

Communication Strategies for Private-IP-Enabled Interoperable Message Passing across Grid Environments

Vijay Velusamy*, Purushotham Bangalore,
Anthony Skjellum
Department of Computer & Information Sciences
University of Alabama at Birmingham
Birmingham, AL – 35209
{velusamv, puri, tony}@cis.uab.edu

Pirabhu Raman
Verari Systems, Inc.
110 12th Street North, Suite D103
Birmingham, AL - 35203
pirabhu.raman@verari.com

Abstract

Grid computing enables deployment of applications across heterogeneous clusters belonging to multiple administrative domains. Majority of the applications deployed on the grid utilize MPI for communication and grid-enabled versions of MPI enable such application deployment. This paper presents a novel approach for deploying MPI across multiple clusters using the IMPI standard and Network Address Translation mechanism. In addition to supporting clusters with public IP addresses assigned to all compute nodes, this approach supports interoperable message passing across clusters with private IP addresses assigned to compute nodes (head nodes are behind firewalls and have public IP address). Preliminary results demonstrate that this approach is feasible and does not require any changes to the applications or firewall settings. Furthermore, this approach imposes minimal overhead to intra-cluster communication. The uniqueness of this approach is the use of the IMPI standard (interoperability between multiple MPI implementations) and support for Private-IP-enabled cluster environments.

1. Introduction

The virtualization, aggregation, and ubiquitous access to computational resources provided by grid computing have resulted in porting of many large-scale scientific and engineering applications to grid computing environments. The Message Passing Interface (MPI) [1, 2] remains one of the dominant paradigms for distributed memory systems. Although MPI can be used in any parallel computing system, it is becoming increasingly popular in cluster computing environments. Most cluster computing systems of

today are built from commodity systems employing fast inter-connects. Although cluster computing systems are comparatively easy to manage, since they are controlled by a single administrative domain, the limited availability of clusters to solve larger applications instigated the development of the computational grids, in order to harness the power of multiple distributed cluster computing domains as a unified computing resource.

Many massively parallel computing systems have been combined to demonstrate the ability of grid computing environments to solve large scientific applications. However, there exist many unique properties of clusters, especially since they belong to individual administrative domains. This imposes severe restrictions on the scalability and performance of grid computing environments. This paper discusses the issues related to combining multiple clusters in a grid computing environment.

The paper is organized as follows: Section 2 discusses existing message passing environments for the grid, and some of the related issues. Section 3 discusses some of the communication strategies and issues applicable to grid environments, followed by a possible solution in Section 4. Some initial performance measurements are provided in Section 5, followed by summary and future work.

2. Message Passing Environments for the Grid

This section describes some of the existing message passing environments for grid computing, such as MPICH-G2 [3], and PACX-MPI [4, 5].

2.1. MPICH-G2

MPICH-G2 is the grid-enabled version of the widely popular MPICH implementation of MPI (version 1.1

*Corresponding author

standard) [6]. It is integrated with many of the Globus services [7] including job startup, authentication, data conversion, and file access. This is done using a new device called globus2. Existing parallel programs that have been developed for MPICH can be ported to the Globus computing infrastructure, by simply recompiling the program with the MPICH-G2 library.

MPICH-G2 converts all messages to TCP for inter-cluster messaging, and vendor-supplied MPI for intra-cluster messaging. All TCP communication is implemented as point-to-point communications, essentially setting up direct socket-connections between the pair of processes.

2.2. PACX-MPI

Parallel Computer eXtension MPI (PACX-MPI) [4, 5] is an implementation of the MPI standard, designed as a library residing between the user application and the local intra-machine MPI implementation. The PACX-MPI layer intercepts MPI calls made by the application and contacts the other sub-cluster if required during the call execution. Otherwise the underlying local MPI call is made, thus using the vendor's MPI implementation for all intra-cluster communication.

