Virtual Organization Clusters

Michael A. Murphy, Michael Fenn, and Sebastien Goasguen
School of Computing
Clemson University
Clemson, South Carolina 29634-0974 USA
{*mamurph, mfenn, sebgoa*}*@cs.clemson.edu*

Abstract

Sharing traditional clusters based on multiprogramming systems among different Virtual Organizations (VOs) can lead to complex situations resulting from the differing software requirements of each VO. This complexity could be eliminated if each cluster computing system supported only a single VO, thereby permitting the VO to customize the operating system and software selection available on its private cluster. While dedicating entire physical clusters on the Grid to single VOs is not practical in terms of cost and scale, an equivalent separation of VOs may be accomplished by deploying clusters of Virtual Machines (VMs) in a manner that gives each VO its own virtual cluster. Such Virtual Organization Clusters (VOCs) can have numerous benefits, including isolation of VOs from one another, independence of each VOC from the underlying hardware, allocation of physical resources on a per-VO basis, and clear separation of administrative responsibilities between the physical fabric provider and the VO itself.

Initial results of implementing a complete system utilizing the proposed Virtual Organization Cluster Model confirm the administrative simplicity of isolating VO software from the physical system. End-user computational jobs submitted through the Grid are executed only on the virtual cluster supporting the respective VO, and each VO has substantial administrative flexibility in terms of software choice and system configuration. Performance tests using the Kernel-based Virtual Machine (KVM) hypervisor indicated a virtualization overhead of under 10% for latency-tolerant scientific applications, such as those that would be submitted to a standard or vanilla Condor universe. Latency-sensitive applications, such as MPI, experience substantial performance degradation with virtualization overheads on the order of 60%. These results suggest that VOCs are suitable for High-Throughput Computing (HTC) applications, where real-time network performance is not critical. VOCs might also be useful for High-Performance Computing (HPC) applications if virtual network performance can be sufficiently improved.

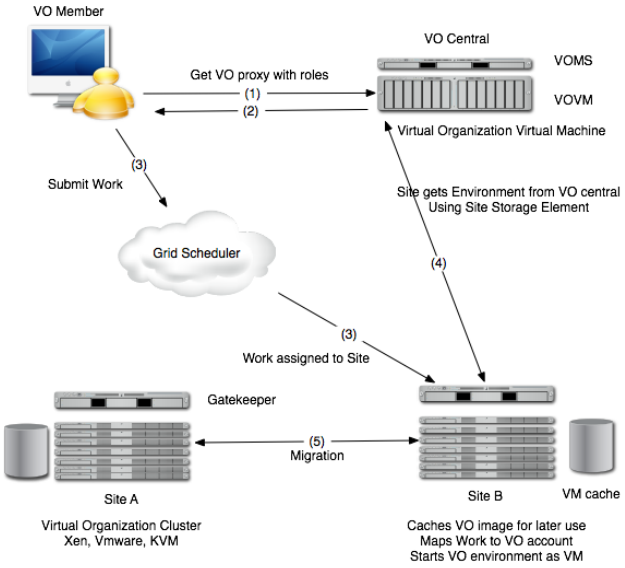

Figure 1. Grid-Based Use Case for a VOC

1. Introduction

Virtual Organizations (VOs) enable scientists to collaborate using diverse, geographically distributed computing resources. Requirements and membership of these VOs often change dynamically, with objectives and computing resource needs changing over time [1]. Given the diverse nature of VOs, as well as the challenges involved in providing suitable computing environments to each VO, Virtual Machines (VMs) are a promising abstraction mechanism for providing grid computing services [2]. Such VMs may be migrated from system to system to effect load balancing and system maintenance tasks [3]. Cluster computing systems, including services and middleware that can take advantage of several available hypervisors [4], have already been constructed inside VMs [5]–[7]. However, a cohesive view of virtual clusters for grid-based VOs has not been presented to date. The purpose of this paper is to bridge this gap by presenting a Virtual Organization Cluster Model.

Implementing computational clusters with traditional multiprogramming systems may result in complex systems that

require different software sets for different users. Each user is also limited to the software selection chosen by a single system administrative entity, which may be different from the organization sponsoring the user. Virtualization provides a mechanism by which each user entity might be given its own computational environment. Such virtualized environments would permit greater end-user customization at the expense of some computational overhead. [2]

The primary motivation for the work described here is to enable a scalable, easy-to-maintain system on which each Virtual Organization can deploy its own customized environment. Such environments will be scheduled to execute on the physical fabric, thereby permitting each VO to schedule jobs on its own private cluster.

