Virtualizing High Performance Computing

Joshua E. Simons
VMware, Inc.
Five Cambridge Center
Cambridge, Massachusetts 02142
simons@vmware.com

Jeffrey Buell
VMware, Inc.
3401 Hillview Ave.
Palo Alto, California 94304
jbuell@vmware.com

ABSTRACT
While virtualization is widely used in commercial enterprise environments, it has not to date played any significant role in High Performance Computing (HPC). However, with the rise of cloud computing and its promise of computing on demand, the HPC community's interest in virtualization (a key cloud enabler) is increasing. Beyond cloud computing, virtualization offers additional potential benefits for HPC, among them reactive and proactive application fault tolerance, secure and fault-isolated use of shared-resource clusters, dynamic provisioning, job migration, and support for heterogeneous HPC facilities. This paper describes both the promises and challenges in this new, emerging area, including a discussion of performance-related issues.

Categories and Subject Descriptors
K.6.4 [**Computing Milieux**]: Management of Computing and Information Systems—*System Management*; C.4 [**Computer Systems Organization**]: Performance of Systems

General Terms
Management, Performance, Reliability

Keywords
HPC, virtualization, resilience, cloud computing

1. INTRODUCTION
While virtualization is now widely deployed as an important component of enterprise IT environments and as the substrate on which cloud computing environments are being built, it has generally not been used for hosting High Performance Computing (HPC) workloads. This is due in part to a perception that virtualization, by definition, unacceptably degrades application performance, and also to a lack of information about the potential benefits virtualization might bring to an HPC environment so that the costs of virtualization can be judged against its benefits.

At VMware we have begun to explore both issues – the extent to which virtualization can adequately support HPC workloads and the values virtualization can deliver to HPC customers. Our intent is two-fold. First, to characterize those workloads we believe can be addressed with current virtualization technology. And, second, to identify the additional engineering work required to support a broader array of these workloads over time.

This paper presents an overview of our current thinking about these issues and is organized as follows. We begin in Section 2 by defining some basic terminology to set the context for the rest of the paper, followed in Section 3 by a description of emerging trends that establish a broader perspective from which to view the issue of virtualized HPC. We then turn in Section 4 to a description of the primary uses we have identified for virtualization in HPC based on our own experiences and through discussions with numerous customers and HPC users. In Section 5 we outline our preliminary measurement efforts and those of others and also discuss the future work needed to gain a more complete understanding of the performance landscape for virtualized HPC. We then end in Section 6 with a discussion of the major challenges we have identified that must be overcome to deliver the full value of virtualization to the HPC community.

2. DEFINITION OF TERMS
This paper is about virtualization for High Performance Computing (HPC). Both of these terms are defined below.

2.1 Virtualization
By "virtualization" we mean *server virtualization* – the hosting of one or more virtual machine (VM) instances on a single physical machine or larger numbers of VMs on a cluster of physical machines. Figure 1 illustrates the main components in this approach. This is in contrast to other virtualization approaches which attempt to unify aggregations of physical machines into a single, logical system either at the administrative level as with Penguin Computing's Scyld ClusterWare, or at the BIOS level as with ScaleMP's vSMP products. Primary examples of server virtualization include VMware vSphere, Citrix XenServer, Microsoft Hyper-V, and KVM-based approaches.

It is also important to note that virtualization encompasses more than the hypervisor – the thin software layer that actually controls a system's underlying hardware and offers

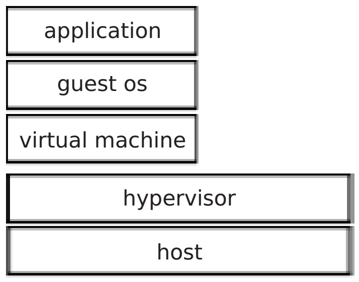

Figure 1: Server virtualization

an abstracted, virtual view of that hardware to guest operating system instances. While we will not discuss these capabilities in detail in this paper, dynamic resource management; system, network, and storage provisioning; security; and monitoring are all critical pieces of any virtual datacenter or virtual cluster deployment.

