Making Wide-Area, Multi-Site MPI Feasible Using Xen VM

Masaki Tatezono[1], Naoya Maruyama[1], and Satoshi Matsuoka[1,2]

[1] Tokyo Institute of Technology, 2-12-1 Ohokayama, Meguro, Tokyo, 152–8552 Japan
{masaki.tatezono, naoya.maruyama, matsu}@is.titech.ac.jp
[2] National Institute of Informatics, 2-2-1 Hitotsubashi, Chiyoda, Tokyo, 101–8430 Japan

Abstract. Although multi-site MPI execution has been criticized in the past as being "impractical" due to limitations in network latency and bandwidth, we believe many of the obstacles can be overcome by various means for wider classes of applications than previously believed. One such technique is transparent dynamic migration of MPI, coupled with aggressively performance-oriented overlay networks, assuming availability of gigabits of bandwidth on future WANs. The problem, of course, is to investigate the exact implications to application performances given the arsenal of such techniques, but such work has been quite sparse. Our current work involves using Xen as the underlying virtual machine layer to implement such migration, along with performance-optimizing migration strategies—this particular paper deals with performance evaluations of MPI on Xen VMs including what are the possible performance hindrances, implications of migrations, as well as the effect of variations in latencies and bandwidth parameters as realized by the overlay network using a software network emulator.

Keywords: MPI, cluster computing, grid computing, virtualization

1 Introduction

Executing MPI on wide-area, multi-site clusters has been considered impractical. One of the reasons for this impracticality is heterogeneity of clusters. Using multiple clusters with different hardware and software is hard for typical applications users. Another reason is regarded as low performance of wide-area networks, imposing high communication overhead. Finally, the asymmetric networks caused by firewalls and NATs make multi-site MPI execution further difficult.

To achieve efficient multi-site MPI execution, we envision that the application user runs his programs on a *virtual cluster* that actually runs on multiple physical clusters connected with future high-speed wide-area networks. The virtual cluster hides heterogeneity of different system configuration using virtual machine monitors (VMMs) such as Xen [1]. It also hides the network asymmetry using overlay networks as a virtual network substrate. Moreover, it can exploit

dynamic migration of VMs to use available resources more efficiently. For example, when a cluster of faster CPUs and larger memory becomes available for use, an already-running virtual cluster can be migrated to the cluster for better performance.

There exists some past work that attempts to combine VMMs with overlay networks for multi-site MPI execution, such as Violin [2]. However, its evaluation of MPI performance has been done only with a single program, HPL, in a single site. Also, as far as we know, there is no performance evaluation on multi-site, wide area virtual clusters. Thus, the performance of more diverse set of applications, running on wide-area overlay networks, is still unknown and remains to be evaluated.

The primary purpose of this paper is a study of performance of MPI on a virtual cluster using Xen as a VMM and various overlay network configurations. First, we compare performance of NAS Parallel Benchmarks (NPB) on Xen-based virtual cluster with the native cluster with the same hardware configuration, consisting of up to 128 physical nodes. Next, we investigate the effect of overlay networks to application performance using different overlay network implementations.

These experimental studies show that the performance of Xen for MPI execution platform is comparable to native machines in terms of pure CPU performance. The average performance degradation of virtual clusters was less than 20%. We also show that doubling the number of nodes from 32 to 64 by adding another cluster cannot lead to speedups due to the bottleneck inter-cluster link. Based on these results, we discuss how to mitigate such network bottlenecks and the requirements to achieve high-performance MPI execution on virtual clusters.

2 Virtualization of Compute Hosts

We describe requirements on virtualization of compute hosts. First, the performance overhead should be minimal. Second, it must support light-weight dynamic VM migration. Third, it should not impose particular constraints to user-level programs; In other words, off-the-shelf MPI implementations and its applications should work with little adaptation.