Figure 1 presents a use case of using Virtual Organization Clusters (VOCs) in a manner based on the operating principles of the Open Science Grid (http://www.opensciencegrid.org) – a grid infrastructure known to support VOs instead of individual users. In the figure, "VO Central" is a database run by the VO manager. It contains a list of members and their associated privileges stored in the Virtual Organization Manager Service (VOMS), and a set of computing environments (in the form of virtual machine images) stored in the "Virtual Organization Virtual Machine (VOVM)." When a VO member wants to send work to the grid, a security proxy is obtained from her VOMS server, and the work is submitted to a VO meta-scheduler (casually depicted as a cloud in this figure). Once work is assigned to a site, this site downloads the proper VM either from the VOVM or from its own VM cache. These data transfers can be done through the OSG data-transfer mechanisms (i.e. Phedex and dCache) and can use the GridFTP protocol. If a site becomes full, work can be migrated to another site using VM migration mechanisms. This use case represents an ideal form of grid operation, which would provide a homogeneous computing environment to the users.

The remainder of this paper is organized as follows: Section 2 presents related work and describes how the VOCM fits into the larger research picture. Section 3 describes the model in detail, providing high-level descriptions of both the VOC and the supporting physical fabric. A test implementation of a system designed according to the model is presented in Section 4, followed by performance validation results in Section 5. Finally, conclusions and future work are presented in Section 6.

2. Related Work

Constructing virtual clusters for use in virtual grid computing has required addressing the issues of installing and provisioning virtual machines. Middleware has been developed to facilitate construction of virtual machine clusters. Several middleware-oriented projects have been undertaken, including In-VIGO [5], [7], [8], VMPlants [6], DVC [9], virtual disk caching [10], and the Globus Virtual Workspace [11], [12].

Each virtual cluster, no matter how constructed or realized, needs a physical cluster upon which to run. Several mechanisms have been developed for the rapid construction of physical computation clusters, including Rocks [13]–[15], OSCAR [16], and the Cluster-On-Demand (COD) system [17]. These systems facilitate the rapid installation and configuration of physical-level clusters that share resources via traditional multiprogramming. In particular, Rocks provides a mechanism for easy additions of software application groups via "rolls," or meta-packages of related programs and libraries [18]. The OSCAR meta-package system also permits related groups of packages to be installed onto a physical cluster system [16].

Several networking libraries have been developed for virtual machines, which permit virtual clusters to use networks logically isolated from the underlying physical hardware. Both Virtual Distributed Ethernet (VDE) [19] and Virtuoso [20] provide low-level virtualized networks that can be utilized for interconnecting VMs. Furthermore, wide-area connectivity of VMs can be achieved through the use of tools such as Wide-area Overlays of virtual Workstations (WOW) [21] and Violin [22]. Live migration of VMs between physical nodes [3] recently has been shown to be possible over wide-area networks such as the Internet [23].

Unlike prior cluster computing and virtualization research, the cluster virtualization model described in this paper focuses on customizing environments for individual VOs instead of individual physical sites. Since *a priori* knowledge of a particular VO's scientific computing requirements is not always available, this model makes few assumptions about the operating environment desired by each individual VO. As a result, the focus of the physical system configuration is to support VMs with minimal overhead and maximal ease of administration. Moreover, the system should be capable of supporting both high-throughput and high-performance distributed computing needs on a per-VO basis, imposing network performance requirements to support MPI and similar packages.

3. Cluster Virtualization Model

The Virtual Organization Cluster Model specifies the high-level properties of systems that support the assignment of computational jobs to virtual clusters owned by single VOs. Central to this model is a fundamental division of responsibility between the administration of the physical computing resources and the virtual machine(s) implementing each VOC. For clarity, the responsibilities of the hardware owners are said to belong to the Physical Administrative Domain (PAD). Responsibilities delegated to the VOC owners are part of the Virtual Administrative

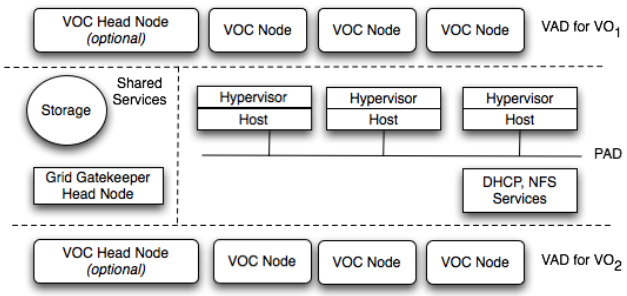

Figure 2. PAD and VAD

Domain (VAD) of the associated VOC. Each physical cluster has exactly one PAD and zero or more associated VADs.