2.2 High Performance Computing

By "HPC" we refer to a broad range of applications that are often floating-point intensive, often storage intensive, and often technical or scientific. HPC applications are found across industry, academia, and government in a variety of disciplines including Life Sciences, Electronic Design Automation (EDA), Mechanical Computer-Aided Engineering (MCAE), Energy, and Homeland Security. HPC applications are also found in less traditional areas including Financial Services, Digital Content Creation (DCC), and, increasingly, in enterprises adopting compute-intensive approaches for analytics or business intelligence.

The compute, storage, and interconnect requirements for HPC applications span a wide range. The workload in some areas of HPC is throughput-oriented, running very large numbers of independent, serial application instances to compute a result (e.g., Monte Carlo simulation). In other areas, thousands to hundreds of thousands of inter-communicating processes run in parallel to cooperatively solve extremely large, grand challenge problems. In practice, the requirements of most HPC users fall within these two extremes.

As we will discuss later, it is clear that virtualization cannot currently meet the requirements of this full range of HPC applications and that an incremental approach will be needed to address the underlying issues.

3. TRENDS

There are two trends that define a larger context around the use of virtualization for HPC. The first relates to current and emerging pain points in HPC cluster deployments. The second concerns the changing relationship between enterprise IT and HPC.

3.1 Application Resilience

As HPC clusters continue to grow to support solving larger problems, the sheer number of hardware components in these systems will make it increasingly unlikely that applications will run to completion without experiencing at least one failure of the underlying hardware. Finding ways for applications to survive these failures is becoming increasingly important.

We believe a fundamental shift within the HPC community will be required to address this problem, with new approaches needed to achieve the required levels of application resiliency. The question is not *if* resiliency should be embraced, but rather *how* to achieve resiliency and how much it will cost because the solutions – whether they involve changes to hardware, operating systems, middleware, programming languages, or algorithms – will add overhead and reduce application efficiency.

This situation is analogous to what occurred during the move away from large SMP systems to Beowulf clusters in the mid 1990s [4]. This shift happened for economic reasons and also to allow higher levels of scale to be achieved. But, regardless of the motivation, the move from tightly-coupled multiprocessor systems to network-connected clusters introduced significant application performance degradations and also drove the broader adoption of message-passing programming models. It was a change with broad ramifications for HPC that are still being felt today.

3.2 IT Convergence

It is widely understood that cluster management complexity is one of the most challenging problems confronting the HPC community. Software for provisioning, updating, managing, and monitoring HPC clusters often consists of suites of purpose-built, open source components, sometimes mixed with components supplied by HPC software vendors. These suites, even when supplied by major vendors, are generally different from equivalent software stacks deployed to perform similar functions in enterprise datacenters. This is partly due to the historical roots of HPC clustering in open source, and also partly due to differing requirements in the two areas (e.g., in an HPC cluster environment many operations are typically applied to all nodes simultaneously since the cluster is essentially a single system from an HPC perspective).

We believe that enterprise and HPC requirements are converging and that it will be increasingly common to see converged management components used in both environments. This alignment is being driven by an increasing number of shared concerns, including power management, horizontal scale, fault tolerance, low-latency interconnect, and ease of management. As the complexities of both types of environments continue to grow, redundant implementations of the same capabilities will become economically infeasible over time. Practically speaking, given the relative sizes of the two markets and the associated sizes of vendor investments in these markets, we believe this means that enterprise management components – suitably augmented to handle HPC needs – will be used increasingly in HPC environments.

This convergence will also be driven by another trend – the broadening of HPC to encompass enterprises with increased requirements for computationally intensive workloads for data analytics and business intelligence, as well as more traditional HPC workloads to improve products through simulation, etc. MapReduce, for example, has become popular as a scalable framework for large-scale data analysis [2]. These organizations will have strong economic incentives to host these new workloads within their existing IT infrastructure rather than instantiating separate "HPC environments" requiring new software infrastructure and new staff to manage these different environments.

As this overlap in requirements continues to increase, it will become economically infeasible at some point to continue developing customized management software stacks for HPC, at least for the bulk of the HPC market. We believe a common, converged approach to address many[1] concerns is both possible and advisable. Indeed, we believe both HPC and enterprise environments will benefit from this convergence.