When the MPI call involves communication between clusters, the communication is forwarded via network using TCP/IP. However, the MPI processes do not exchange messages directly. Instead, two special nodes are reserved on each cluster for each communication direction (for incoming and outgoing). On each of these nodes, the daemon MPI process running takes care of the communication with the local nodes, compression and decompression of data for remote communication, and communication with the daemons of other clusters.

3. Cluster in Grid Environments

Grid Environments comprising of multiple clusters pose many issues which are mostly addressed by the commonly used Grid-enabled MPI implementations. In this section some of the issues associated with communication strategies that are commonly used, and a technique that addresses them are discussed.

3.1. Private IP Cluster

In the Private IP cluster approach, typically the clusters are comprised of individual computers, each possessing a private IP address, communicating directly with every other computer, as shown in Figure 1.

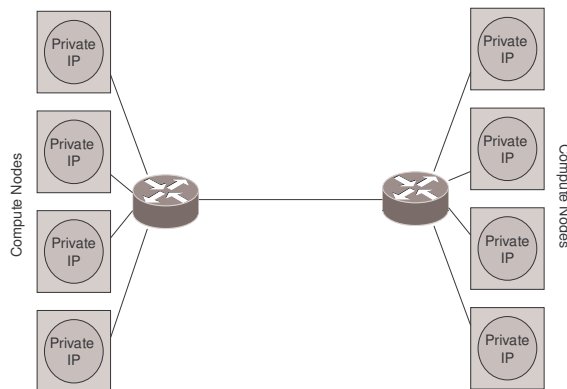


Figure 1. Private cluster approach

This, however, ignores the possibility for multiple administrative domains. Furthermore, this does not take into consideration the heterogeneity of the cluster environments into consideration. For example, it would be difficult to combine multiple clusters which employ different MPI implementations.

3.2. Public IP Clusters

Most MPI implementations that are designed for the grid, support communication across clusters only if all the cluster nodes have public IP addresses, as shown in Figure 2. Clusters may belong to different administrative domains, even though they all have public IP addresses, and every node in cluster 1 is able to communicate with every node in cluster 2. In this case, the Grid-enabled MPI implementations use the local MPI implementations for intra-cluster communication, and TCP/IP for inter-cluster messages.

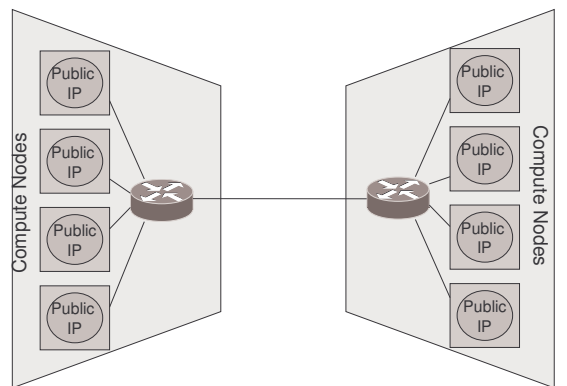


Figure 2. Nodes with Public IP addresses

This however introduces severe administrative problems. Large grid-node installations, typically

comprising of thousands of node require as many public IP addresses. Not only is obtaining so many public IP addresses (especially with current IPv4 limitations) an issue, but also configuring firewalls to support the port ranges in such an installations, and exposing the system to external attacks since they have public IP addresses.

3.3. Network Address Translation

In typical grid environments comprising of multiple administrative domains, the clusters have sets of private IP addresses. In this scenario, packets with private addresses are normally not routed outside the dedicated area. In a Local Area Network (LAN), since the packets will travel across switches and routers that are specifically configured to route these packets, it would be possible to combine two clusters. Network Address Translation (NAT) enables such a scenario. NAT is commonly used to masquerade internal networks by a public IP address [8].