With the recent advance in virtualization techniques, we see these requirements are achievable using available VMMs, such as Xen and VMWare ESX Server [3]. For example, Xen is reported to migrate a VM within 60ms on a LAN environment [4], with negligible CPU performance overhead [1]. Thus, we choose to use existing implementations of VMMs. We describe our current instance of this approach using Xen in Section 4.

3 Overlay Networks

We describe key requirements on underlying overlay networks where virtual compute hosts are hosted, and our approach to them.

3.1 Requirements

We require the following four criteria to be satisfied.

Providing Constant IP Addresses To support migration of MPI processes, hosts' IP addresses need to be constant. The reason for this is that typical MPI implementations do not allow IP addresses to change at run time. We see this complexity should be handled by the overlay network layer, instead of the virtual cluster user.

Low Overhead Since we intend virtual clusters as MPI execution platforms, the overhead needs to be kept minimal.

Security To operate across WANs we need some level of security. At a minimum, we require overlay networks to have standard secure authentication mechanisms, such as SSL or SSH. Furthermore, since some user may require encryption of messages in communication links, it is preferable to be able to support message encryption.

Reachability of Compute Nodes Typical MPI, such as MPICH [5], make a connection for each pair of participating processes, thus requiring that each process is able to connect to every other process. Although this requirement is not an issue on LAN clusters, such implementations cannot operate across firewalls and NATs. Even so-called "grid-enabled" MPI implementations, such as GridMPI [6] and MPICH-G2 [7], require each process to be globally accessible. Therefore, overlay networks have to provide reachability across firewalls and NATs even for compute nodes.

3.2 Approach

To achieve the requirements, we employ *site-to-site VPNs* as overlay networks. As depicted in Figure 1, in a site-to-site VPN, the physical gateway for each site establishes a secure connection to every other gateway. This achieves security by requiring authentication and possibly encryption of messages. By assuming the gateways are globally addressable with no firewalls, it provides reachability of compute nodes across firewalls and NATs.

The expected performance of this approach is superior to other alternatives, such as *client-to-server VPNs*, where each participating node connects to a single VPN server. Since packets traverse the additional layer only when crossing a site boundary, the expected performance overhead is less than that of the client-to-server VPN approach. A downside of this approach is that since it is the gateway of each LAN that makes a tunneling connection to every other gateway, it forces even underlying physical networks to join the virtual network, resulting in a significant change in the original network administration. We expect this is not necessarily a limiting barrier in HPC cluster environments.

4 Components of the Prototype Virtual Cluster

To evaluate the feasibility and performance of virtual clusters, we construct prototype virtual cluster environments using existing VMM and VPN implementations that satisfy our requirements.

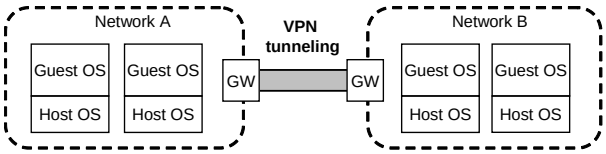


Fig. 1. Site-to-site VPN

As a VMM, we choose Xen for its performance and support of dynamic VM migration (a.k.a., live migration [4]). It optimizes virtualization performance by para-virtualization, and is reported that the downtime during VM migration is as low as 60ms in a LAN environment [4].

Among existing implementations of overlay networks, we use OpenVPN, an open-source VPN implementation [8], and PacketiX, a commercial product by SoftEther Corporation [9], as sample implementations. Both implementations provide a software-emulated virtual Ethernet over the standard TCP/IP network. They support secure authentication and message encryption via SSL with X.509 certificates. Providing constant IP addresses is also doable by configuring a single-subnet site-to-site VPN. Although PacketiX employs performance optimization mechanisms, including parallel connections and automatic tuning of its number, its performance for HPC applications is still unknown.

5 Experimental Studies

To evaluate performance implication of virtual clusters, we conduct several experiments using our prototype virtual cluster. First, to examine the baseline performance of Xen-based virtual clusters , we compare performance of MPI applications on virtual compute nodes with native compute nodes. Note that to observe the overhead caused by host virtualization, we do not used VPNs for this experiment. Second, we evaluate the performance of virtual clusters on multi-site environments. In this study, we do not deploy the virtual clusters on real multi-site environments; rather we use software-emulated two-site clusters.