Figure 2 illustrates an example system designed using the VOC model. In this example, the PAD contains all the physical fabric needed to host VOCs and connect them to the Grid. Each physical compute host in the PAD is equipped with a hypervisor for running VOC nodes. Shared services, including storage space, a Grid gatekeeper, and networking services are also provided in the PAD. Two VOCs are illustrated in figure 2, each having its own, independent VAD. Each VOC optionally could include a virtual head node, which would receive incoming Grid computational jobs from the shared gatekeeper in the PAD. Alternatively, each VOC node could receive jobs directly from the shared gatekeeper, by means of a compatible scheduler interface.

In practice, Virtual Organization Clusters can be supplied by the same entity that owns the physical computational resource, by the Virtual Organizations (VOs) themselves, or by a contracted third party. Similarly, physical fabric on which to run the VOCs could be provided either by the VO or by a third party.

3.1. Physical Administrative Domain

The Physical Administrative Domain (PAD) contains the physical computer hardware (see figure 2), which comprises the host computers themselves, the physical network interconnecting those hosts, local and distributed storage for virtual machine images, power distribution systems, cooling, and all other infrastructure required to construct a cluster from hardware. Also within this domain are the host operating systems, virtual machine hypervisors, and central physical-level management systems and servers. Fundamentally, the hardware cluster provides the hypervisors needed to host the VOC system images as guests.

An efficient physical cluster implementation requires some mechanism for creating multiple compute nodes from a single VO-submitted image file. One solution is to employ a hypervisor with the ability to spawn multiple virtual machine instances from a single image file in a read-only mode that does not persist VM run-time changes to the image file. Another solution would be to use a distributed file copy mechanism to replicate local copies of each VM image to each execution host. Without this type of mechanism, the VO would be required to submit one VM image for each compute node, which would result in both higher levels of Wide Area Network traffic and greater administration difficulty.

3.2. Virtual Administrative Domain

Each Virtual Administrative Domain (VAD) consists of a set of virtual machine images for a single Virtual Organization (VO). A VM image set contains one or more virtual machine images, depending upon the target physical system(s) on which the VOC system will execute. In the general case, two virtual machine images are required: one for the head node of the VOC, and one that will be used to spawn all the compute nodes of the VOC. When physical resources provide a shared head node, only a compute node image with a compatible job scheduler interface is required.

VMs configured for use in VOCs may be accessed by the broader Grid in one of two ways. If the physical fabric at a site is configured to support both virtual head nodes and virtual compute nodes, then the virtual head node for the VOC may function as a gatekeeper between the VOC and the Grid, using a shared physical Grid gatekeeper interface as a proxy. In the simpler case, the single VM image used to construct the VOC needs to be configured with a scheduler interface compatible with the physical site. The physical fabric will provide the gatekeeper between the Grid and the VOC (figure 2), and jobs will be matched to the individual VOC.

3.3. Provisioning and Execution of Virtual Machines

Virtual Organization Clusters are configured and started on the physical compute fabric by middleware installed in the Physical Administrative Domain. Such middleware can either receive a pre-configured virtual machine image (or pair of images) or provision a Virtual Organization Cluster on the fly using an approach such as In-VIGO [5], VMPlants [6], or installation of nodes via virtual disk caches [10]. Middleware for creating VOCs can exist directly on the physical system, or it can be provided by another (perhaps third-party) system. To satisfy VAD administrators who desire complete control over their systems, VM images can also be created manually and uploaded to the physical fabric with a grid data transfer mechanism such as the one depicted in the use case presented in figure 1.

Once the VM image is provided by the VO to the physical fabric provider, instances of the image can be started to form virtual compute nodes in the VOC. Since only one

VM image is used to spawn many virtual compute nodes, the image must be read-only. Run-time changes made to the image are stored in RAM or in temporary files on each physical compute node and are thus lost whenever the virtual compute node is stopped. Since changes to the image are non-persistent, VM instances started in this way can be safely terminated without regard to the machine state, since data corruption is not an issue. As an example, VM instances started with the KVM hypervisor are abstracted on the host system as standard Linux processes. These processes can be safely stopped (e.g. using the SIGKILL signal) instantly, eliminating the time required for proper operating system shutdown in the guest. Since there is no requirement to perform an orderly shutdown, no special termination procedure needs to be added to a cluster process scheduler to remove a VM from execution on a physical processor.

Once mechanisms are in place to lease physical resources and start VMs, entire virtual clusters can be started and stopped by the physical system. VOCs can thus be scheduled on the hardware following a cluster model: each VOC would simply be a job to be executed by the physical cluster system. Once a VOC is running, jobs arriving for that VOC can be dispatched to the VOC. The size of each VOC could be dynamically expanded or reduced according to job requirements and physical scheduling policy. Multiple VOCs could share the same hardware using mechanisms similar to sharing hardware among different jobs on a regular physical-level cluster.