Virtualization is now well-established in the enterprise datacenter. In the rest of this paper we explain why virtualization should be viewed not as an enterprise technology, but instead as a component of converged IT – one that will bring significant value to both the enterprise and to High Performance Computing.

4. VIRTUALIZATION FOR HPC

In the enterprise datacenter, the primary value of virtualization is its ability to greatly reduce the number of physical servers required to host an organization's workload. Due to the performance profiles of many enterprise applications, it is often possible to host many virtual machines on a smaller number of physical servers and still deliver good application performance despite over-subscription of the underlying hardware resources. In other words, while the total number of virtual CPUs and memory allocated across all VMs running on a single physical machine may greatly exceed the machine's total physical memory and number of physical CPUs, the workload can still be adequately served. HPC workloads, in contrast, are much more resource intensive and would not therefore benefit from this kind of massive consolidation. Though some degree of consolidation (without resource over-subscription) will be useful for some types of HPC workloads, the primary value of virtualization for HPC generally lies elsewhere.

We have identified several uses for virtualization in HPC environments, each of which is described below. Because of the diversity of HPC requirements it is perhaps not surprising that customers and others we have talked with have not identified a single "killer use-case" for virtualization in HPC. However, all of those we spoke with identified at least one of these use-cases as potentially valuable for their HPC environments.

4.1 Cloud Computing

While accessing remote compute resources has been a common HPC model for many years[2], the use of *virtualized* remote resources as provided, for example, in current cloud deployments is a new aspect that will require a deeper understanding of which applications can run well in these environments and what enhancements are needed to both the cloud infrastructure as well as the base virtualization platform to enable good HPC application performance.

It is clear, however, that this desire to exploit access to potentially large amounts of computational power in the cloud without the capital expenditures traditionally required to run large-scale computations is alluring to HPC users. It is one of the two most common reason members of the HPC community have begun to express interest in virtualization for HPC workloads.

4.2 Application Resilience

As a lack of application resilience has come to be recognized as one of the largest barriers to increased application scaling on future systems, it has become more critical to find effective ways to safeguard application state in the presence of failing hardware and software. Protecting applications from underlying hardware failures is the second primary reason we've seen an increasing interest in virtualization for HPC. Virtualization can be used to provide both reactive and proactive fault tolerance.

Reactive fault tolerance provides protection by taking remedial action once a failure has occurred. Checkpoint/Restart – the ability to save and restore the state of a running job to disk – is the standard approach used in HPC environments. Capturing the complete state of an application has been difficult at least in part because saving and restoring the state of individual processes within a running operating system instance is complex. In addition, saving the state of an application's process (or processes) does not in general capture the full state of an application that reads and writes file system data.

Virtualization offers the potential of a better way to checkpoint based on snapshot functionality already available for virtual machines. The clean interface between hypervisor and virtual machine allows the entire state of the virtual machine to be captured, which can include the state of the virtual machine's virtual disk as well. In cases where application data resides on shared storage it would be possible in some cases to use a storage system's snapshot capabilities to capture file system state consistent with that of the saved VM, thus capturing the entire state of the application.

MPI applications could be handled in a similar way: By working in conjunction with an MPI implementation, it is possible to orchestrate the draining of in-flight messages and the writing of a checkpoint as a set of coordinated virtual machine snapshots. Open MPI is one such implementation [3, 5].

While checkpointing is an important capability, it is an expensive operation that requires writing the full state of all virtual machines to disk periodically, including the full memory state of each VM. In addition, the data must be written quickly to avoid interruption by hardware failures. As systems and their memories become larger and as failures become more frequent due to increased component counts, checkpointing becomes more problematic. For high-scale systems the component failure rate is projected to such that it will not be possible to write a full checkpoint before the next failure occurs [1].

[1]High-end, mission-critical HPC will likely continue to develop and deploy customized software solutions to address their specific concerns.