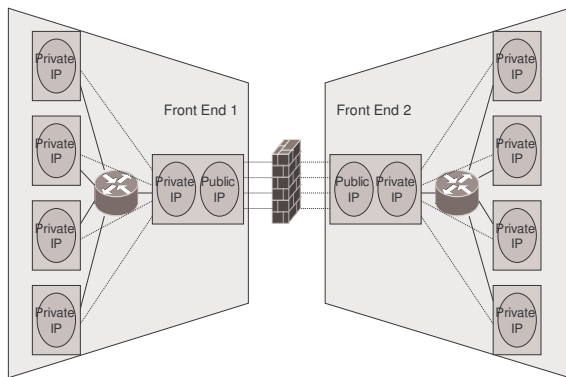


Figure 3. Network Address Translation

As shown in Figure 3, the front-end nodes are provided with Public IP addresses, whereas the nodes within each administrative domain are provided with Private IP addresses. The NAT implementation on Front End 1 will take requests to its public IP on a given port and transparently forward these requests to a specific machine/port in the private network. This will similarly work on Front End 2.

The following section discusses how NAT could be deployed for message passing in a grid computing environment, using Interoperable Message Passing Interface (IMPI).

4. NAT-based IMPI

MPICH-G2 provides a solution for communication between heterogeneous networks by using TCP for inter-cluster communication. Although MPICH-G2

has been demonstrated to use NAT [9], there are certain limitations posed by MPICH-G2. MPICH-G2 imposes restrictions on the local MPI implementation that can be used on the grid-nodes. MPICH-G2 supports only vendor supplied MPI implementations or MPICH based MPI implementations. The Interoperable Message Passing Interface (IMPI) Standard [10] developed by the National Institute of Standards and Technology defines a cross implementation protocol for MPI that enables heterogeneous computing. It enables a parallel message passing computation to be able to span across multiple systems that use native vendor message passing libraries on each system, without any modification to the application source. IMPI also has the advantage of being an actual standard that supports the interoperability of multiple MPI implementations.

Although inter-system communication is important, the main performance goal of the design is to not have any significant effect on the intra-system communication performance. In essence, as long as there is no inter-system communication, the performance of native intra-system communication should not be affected by the hooks added to the MPI implementations in order to support interoperability. It has been observed that MPICH-G2 has significant performance reduction even in the same administrative domain [11].

It is believed that IMPI coupled with NAT provides an ideal alternative for the grid environment in the influence of complex issues such as heterogeneity and administrative limitations. Figure 4 shows the architecture for the IMPI implementation coupled with NAT. Even though the figure shows two clusters participating, the idea can be easily extended to any number of clusters communicating through firewalls.

One of the main design features of IMPI to support multiple administrative domains is the start-up mechanism. In order to establish communication channels between the MPI processes running on the different systems, a single, implementation independent process, the IMPI server, is used as a rendezvous point. The IMPI server developed by Laboratory for Computer Science at the University of Notre Dame provides a portable IMPI server that be used for this purpose. The server simply relays all the information it receives all of the participating systems. The number of systems participating is the only information required by the server at start-up.

In Figure 4, the IMPI server appears to be running on a separate administrative domain, protected by a firewall from outside access. Although the IMPI server may also run on any one of the cluster head nodes, performance is not affected by having the IMPI server running separately, since it is only involved during the

startup of the various MPI processes, and not during the communication.

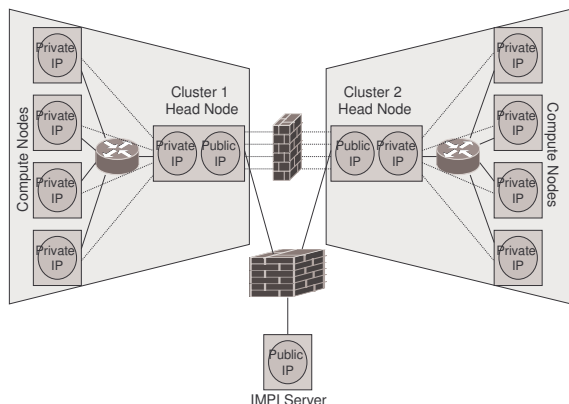


Figure 4. IMPI using NAT

Port numbers that are favorable for use in IMPI (range of ports allowed through the firewall), may be specified through environment variables (use `GLOBUS_TCP_PORT_RANGE` [12] as specified in Globus).