4. Initial Test Implementation

An initial cluster implementation was performed to test the Virtual Organization Cluster Model. This section presents in detail the procedure that was followed to set up the physical cluster, configure the physical fabric to support virtualization, and to construct the VOC itself. The physical test cluster was based upon the Kernel-based Virtual Machine (KVM) hypervisor, which was installed on physical hosts running Slackware Linux 12. A Virtual Organization Cluster was constructed around a single virtual machine image into which CentOS 5.1 had been installed. In this particular implementation, the head node for the VOC was provided as part of the physical fabric, even though it was actually implemented inside a virtual machine. This head node was connected to the Open Science Grid Integration Testbed.

4.1. Physical Cluster Construction

The hardware cluster for the test installation consisted of sixteen nodes: fifteen Dell PowerEdge 860 1U rackmount systems, and one Dell PowerEdge 2970 2U rackmount server. One PowerEdge 860 system was employed to host the VOC head node, while the other fourteen were each prepared to host two VOC virtual compute nodes. Each PowerEdge 860 machine used in this test was configured with a 2.66 GHz dual-core Intel Xeon CPU, 4 GiB of RAM, and an 80 GB hard disk drive. The 2U PowerEdge 2970 server was employed to host installation images, user home directories, network services, and a shared VM image store exported via a Network File System server. Prior benchmarks and considerable network test results [24] were obtained during a prior CentOS implementation using the same hardware.

To provide hypervisor services, the Kernel-based Virtual Machine (KVM) was installed on each compute node and on the physical head node. KVM was chosen primarily due to its compatibility with the most recent kernel release at cluster construction time, as the most recent drivers were needed for optimal performance of certain hardware components.

Network access was provided to each virtual machine by means of bridging the physical Ethernet card in each physical compute node to both the physical node itself and each guest machine (two guests per host). Thus, three logical devices shared each physical device. MAC addresses were assigned to each VM instance by KVM, using a custom script to generate the MAC addresses deterministically based on the host machine. On the physical head node, two separate bridges were employed: one to the cluster's private LAN, the other to the University network and public Internet. Network Address Translation and iptables firewalls were implemented on both the physical head node and utility system, allowing them to serve as edge routers for the entire private LAN. Since each VM obtained an IP address from the DHCP server, and each IP address was part of the same subnetwork without regard to physical or virtual host status, each VM instance had both Internet access and local connectivity to other VMs in VOC.

In the test cluster, a common VOC head node was provided as part of the PAD. For administrative simplicity, this CentOS 5.1 node was implemented as a virtual machine that was bridged to the public Internet. To supply job scheduling, Condor 7.0.0 was installed on the shared VOC head node as well as on all compute nodes. Open Science Grid Integration Testbed membership was achieved by installing the OSG Virtual Data Toolkit (VDT) and connecting to OSG by configuring an OSG compute element. The compute element that ran Globus GRAM was set up as a shared head node for both the physical and virtual compute nodes. Differentiation between the PAD and VAD was done through the attributes advertised by each compute node's Condor *startd*. This setup, shown in figure 3, used the VOCM variant in which each VO shared the same head node.

4.2. Virtual Cluster Construction

Constructing the Virtual Organization Cluster for the test system was a straightforward task, since only 1 Virtual

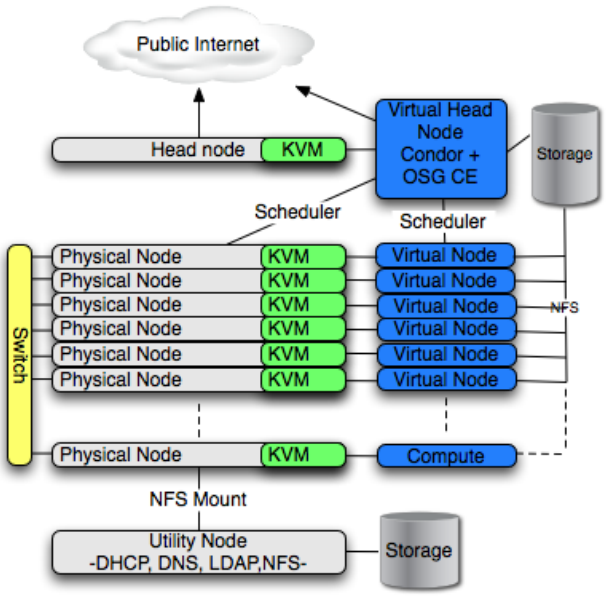

Figure 3. Initial Test Cluster Setup

Machine image was required to implement the whole VOC. CentOS 5.1 was installed into a VM image, then Condor was installed and configured to run a *startd* process to enable jobs to be scheduled. The VM image was configured for DHCP networking, and the primary assumptions made about the underlying Physical Administrative Domain were that jobs would arrive via the Condor scheduler and that the KVM hypervisor would be used to execute the VMs.