5.1 Experimental Setups

Our experimental platform consists of an x86 cluster, called PrestoIII cluster, a pair of VPN tunneling machines for each of OpenVPN and PacketiX, and a NIST Net network emulator.

PrestoIII: Base Evaluation Platform PrestoIII cluster consists of 256 compute hosts of dual AMD Opteron 242 1.6GHz with 2GB of RAM, running Linux kernel v2.6.12.6. Each node has a gigabit Ethernet interface, connected to a 24-port gigabit switch of Dell PowerConnect 5224. As depicted in Figure 2, the entire cluster nodes are interconnected with twelve of the 24-port switches and

an 8-port gigabit switch of 3Com SuperStack3 3848. Each of the Dell PowerConnect 5224 hosts 20 compute hosts, and is further connected to the 3Com switch with four 1Gbps uplinks.

Virtual Machines Each of the PrestoIII cluster nodes hosts a single VM (a.k.a., DomU), using Xen v3.0.2 on Linux kernel v2.6.16 with the Xen patch applied. Each VM and its host OS (a.k.a., Dom0) are assigned 512MB and 128MB of RAM, respectively.

Overlay Network Testbeds Three kinds of overlay network testbeds are used: OpenVPN and PacketiX VPNs, and latency-inserted emulated two-site environments, as shown in Figure 3. To emulate a wide-area link, a software-implemented network emulator called NIST Net [10] is used. It allows to insert configurable amount of delay into a standard Linux packet router.

OpenVPN site-to-site VPN Two 32-node clusters are created using PrestoIII compute nodes, each of which is interconnected with a 48-port gigabit switch. The switch is then connected to a VPN tunneling gateway implemented with OpenVPN v2.0.7. Its encryption and compression options are disabled. Each gateway runs on a node with the same configuration as the compute nodes of PrestoIII cluster, but with an additional gigabit NIC.

PacketiX site-to-site VPN The PacketiX site-to-site VPN testbed is organized in the same way as the OpenVPN site-to-site testbed, except for the gateway machines. A pair of Windows XP machines with PacketiX v2.0 is used. They run on Intel Pentium4 662 3.6GHz with 1GB RAM, and two gigabit NICs. PacketiX's encryption and compression options are disabled.

Emulated Two-Site Clusters using NIST Net This testbed differs from the other testbeds only on the inter-cluster link. The two clusters are interconnected by a linux router with NIST Net version 2.0.12b. It runs on AMD Athlon MP 2000+ 1.6GHz with 1GB of RAM, running Linux kernel v.2.4.31. It has two gigabit NICs, each of which is connected to one of the clusters via a gigabit switch.

Network and MPI Benchmark Programs To evaluate latencies and bandwidths, NetPIPE v3.6.2 is used. For the studies of MPI performance, NPB v.3.1 with MPICH 1.2.7p1 is used.

5.2 Network Benchmark Results on Xen VMs

To evaluate the performance of network I/O on Xen VMs, we compare the latency and bandwidth on Xen guest OSes with those on native and Xen host OSes. On two of the PrestoIII compute nodes, we used NetPIPE's ping-pong test, which bounces messages of increasing size between two processes.

The latencies on the Xen guest, Xen host, native OSes were 0.08ms, 0.05ms, and 0.04ms, respectively. The bandwidths on the Xen guest, Xen host, native