There are now potentially two strategies to use NAT with the IMPI based on the administrative privileges provided to MPI processes (with respect to NAT rule setting).

4.1. Restricted administrative privileges

Static network address translation rules are formulated and the rules are used by the root MPI process during startup.

4.2. Relaxed administrative privileges

If the MPI root processes are allowed to reset NAT rules, during startup, the root MPI process sets the NAT rules on Cluster 1 Head Node. The dynamic port information on cluster 1 is then sent to the IMPI server (as in our case). The IMPI enabled MPI implementation uses the port information to wait for connections from the other cluster(s) systems. The IMPI server relays the port information across to the other cluster(s) systems which then connect using this information. This however would require minor additions to the MPI implementation to setup the NAT rules during start-up and release them during shutdown.

5. Experiments

The following is a brief description of the initial experimentation with IMPI for grid environments. MPI/Pro [13] a commercial IMPI enabled

implementation of the MPI 1.2 standard was used for our test bed. First the Latency and Bandwidth of MPI/Pro were measured with and without IMPI. A cluster comprising of 4 Intel 2.4GHz Pentium 4's with 512K cache and 1GBRAM connected via a Gigabit Ethernet switch was used for this purpose.

A stock version of the IMPI server provided by Laboratory for Computer Science at the University of Notre Dame was used for purposes of these experiments. The applications were compiled with the new MPI/Pro libraries, without any modifications to the source code. The only modifications done for this demonstration was to the MPI/Pro library (as explained in Section 4.2). Additions were made to the start-up and shutdown procedure in the MPI/Pro library to setup the NAT rules on initiation, and release them during shutdown. Ports specified in the environment variable `GLOBUS_TCP_PORT_RANGE` were used to enable NAT, since these ports were already setup for access through the firewall in order to enable Globus. This avoids any unnecessary disturbance of the firewall settings.

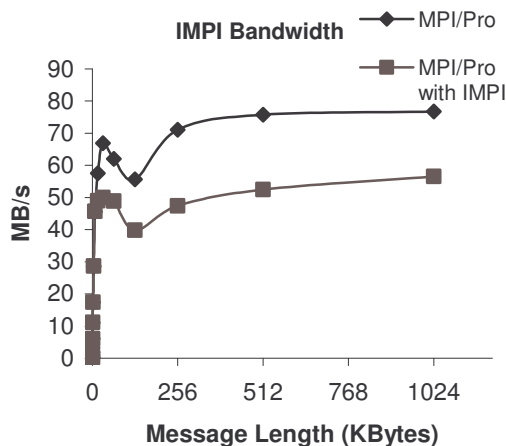


Figure 5. IMPI Bandwidth

5.1. IMPI Latency

Table 1 shows the latency measured between two machines in the cluster, both without using IMPI and with using IMPI. An overhead of 4.9us is observed in this case.

Table 1. MPI/Pro Latency

	Latency (us)
MPI/Pro without IMPI	142.45
MPI/Pro with IMPI	147.35

5.2. IMPI Bandwidth

Figure 5 shows the bandwidth measured for MPI for the same testbed. Peak bandwidths of upto 56.57MB/s and 76.78MB/s were achieved with and without IMPI respectively.

5.3. IMPI using NAT

For the following experiments, the above cluster was coupled with a cluster comprising of 10 dual Intel 2.4GHZ systems with 512 KB cache and 1GB RAM connected via a 100MB/s switch. The head nodes of the two clusters were also connected via the campus network.

The latency and bandwidth of using IMPI across the two clusters was measured. The testbed was setup as shown in Figure 4. One head node on each cluster had both a private, as well as public IP address, allowing the head nodes to communicate across the campus network using their public IP addresses. One MPI process was made to execute on a compute node that only had a private IP address on each of the clusters. So the latency and bandwidth were related to the two MPI processes communicating through NAT setup at the head node of Cluster 1. Even though this experiment uses MPI/Pro on both clusters, any MPI implementation that supports IMPI may be used.