As a result of the hardware emulated by KVM, implicit low-level requirements were imposed upon the VOC system. In practice, these requirements were not substantial, since the Linux system used in the VOC was generic enough to support the emulated hardware. However, a different choice of guest operating system might have required additional configuration steps for the VO administrator. In particular, KVM could execute only 32-bit, x86-compatible operating systems.

5. System Performance Validation

Several performance tests were conducted to ensure that the Virtual Organization Cluster Model was viable. Viability of a VOC was defined as the ability both to start a VOC in a reasonable amount of time and to execute scientific applications with reasonable performance. In order to evaluate the viability of the test VOC, two major installations were performed: a Slackware Linux 12 installation directly on the physical hardware and a CentOS 5.1 installation into an operational Virtual Organization Cluster. Following the installations, boot times were measured for both the physical and virtual systems. A High Performance Linpack (HPL) benchmark was performed on the physical compute nodes, followed by a second HPL benchmark on the VOC. Several different process grid sizes were used in the benchmark tests. To determine the cause of observed poor performance with HPL on the operational VOC, a set of network tests was conducted. These tests included bandwidth measurement and ping Round-Trip Time (RTT) measurements to assess network latency.

To effect the performance tests, the High Performance Computing Challenge (HPCC) benchmark suite was used, which included HPL. Boot times were measured manually for the physical boot procedure, while a simple boot timing server was constructed to measure VM booting time. Network bandwidth was measured using both the Iperf bandwidth measurement tool and the RandomRing bandwidth assessment in the HPCC suite. Latency in network communications under load also was assessed using the RandomRing benchmark. Measurement of Round-Trip Time (RTT) of ICMP Echo packets generated by the UNIX ping tool was used as an additional measure of network latency both under load (with the HPCC suite running) and without computational load on the VOC.

5.1. System Performance

Following system installation, boot times were recorded for both the physical and virtual systems. Since VM startup was scripted, automated means were devised to measure the VM boot times. A simple server was deployed on the physical utility node, which received boot starting notifications from the physical nodes and boot complete notifications from the associated virtual nodes. Timing of the boot process was performed at the server side, avoiding any clock skew potentially present between physical and virtual nodes, but possibly adding variable network latency. Boot times for the physical nodes were subject to even greater variation, as these were measured manually.

Results of the boot time tests are summarized in table 1. For the physical system, the boot process was divided into three phases: a PXE timeout, a GRUB timeout, and the actual kernel boot procedure. While total boot times ranged from 160 to 163 seconds, 105 to 107 seconds of that time were utilized by the PXE timeout, and 10 seconds were attributed to the GRUB timeout. Thus, the actual kernel boot time ranged from 43 to 46 seconds. In contrast, the virtual compute nodes required 61.2 to 70.2 seconds to boot. These virtual machines were configured with a different operating system (CentOS 5.1) and started approximately 10 additional processes at boot time, compared to the physical systems. As a result, not all the boot time discrepancy could be attributed to virtualization overhead. Nonetheless, the overhead was small enough that booting the VOC did not require an inordinate amount of time. Thus, by the requirement that VOCs boot quickly, the test VOC was viable.

Table 1. Boot Times (seconds)

Physical Node				VM
Statistic	PXE Timeout	Total Boot	Actual Boot	VM Boot
Minimum	105	160	43	61.2
Median	106	160.5	44	65.4
Maximum	107	163	46	70.2
Average	106.4	160.9	44.5	65.5
Std Deviation	0.63	1.03	1.09	2.54

Table 2. Slackware 12 vs. CentOS 5.1

Process Grid (PxQ)	14x2
Problem Size	77,000
CentOS GFLOPS	115.6
Slackware GFLOPS	129.6
Performance Increase	12.11%

Table 3. Physical Cluster vs. VOC

Process Grid (PxQ)	1x1	7x2	7x4
Problem Size	10,300	38,700	54,800
Physical GFLOPS	7.29	74.39	143.45
VOC GFLOPS	6.57	29.74	63.09
Virtualization Overhead	9.86%	60.02%	56.02%

Following the boot procedures, HPL benchmark data were obtained for both the physical and operational VOC nodes (tables 2 and 3). First, an HPL benchmark previously conducted on the prior CentOS physical installation was performed on the Slackware hosts. A 12% performance increase was noted as a result of the Slackware installation. Although the cause of this increase could not be conclusively determined, it was believed that the customization of the installation – including Linux kernel optimization – and minimization of unnecessary services contributed to the additional performance. This result showed that keeping the metal configuration as lightweight and simple as possible not only made it easier for the cluster administrator to maintain but also increased the overall performance, thereby benefiting all VOs using the cluster.

