process, in general, can be implemented either within the kernel or as a user-level process. Although the kernel-level proxy approach has the best performance, it is not widely used due to its poor portability. The user-level proxy scheme [2][4] is easy to implement but has the performance overhead such as those incurred by the TCP/IP stack traversal and the context switching between the kernel and the user mode. All packets sent from one node to the other nodes located in other clusters have to go through the user-level proxy twice, which decreases the performance further.

In MPICH-GP, only incoming messages are handled by a user-level proxy to relay them into proper nodes inside the cluster, while the outgoing messages are handled by the NAT service at the front-end node of the cluster. This brings performance improvement against the user-level two-proxy scheme since all packets pass through the user-level proxy once. By using the NAT service, which is generally provided by traditional operating systems, we could easily apply our scheme to MPICH-GP and implement a user-level proxy without modifying operating system kernel.

2.2 User-level Proxy Daemon

In MPICH-GP, a user-level proxy daemon is running at the front-end node. This proxy daemon accepts requests from one end of the MPI process and forwards the

requests to the other MPI process. Since the protocol between *MPI_Send* and *MPI_Recv* is stateful (i.e., Each MPI process maintains states such as 'await_instruction', 'await_format', 'await_header', 'await_data' and etc.), the proxy daemon is developed with a stateful server approach, where the daemon keeps the same states with the MPI processes (See Fig. 1). To maintain the source-level consistency with Globus, we designed the user-level proxy daemon with Globus I/O library [5], where we heavily used the callback mechanism for the communications among processes.

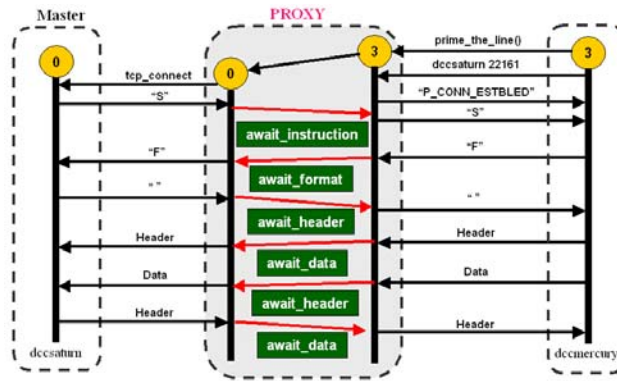


Fig. 1 User-level Proxy Daemon in MPICH-GP

2.3 Locating the Destination

When the *MPI_Send* operation is invoked in MPICH-GP, it first has to decide if the destination node is within the same cluster or outside the cluster. This allows us to determine if we need to send the data directly to the destination (if the destination is within the same cluster) or send the data via the proxy (if the destination is outside of the cluster). To decide the location of the destination, we have defined and added one proprietary field called GP_GUID into the channel data structure in Globus.

The GP_GUID structure contains the name or IP address of the front-end node where the compute node is located, the name or IP address of the compute node, etc. If the compute nodes are connected to the public network, the name of the front-end node is the same as that of the compute node. The name or IP address of the front-end node is initially set with an environment variable and is given to each MPI process. When each MPI process starts execution with *MPI_Init* function, it obtains the information from the environment variable and builds the GP_GUID structure. With the information in GP_GUID structure, any MPI process can decide whether we can directly connect to the destination node or via the proxy.

3. Experimental Results

In this section we present two benchmarking results to evaluate the performance of

MPICH-GP. The first benchmarking is to check the overhead of the user-level proxy and to compare the performance with that of the user-level two-proxy scheme used in PACX-MPI. The performance of MPICH-GP is also evaluated via applications and is compared with that of PACX-MPI. For the application benchmarking, we use the NAS Parallel Benchmark suite [6]. We only report the IS benchmark in this paper.

3.1 Forwarding Performance

The goal of the measurements presented here is a comparison of two different approaches, our NAT-proxy scheme and two-proxy scheme. Fig. 2 shows the latency between two private IP clusters. The latency was measured via *ping-pong* program using small sized messages (i.e., 128 bytes). As we can see from Fig. 2, the NAT-proxy scheme shows large performance improvement over two-proxy approach by about 144%. For example, the measured latency using NAT + proxy was 1923 *usec*, while the latency using two user-level proxies was 2756 *usec*. It is clear from the result that the overhead incurred by using NAT was much lower than that of using two user-level proxies.