Latency of 460.17us was observed. Figure 6 shows the bandwidth, and streaming bandwidth for message sizes varying from 0 to 1024 Kbytes. Peak Streaming Bandwidth of up to 8.64MB/s was observed.

6. Summary

This paper has provided a unique approach to deploying MPI across multiple clusters in a grid environment. This approach uses the IMPI standard to address inter-cluster communication when cluster nodes have private IP addresses and the head nodes are behind firewalls. Initial studies of using IMPI coupled with Network Address Translation have demonstrated the feasibility of using such strategies. Preliminary results indicate that IMPI introduces minimal overhead for intra-cluster communication. The performance of inter-cluster communication of this approach requires further improvements. To the best of the authors' knowledge this study is the first investigation of using the IMPI standard over Private-IP-enabled clusters. Further work is required to evaluate the performance impacts of this approach compared with other traditional grid-enabled message

passing systems. Future work would involve the incorporation of Grid services into the IMPI framework.

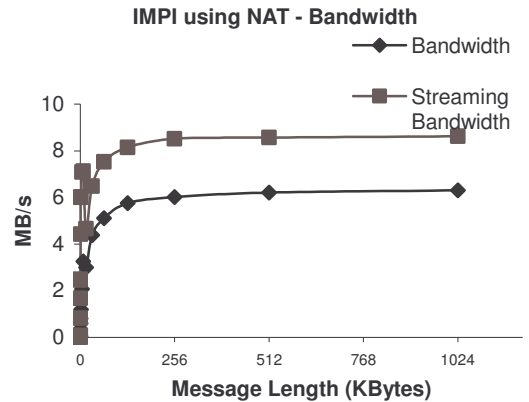


Figure 6 IMPI using NAT – Bandwidth

References

- [1] M. P. I. Forum, "MPI2: A Message-Passing Interface Standard," *International Journal of Supercomputer Applications and High Performance Computing (Special Issue)*, vol. 12, pp. 1-299, 1998.
- [2] M. P. I. Forum, "MPI: A Message-Passing Interface Standard," vol. UT-CS-94-230, 1994.
- [3] N. Karonis, B. Toonen, and I. Foster, "MPICH-G2: A Grid-Enabled Implementation of the Message Passing Interface," *Journal of Parallel and Distributed Computing*, 2003.
- [4] T. Beisel, E. Gabriel, and M. Resch, "An Extension to MPI for Distributed Computing on MPPs," presented at PVM/MPI, 1997.
- [5] E. Gabriel, M. Resch, T. Beisel, and R. Keller, "Distributed Computing in a Heterogeneous Computing Environment," presented at 5th European PVM/MPI Users' Group Meeting on Recent Advances in Parallel Virtual Machine and Message Passing Interface, 1998.
- [6] W. Gropp, E. Lusk, N. Doss, and A. Skjellum, "A High-Performance, Portable Implementation of the MPI Message Passing Interface Standard," *Parallel Computing*, vol. 22, pp. 789-828, 1996.
- [7] I. Foster and C. Kesselman, "The Globus toolkit," in *The Grid: Blueprint for a New Computing Infrastructure*, I. Foster and C. Kesselman, Eds. San Francisco, California: Morgan Kaufmann, 1999, pp. 259--78.
- [8] P. Srisuresh and K. Egevang, "Traditional IP Network Address Translator (Traditional NAT)," January 2001.
- [9] M. Müller, M. Hess, and E. Gabriel, "Grid enabled MPI solutions for Clusters," presented at 3rd International Symposium on Cluster Computing and the Grid, Tokyo, Japan, 2003.

- [10] 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, pp. 343-428, 2000.
- [11] R. Aleri and F. Spataro, "MPICH-G2 performance evaluation on PC clusters," February 2001.
- [12] O. Mulmo and V. Welch, "Using the Globus Toolkit with Firewalls," in *CLUSTERWORLD*, vol. 2, 2004, pp. 2-4.
- [13] R. Dimitrov and A. Skjellum, "Software Architecture and Performance Comparison of MPI/Pro and MPICH."