HPL and HPCC tests were performed on both the physical machines and the VOC, using the same process grid layouts and problem sizes across administrative domains. As shown in table 3, the virtualization overhead in terms of HPL observed throughput was only 9.86% when no inter-node communication occurred. This type of HPL test on a single compute node simulated High-Throughput Computing (HTC) jobs, such as those that would be run in a "vanilla" Condor universe. MPI jobs that utilized inter-node communications incurred substantial performance overheads on the order of 56% to 60%. Network latency was suspected for this observed overhead, as latency has been implicated as a cause of performance reduction in prior studies involving MPI [25], [26]. With MPI applications comprising a significant fraction of all scientific computing endeavors, it was desirable to be able to deploy a VOC that had good MPI performance. The single node HPL performance showed that running single-processor Condor-based jobs was entirely viable and will only suffer a performance penalty on the order of 10%. In contrast, the test VOC was not viable for High-Performance Computing (HPC) applications using MPI. Additional network investigations were undertaken to evaluate the cause of this problem.

5.2. Network Performance

Several networking issues were suspected in the initial setup. Two VMs shared a single Linux TUN/TAP bridge to a single physical Gigabit Ethernet port, which was also shared by the host for host-level network connectivity. Each KVM instance also emulated an Intel 82540EM Gigabit Ethernet Network Interface Card (NIC), which was presented to the guest OS and utilized as if the card were an actual physical device. The physical NIC on the host was configured in promiscuous mode, bypassing the internal NIC packet filtering code and offloading the low-level network processing onto the host CPU. Furthermore, the bridge component of the kernel and NIC emulation components of KVM also relied upon the host CPU to effect communications. As a result, the host CPU was taxed not only with the computationally-intensive HPL routines, but also with low-level networking operations typically carried out in the NIC hardware.

Table 4 summarizes the results of cluster network testing using Iperf, ping, and the Random-Ring bandwidth and latency benchmarks in HPCC. Iperf showed 941 $\mathrm{Mb} \cdot \mathrm{s}^{-1}$ of available bandwidth when the cluster was not under load, decreasing to 882 $\mathrm{Mb} \cdot \mathrm{s}^{-1}$ when HPL benchmarks were running on the hosted VOC. This decrease could be attributed to inter-node MPI communications, which would have consumed a portion of the network resources. The decrease in measured bandwidth between VMs was more significant, dropping from 708 $\mathrm{Mb} \cdot \mathrm{s}^{-1}$ to 636 $\mathrm{Mb} \cdot \mathrm{s}^{-1}$ for communications between VMs hosted by different physical nodes. Communications between two VMs sharing the same bridge were found to have substantially lower available bandwidth, with only 499 $\mathrm{Mb} \cdot \mathrm{s}^{-1}$ (roughly half the nominal bandwidth of Gigabit Ethernet) available when not under load. During the HPL tests, this intra-bridge bandwidth fell to an available level of 206 $\mathrm{Mb} \cdot \mathrm{s}^{-1}$. Bandwidth as

Table 4. Bandwidth and Ping Latency

Condition	No-Load			Under Load		
Parameter	P	SB	BTB	P	SB	BTB
Iperf ($\mathrm{Mb} \cdot \mathrm{s}^{-1}$)	941	499	708	882	206	636
RRB ($\mathrm{Mb} \cdot \mathrm{s}^{-1}$)	N/A			544	24	32
Ping RTT (μs)	106	215	312	191	360	484
RRL (μs)	N/A			54	379	233

Key: P – Physical, SB – Virtual links across the Same Bridge, BTB – Virtual links from one bridge (physical host) to another, RRB – RandomRing Bandwidth, RRL – RandomRing Latency

measured under load by the Random-Ring benchmarking was substantially lower in all cases: 544 $\mathrm{Mb} \cdot \mathrm{s}^{-1}$ for the physical hosts, and 24 $\mathrm{Mb} \cdot \mathrm{s}^{-1}$ to 32 $\mathrm{Mb} \cdot \mathrm{s}^{-1}$ for the VMs. Lower bandwidth was observed when the MPI rings included intra-bridge links (SB column of the table) than when only inter-bridge links (links between VMs hosted by different physical systems) were included in the MPI rings. Unlike the Iperf tests, the Random-Ring test data for bandwidth across intra-bridge links is also averaged with the available bandwidth between bridges; without this averaging, it is likely that the intra-bridge links would have shown lower available bandwidth, based upon the Iperf tests.