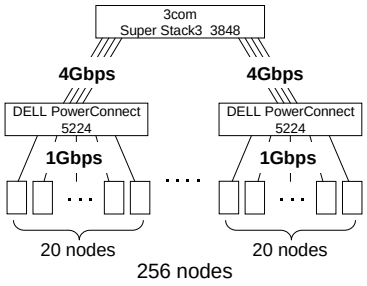


Fig. 2. PrestoIII Networking

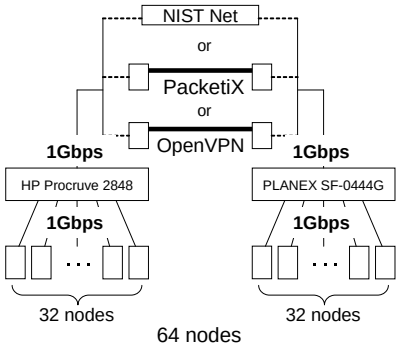


Fig. 3. Overlay Network Testbeds

OSes were 430Mbps, 883Mbps, and 896Mbps, respectively. Thus, the latency on the Xen guest OS was about twice as long as the native OS, while the bandwidth on the guest OS was about half of the native OS.

5.3 MPI Benchmark Results on Xen VMs

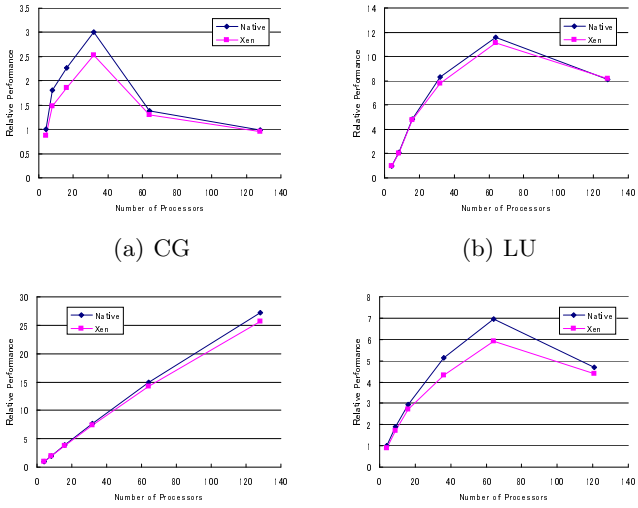
Figure 4 shows the performance of NPB CG, LU, EP, and BT class B on a cluster of Xen VMs and another cluster of native machines. The x-axes represent the number of processors used, while the y-axes the relative performance to the experiment using four processors. The performance metric is MOPS (i.e., mega operations per second).

Each graph in Figure 4 shows that the difference between the native and VMs is small: less than 20%, and nearly 0% in the case of EP and LU.

5.4 Network Benchmark Results on the Overlay Network Testbeds

Table 1 shows latencies and bandwidths of the site-to-site VPN and emulated two-site testbeds. We conducted the ping-pong tests over the OpenVPN and PacketiX gateway machines as well as the NIST Net router. The graph named "Native" in each figure shows the performance when no gateway or route is interposed between the two switches.

As shown in Table 1, the latency overhead by OpenVPN and PacketiX was 0.11ms and 0.32ms, respectively; The bandwidth of OpenVPN was decreased by a factor of three to five. To identify the reason of the low bandwidth, we measured the CPU usage of the OpenVPN gateway machines during the NetPIPE experiments. We see that on each gateway, the CPU usage was always close to 100%. We predict that the bandwidths of the VPN gateways are bound by CPU performance. Further analysis of the bottleneck remains to be conducted.



(a) CG (b) LU

(c) EP (d) BT

Fig. 4. The relative performance of NPB class B benchmarks.

Table 1. Latencies and bandwidth of the overlay networks.

	Latency (ms)	Bandwidth (Mbps)
Native	0.04	896
OpenVPN	0.15	290
PacketiX	0.36	170
0ms	0.12	700
0.1ms	0.28	636
0.5ms	0.65	392

5.5 MPI Benchmark Results on the Overlay Network Testbeds

Figure 5 shows the results of four MPI benchmarks on the OpenVPN and PacketiX site-to-site VPN testbeds as well as emulated two-site testbeds. We configure the NIST Net emulator to impose delays of 0ms, 2.5ms, and 5.0ms. The 0ms-delay network gives the baseline performance using two clusters excluding the effect of VPN overhead. We also present the performance when 64 nodes in PrestoIII are used without the overlay networks to show the ideal performance of the 64-node cluster. The heights of the bars represent the relative performance of each configuration against the configuration using 32 native machines in a single cluster. The performance metric is MOPS. Note that the Y value greater than 1 means that its configuration achieves speedup by using 64 multi-site nodes against 32 nodes.