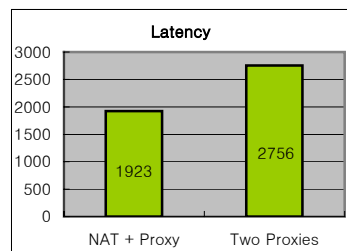


Fig. 2 Latency Between Two Private IP Clusters

3.2 Application Performance

To evaluate the latency impact on applications, the NAS Parallel Benchmark suite (NPB 3.1) [6] was run both on MPICH-GP and MPICH-G2. We have selected three benchmarks, IS, CG and LU. However, we only report the performance results of IS benchmark in this paper. By varying the problem size of the benchmarks from class S to class B ($S < W < A < B$), we measured the latency. The number of processors was fixed to 4.

Figure 3 shows the experimental results of IS benchmark. The latency means the total execution time of the benchmark. The line with a tag GP/G2 indicates the ratio of the performance, MPICH-GP/MPICH-G2 (that is the overhead of MPICH-GP compared to that of MPICH-G2). As we can see from Figure 3 (a), the latency increases as we increase the message size and the overhead of using user-level proxy keeps decreasing. This means that the overhead of using proxy gets amortized as we increase the message size. Figure 3 (b) compares the latency between MPICH-GP, MPICH-G2 and PACX-MPI using IS benchmark. For small messages, PACX-MPI shows the worst performance, but as we increase the message size, PACX-MPI shows

the best performance among them. It should be noted that PACX-MPI compresses the messages by default before sending and it can achieve better performance as we increase the message size.

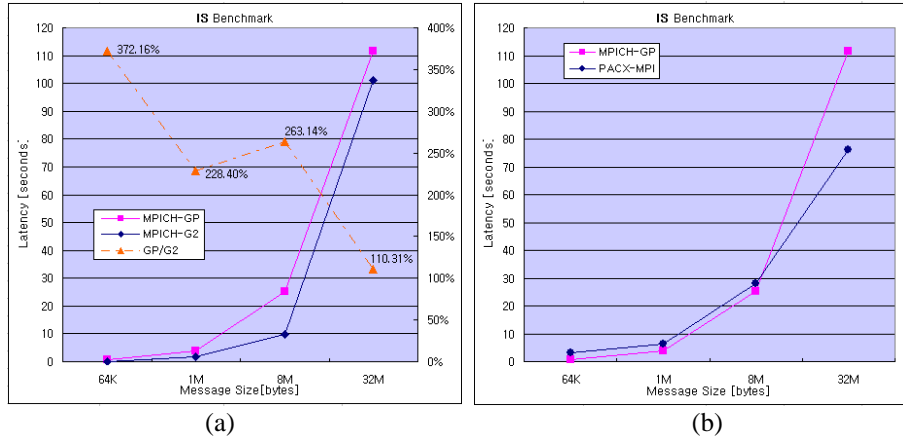


Fig. 3 IS Benchmark

4. Conclusions

In this paper we have presented the implementation issues of MPICH-GP and evaluated our implementation. We have also proposed a communication relay scheme based on the NAT and a user-level proxy, and compared our scheme with that of two user-level proxy scheme. From the forwarding bandwidth experiments, we showed that the performance of our scheme was better than that of two user-level proxy scheme. The application performance also indicated that the overhead of using proxy is minimal, especially in large message size.

References

1. Karonis, N.T., Toonen, B., Foster, I.: MPICH-G2: A Grid-Enabled Implementation of the Message Passing Interface, ANL/MCS Technical Reports P942-0402, (2002).
2. Gabriel, E., Resch, M., Beisel, T., Keller, R.: Distributed computing in a heterogeneous computing environment, LNCS 1497, pp.180-188, (1998).
3. Müller, M., Hess, M., Gabriel, E.: Grid enabled MPI solutions for Clusters, CCGRID'03, pp.18-24, (2003).
4. Tanaka, Y., Sata, M., Hirano, M., Nakata, H., Sekiguchi, S.: Performance Evaluation of a Firewall-compliant Globus-based Wide-area Cluster System, HPDC 2000, pp.121-128, (2000).
5. Ian Foster, Carl Kesselman eds, Grid: Blueprint for a Future Computing Infrastructure, Morgan Kaufmann Publishers, (1999).
6. NAS Parallel Benchmarks, <http://www.nas.nasa.gov/Software/NPB>