Latency between nodes was found to be higher between virtual hosts than between the underlying physical hosts. Measuring the Round-Trip Time (RTT) of the ping (ICMP echo) operation yielded an average of 106 μs without load, increasing to 191 μs under load. Ping operations across a single bridge (intra-bridge) required longer times to execute: 215 μs in the absence of load, increasing to 360 μs under load. RTTs for ping operations between VMs on different hosts were the longest, beginning at 312 μs and increasing to 484 μs under load, suggesting that inter-bridge communications incurred the greatest latency. However, the Random-Ring benchmarks indicated greater latency between VMs sharing a single bridge, with bridge-to-bridge latencies 146 μs lower at 233 μs. Both VM latency figures were an order of magnitude higher than the measured 54 μs latency on the physical network.

One significant limitation of the network architecture used for the first implementation was identified as a result of the test procedures. Two VMs and one physical host were configured to share one physical Ethernet NIC on each physical node. Thus, parallel communication between two pairs of VMs on two separate physical hosts would have been converted to sequential networking operations, with packet queuing needed either at the bridges or at the physical switch. Queuing, in turn, could have introduced added latency into the communications, which may have reduced MPI performance. Moreover, an increase in queuing could have increased packet transmission time, thus causing the TCP protocol used by MPI to place more packets in flight to fill the sliding sender window. Such an increase in packet saturation on the network used in the test cluster has been shown to increase queuing delays, thereby increasing latency and further aggravating communications difficulties [24].

The combination of virtual machine overhead, latency introduced by the bridged networking, and delay properties of the underlying physical network resulted in a network environment that could not support MPI or other latency-sensitive applications inside VOCs. Latency in the underlying physical network was already on the order of 50 μs for one-way unicast traffic. VOC traffic latency was greatly increased as a result of the addition of the emulated NIC, the use of the Linux bridge facility, and the reassignment of low-level network processing from the physical NIC to the host CPU. The unsatisfactory performance results obtained from this experiment indicated that an alternative mechanism for providing network connectivity to VMs would be needed if VOCs were to support HPC jobs.

6. Conclusions

The Virtual Organization Cluster Model suggests a system architecture in which each Virtual Organization (VO) is given a dedicated computational cluster for which it has complete administrative access and isolation from other VOs. As demonstrated by the constructed physical cluster and Virtual Organization Cluster (VOC), virtualization provides a practical mechanism by which Grid-connected systems can be designed according to the model. As demonstrated by the system performance tests, VOCs can experience a performance loss of under ten percent when the underlying physical services are minimized. However, performance losses are unacceptably high when latency-sensitive applications such as MPI are executed, owing to limitations observed in the networking layer. Additional research is needed to determine if the latency observed with the current implementation can be reduced to a level sufficient for the execution of MPI jobs.

As demonstrated by the test implementation, the Virtual Organization Cluster Model provides a methodology by which Virtual Organizations may have customized cluster computing environments. Such environment customization will enable 21st-century scientists to have complete discretion over the scientific software packages used in their research endeavors.

Acknowledgment

This material is based upon work supported under a National Science Foundation Graduate Research Fellowship.

References

[1] I. Foster, C. Kesselman, and S. Tuecke, "The anatomy of the Grid: Enabling scalable virtual organizations," *International*

Journal of Supercomputing Applications, vol. 15, no. 3, pp. 200–222, 2001.

[2] R. J. Figueiredo, P. A. Dinda, and J. A. B. Fortes, "A case for grid computing on virtual machines," in *Proceedings of the 23rd International Conference on Distributed Computing Systems*, 2003.

[3] C. Clark, K. Fraser, S. Hand, J. G. Hansen, E. Jul, C. Limpach, I. Pratt, and A. Warfield, "Live migration of virtual machines," in *Proceedings of the 2nd ACM/USENIX Symposium on Networked Systems Design and Implementation*, Boston, MA, May 2005, pp. 273–286.

[4] B. Quetier, V. Neri, and F. Cappello, "Selecting a virtualization system for Grid/P2P large scale emulation," in *Proceedings of the Workshop on Experimental Grid Testbeds for the Assessment of Large-scale Distributed Applications and Tools (EXPGRID'06)*, Paris, France, June 2006.

[5] S. Adabala, V. Chadha, P. Chawla, R. Figueiredo, J. Fortes, I. Krsul, A. Matsunaga, M. Tsugawa, J. Zhang, M. Zhao, L. Zhu, and X. Zhu, "From virtualized resources to virtual computing grids: the In-VIGO system," *Future Generation Computer Systems*, vol. 21, no. 6, pp. 896–909, June 2005.

[6] I. Krsul, A. Ganguly, J. Zhang, J. A. B. Fortes, and R. J. Figueiredo, "VMPlants: Providing and managing virtual machine execution environments for grid computing," in *Proceedings of the 2004 ACM/IEEE Conference on Supercomputing*, 2004.