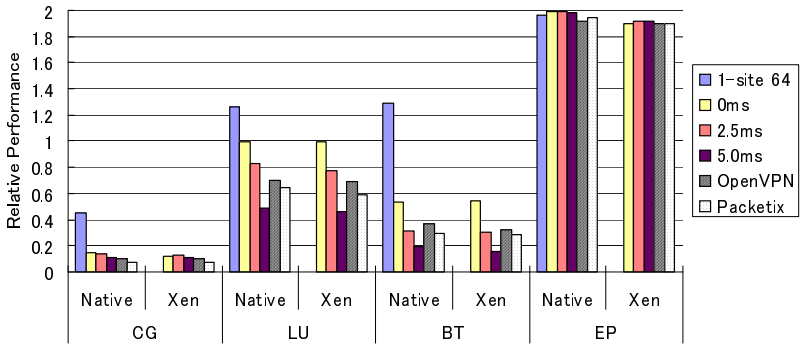


Fig. 5. Performance of the NPB benchmarks on the overlay networks.

On these results, we point out three remarks. First, as expected, the performance of NPB EP is mostly irrelevant to underlying networks, and achieves a linear speedup using two 32-node clusters. Second, we see that the native machines and the Xen VMs exhibit almost the same performance patterns with respect to the variations in underlying networks. Third, the achieved speedups using two 32-node clusters are less than 1 for most cases except for EP, while those using a 64-node cluster are greater than 1.2 in both LU and BT. Thus, in these configurations, using two 32-node clusters does not make the performance better than a single 32-node cluster. On native machines, using OpenVPN degrades the performance of CG, LU, and BT, to 22%, 54%, and 28%, of the single 64-node cluster, respectively; the 0ms-delay network does to 31%, 79%, and 41%. These results suggest that the most significant source of overhead of site-to-site VPNs comes from the inter-cluster link, not the VPN gateways.

6 Discussion and Future Directions

6.1 Performance Implications of using VMs for MPI

The experimental results of MPI on Xen-based VMs using a physical single cluster shows that virtualization overhead ranges from 0% to 20% on 4–128-node clusters. Within four benchmarks, CG, which is the most communication intensive, exhibits the largest overhead. On the other hard, EP, which is mostly compute-intensive, achieves nearly the same performance as that on native machines. Therefore, we see that major source of overhead in MPI execution comes from its communication.

While some users would find 0–20% degradation unacceptable, we see that the advantage of VMs would outweigh for other users. One of such advantages is isolation of system environments. For example, virtual clusters allows the user to

customize environments without modifying underlying resources. Such isolation is likely to be useful in resource-sharing environments such as computing centers.

6.2 Overlay Networks for MPI Execution

To improve the site-to-site VPN performance, we explore three directions. First, we will investigate optimization of collective operations. As discussed in Section 5.5, wide-area links are the most significant bottleneck in site-to-site VPNs. Thus, MPI performance could be improved by optimizing its collective operations for multi-site execution, as proposed by Kielmann et al. [11].

Second direction is to schedule wide-area communication based on load imbalance of parallel programs. Although load balancing is one of the research areas that have been received the most attention, the problem still exists in a wide variety of scientific programs. Our idea is to give a higher priority to the processes that are predicted to reach a barrier point later than the other processes. The VPN tunneling machine, in turn, could prioritize the messages to and from those higher-priority processes. We expect this priority-based scheduling to hide latencies of WANs to some extent.

Third, we will explore more advanced alternatives such as parallel implementations that use available multiple processors and nodes. Although the main bottleneck lies in wide-area links, we still expect there is a chance to improve the MPI performance by increasing the performance of VPN gateways.