[2]NCSA was founded 1985, SDSC in 1985, PSC in 1986, and Teragrid in 2000

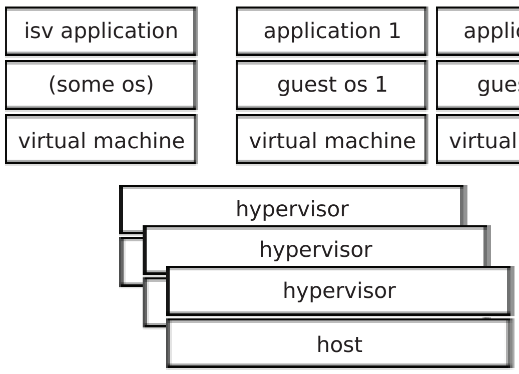

Figure 2: Heterogeneous HPC clusters

Proactive fault tolerance avoids failures by taking action to protect an application *before* an impending failure occurs. Assuming platform fault management agents could correctly predict useful classes of imminent system failures (e.g., an increasing number of soft memory errors as evidence that more serious failures are likely to occur), a running application (or pieces of a distributed application) could be live-migrated from a failing node to a healthy node to avoid the impending failure. This migration capability, called vMotion by VMware, is a mature and often-used capability in the virtualized enterprise datacenter. Migration of single-VM applications is relatively straightforward. More work would be required to migrate VMs containing individual MPI processes of an MPI job to enable MPI applications to survive underlying failures. Achieving this level of resilience would be a significant advance beyond checkpointing which, as discussed, may become untenable at extremely high scale.

4.3 Heterogeneity

Current HPC cluster systems are for the most part aggregations of homogeneous compute nodes all running an identical version of a pre-selected operating system. While this uniformity reduces administrative complexity and allows supported applications to be scheduled to any node, it is also extremely inflexible. Consider, for example, a large HPC installation designed to serve a user base with disparate application requirements. These applications may require Windows, Solaris, or an older version of Linux than is installed on the cluster. Since temporarily re-provisioning, rebooting, and reconfiguring nodes to satisfy these user requirements is not feasible, these users are often unable to use the shared resource for their work. They must instead look elsewhere for a more appropriate shared resource; port their application to the site's supported operating system; or purchase and configure a system for their own use. In some cases organizations run multiple clusters to support differing user requirements which can result in considerable resource under-utilization. Virtualization can greatly improve this situation.

By deploying a virtualized infrastructure across all compute nodes (Figure 2) rather than installing a standard operating system, administrators now free users to run virtual machines on the shared resource. Because the virtual machine is a black box from the site administrator's perspective, users are free to run any operating system with whatever HPC software stack they require to support their work. Old operating system revisions, completely different operating systems, old MPI libraries, experimental MPI libraries – users are free to make any such choices when building and configuring their virtual machines for later deployment on the shared cluster resource. This flexibility is beneficial for individual users and also for research communities that have created standardized software environments and applications to support distributed collaboration.

In addition to allowing any type of virtual machine to be run on their virtualized HPC system, a site may also elect to supply a standard virtual machine with a standard operating system and software stack for those end users who do not require the additional flexibility afforded by complete customization, thus supporting existing users while also adding the ability to address the requirements of a broader user base.

This shift to virtualized compute clusters will also benefit Independent Software Vendors (ISVs) who create the applications run by end-users at many HPC sites. ISVs currently must test their applications on all supported operating systems prior to release, which is arguably one of the most expensive components of their product development cycle. With virtualized compute clusters the ISV can instead choose a single operating system and middleware stack on which to develop, test, and deploy their application. Because the application can be shipped as a pre-configured virtual appliance, the ISV is now free to choose their operating system, compilers, libraries, etc., based on the quality of their implementation, the availability of advanced software development and debugging tools, the performance of the components, and perhaps the level of partnership and commitment of the OS and middleware providers selected, rather than on end-user requirements for a specific operating system. This shift to a single software configuration for development, testing, and deployment will allow ISVs to deliver products more quickly and with higher quality and higher performance.

4.4 Clean Computing

The shift from physical to virtual cluster infrastructure has other benefits as well. There are several in particular that can collectively be referred to as *clean computing*, which address some existing HPC pain points related to the current practice of running multiple applications within single operating system instances on nodes in a shared cluster resource.