[7] A. Matsunaga, M. Tsugawa, M. Zhao, L. Zhu, V. Sanjeepan, S. Adabala, R. Figueiredo, H. Lam, and J. A. Fortes, "On the use of virtualization and service technologies to enable grid-computing," in *11th International Euro-Par Conference*, August 2005.

[8] J. A. B. Fortes, R. J. Figueiredo, and M. S. Lundstrom, "Virtual computing infrastructures for nanoelectronics simulation," *Proceedings of the IEEE*, vol. 93, no. 10, pp. 1839–1847, October 2005.

[9] W. Emeneker and D. Stanzione, "Dynamic virtual clustering," in *IEEE Cluster 2007*, Austin, TX, September 2007.

[10] H. Nishimura, N. Maruyama, and S. Matsuoka, "Virtual clusters on the fly - fast, scalable, and flexible installation," in *CCGRID 2007: Seventh IEEE International Symposium on Cluster Computing and the Grid*, May 2007.

[11] I. Foster, T. Freeman, K. Keahey, D. Scheftner, B. Sotomayor, and X. Zhang, "Virtual clusters for grid communities," in *CCGrid 2006*, Singapore, May 2006.

[12] K. Keahey, I. Foster, T. Freeman, X. Zhang, and D. Galron, "Virtual workspaces in the Grid," in *11th International Euro-Par Conference*, Lisbon, Portugal, September 2005.

[13] M. J. Katz, P. M. Papadopoulos, and G. Bruno, "Leveraging standard core technologies to programmatically build Linux cluster appliances," in *Cluster 2002: IEEE International Conference on Cluster Computing*, April 2002.

[14] P. M. Papadopoulos, M. J. Katz, and G. Bruno, "NPACI Rocks: Tools and techniques for easily deploying manageable Linux clusters," in *Cluster 2001: IEEE International Conference on Cluster Computing*, October 2001.

[15] P. M. Papadopoulos, C. A. Papadopoulos, M. J. Katz, W. J. Link, and G. Bruno, "Configuring large high-performance clusters at lightspeed: A case study," in *Clusters and Computational Grids for Scientific Computing 2002*, December 2002.

[16] J. Mugler, T. Naughton, and S. L. Scott, "OSCAR metapackage system," in *19th International Symposium on High Performance Computing Systems and Applications*, May 2005.

[17] J. S. Chase, D. E. Irwin, L. E. Grit, J. D. Moore, and S. E. Sprenkle, "Dynamic virtual clusters in a grid site manager," in *HPDC '03: Proceedings of the 12th IEEE International Symposium on High Performance Distributed Computing*, June 2003.

[18] G. Bruno, M. J. Katz, F. D. Sacerdoti, and P. M. Papadopoulos, "Rolls: Modifying a standard system installer to support user-customizable cluster frontend appliances," in *IEEE International Conference on Cluster Computing*, September 2004.

[19] R. Davoli, "VDE: Virtual Distributed Ethernet," in *First International Conference on Testbeds and Research Infrastructures for the Development of Networks and Communities (Tridentcom 2005)*, Trento, Italy, February 2005.

[20] A. I. Sundararaj and P. A. Dinda, "Towards virtual networks for virtual machine grid computing," in *Proceedings of the Third Virtual Machine Research and Technology Symposium*, San Jose, CA, May 2004.

[21] D. Wolinsky, A. Agrawal, P. O. Boykin, J. Davis, A. Ganguly, V. Paramygin, P. Sheng, and R. Figueiredo, "On the design of virtual machine sandboxes for distributed computing in Wide-area Overlays of virtual Workstations," in *First International Workshop on Virtualization Technology in Distributed Computing*, 2006.

[22] P. Ruth, X. Jiang, D. Xu, and S. Goasguen, "Virtual distributed environments in a shared infrastructure," *Computer*, vol. 38, no. 5, pp. 63–69, 2005.

[23] E. Harney, S. Goasguen, J. Martin, M. Murphy, and M. Westall, "The efficacy of live virtual machine migrations over the Internet," in *Second International Workshop on Virtualization Technology in Distributed Computing*, Reno, NV, November 2007.

[24] M. A. Murphy and H. K. Harton, "Evaluation of local networking in a Lustre-enabled virtualization cluster," Clemson University, Tech. Rep. CU-CILAB-2007-1, 2007.

[25] M. Matsuda, T. Kudoh, and Y. Ishikawa, "Evaluation of MPI implementations on grid-connected clusters using an emulated WAN environment," in *IEEE International Symposium on Cluster Computing and the Grid (CCGrid03)*, 2003.

[26] P. Luszczek, D. Bailey, J. Dongarra, J. Kepner, R. Lucas, R. Rabenseifner, and D. Takahashi, "The HPC Challenge (HPCC) benchmark suite," in *Supercomputing '06*, 2006.