7 Related Work

Virtual Workspace [12] deploys VMs by extending Workspace Service of Global Toolkit 4. Unlike our proposed virtual cluster vision, it does not do dynamic VM migration. Besides, in Virtual Workspace, no mechanism is provided for the reachability across NATs and firewalls. VioCluster [13] uses UML to virtualize compute nodes, and VIOLIN [2] as underlying overlay networks. With experimentation using HPL on a single-site VioCluster, they estimated the performance overhead of VioCluster is at most 15%. They have not evaluated performance of multi-site MPI executions.

There has been some work on MPI that extends the standard single-site MPI execution model to multi-site execution, such as MPICH-G2 [7], GridMPI [6], and MagPIe [11]. Although none of the past work can achieve the potential optimization through the VM migration, their optimized collective operations for wide-area links would be beneficial in our virtual cluster environments.

8 Conclusion

We have discussed the motivation and requirements of virtual clusters, and presented our current approach to them. It consists of Xen for virtualization of compute nodes, and site-to-site VPNs for overlay networks. The experimental

studies suggest that, while the overhead caused by Xen VMs is relatively small, the underlying network can significantly degrade application performance. Finally, we discussed the possible strategies to mitigate the performance problem.

Our future work includes more elaborate performance studies as well as those discussed in Section 6. In addition, based on the detailed performance studies, we will explore the possibility of constructing performance models for MPI on virtual clusters. With the performance models, we plan to study global scheduling algorithms for virtual clusters on grids.

Acknowledgments

This work is supported in part by Japan Science and Technology Agency as a CREST research program entitled "Mega-Scale Computing Based on Low-Power Technology and Workload Modeling", and in part by the Ministry of Education, Culture, Sports, Science, and Technology, Grant-in-Aid for Scientific Research on Priority Areas, 18049028, 2006.

References

1. Barham, P., Dragovic, B., Fraser, K., Hand, S., Harris, T., Ho, A., Neugebauery, R., Pratt, I., Warfield, A.: Xen and the art of virtualization. In: SOSP, Bolton Landing, New York (2003)
2. Jiang, X., Xu, D.: Violin: Virtual internetworking on overlay infrastructure. In: Department of Computer Sciences Technical Report CSD TR 03-027, Purdue University (2003)
3. VMWare: Esx server architecture and performance implications. White Paper (2005)
4. Clark, C., Fraser, K., Hand, S., Hanseny, J.G., July, E., Limpach, C., Pratt, I., Warfield, A.: Live migration of virtual machines. In: NSDI. (2005)
5. Gropp, W., Lusk, E., Doss, N., Skjellum, A.: A high-performance, portable implementation of the MPI message passing interface standard. Parallel Computing **22**(6) (1996) 789–828
6. GridMPI: Gridmpi project. `http://www.gridmpi.org/` (2006)
7. Karonis, N.T., Toonen, B., Foster, I.: MPICH-G2: A grid-enabled implementation of the message passing interface. Journal of Parallel and Distributed Computing (JPDC) **63**(5) (2003) 551–563
8. OpenVPN Solutions LLC: Openvpn. `http://openvpn.net/` (2006)
9. SoftEther Corp.: Packetix. `http://www.softether.comn/` (2006)
10. Carson, M., Santay, D.: NIST Net: A linux-based network emulation tool. SIGCOMM Comput. Commun. Rev. **33**(3) (2003) 111—126
11. Kielmann, T., Hofman, R.F.H., Bal, H.E., Plaat, A., Bhoedjang, R.A.F.: Magpie: Mpi's collective communication operations for clustered wide area systems. In: PPoPP, Atlanta, GA (1999) 131–140
12. Foster, I., Freeman, T., Keahey, K., Cheftner, D., Sotomayor, B., Zhang, X.: Virtual clusters for grid communities. In: CCGRID. (2006) 513–520
13. Ruth, P., McGachey, P., Xu, D.: Viocluster: Virtualization for dynamic computational domains. In: IEEE International Conference on Cluster Computing, Boston, MA (2005)