4.4.1 Non-interference

Current physical compute clusters use standard HPC distributed resource managers (e.g., Sun Grid Engine, Platform Computing LSF, etc.) to schedule jobs onto compute nodes, often scheduling multiple applications per node to use all available resources. This can work well when no problems occur, but in situations in which jobs abort and leave the system in a corrupted state (e.g. /tmp has filled or an aborted job has not shut down properly and is holding a critical resource), workload throughput can be greatly reduced as further jobs are scheduled onto these broken nodes. System administrators spend significant time crafting home-grown approaches for dealing with these problems, including periodic node reboots to reset nodes to a known good state.

In a virtual environment in which each application runs within its own pre-configured virtual machine and starts in a well-defined and pristine state, each application is isolated from the failure of other applications or operating systems that occur in other virtual machines.

4.4.2 Consistency
Similarly, when launching an MPI job that is encapsulated within a set of virtual machines, all processes of the job are guaranteed to start in identically configured environments, avoiding issues of MPI or operating system misconfiguration that can prevent MPI applications from running properly.

4.4.3 Security
Security is another aspect of clean computing. Running each application in its own virtual machine allows multiple jobs to be run on the same physical hardware while providing a security barrier to prevent information leakage between jobs or users. This is especially important in shared environments in which multiple populations must be separated; for example, two departments within a company or academic and industrial users on a shared university resource or in a cloud environment.

4.5 Software Development
Virtualization can be useful for HPC developers as well. For example, debugging problems in long-running applications can be made more efficient using VM snapshots to save an application's state just prior to the point at which it fails. The debugging session can then be continually reset to this saved state and the program execution continued to collect more debugging information at the failure location.

In addition, virtualization can increase the effectiveness of distributed development teams. Such teams currently use centralized code repositories and version management systems to support development. Using virtualization technology, such teams can build and deploy standard developer VMs with pre-installed and pre-configured tools to guarantee that team members are using identical environments for development, avoiding issues with version skew on compilers and other support tools, and ensuring that operating systems are configured identically for all developers.

Virtualization can also be used to cost-effectively debug high-scaling HPC applications. For example, the correctness of a new MPI application can be explored on a modestly-sized physical test cluster prior to running the full-scale job on a large, expensive, shared resource by running a very large number of virtual machines on each node of the test cluster to simulate a full-scale application run. While many performance aspects cannot be tested in this way, the creation of this kind of large virtual cluster would allow a degree of correctness testing in an environment that closely mimics the full-scale production environment.

4.6 Dynamic Workload Migration
vMotion – the ability to move running virtual machines and applications from one node to another – was described earlier as an enabling technology for proactive application fault tolerance. This is one of several ways HPC environments can use this capability. In this section we describe several others.

4.6.1 Power Management
Migration can be used for power management by actively shifting running applications onto subsets of nodes when utilization drops, allowing other nodes to be powered off or placed in lower power states. To make effective power management decisions the management policies will need to be coupled to a representation of a site's physical layout and cooling technology.

4.6.2 Resource Efficiency
Current HPC distributed resource managers generally make job placement decisions based on the current load on cluster nodes and the resource needs of the next job to be scheduled. Once placed, jobs run until they complete, fail, or are suspended or killed to make room for higher-priority jobs. This static placement of workload can lead to inefficient use of resources. Using live-migration, it is possible to revisit placement decisions as new workload requests arrive. It can be used to rearrange running applications on a cluster to make room for jobs whose resource requirements cannot be met due to the current placement of workload. More sophisticated scenarios are supportable as well. For example, applications whose resource requirements change over the course of a run could be automatically shifted to another system to ensure its new resource requirements are met. VMware's Dynamic Resource Scheduler (DRS) performs exactly this function in current virtualized enterprise datacenters.

5. PERFORMANCE
We have outlined some of the primary benefits of virtualization for HPC. We turn now to a discussion of application performance in virtualized environments.

Previous studies have examined computational and I/O aspects of virtualized HPC workloads [13, 14, 10, 11, 7]. Current hardware support for virtualization (Intel VT, AMD-V) and the capabilities of modern virtualization software now generally deliver near-native performance on compute-bound workloads. The belief, based on experimentation with previous generations of hardware and virtualization software, that such workloads are significantly impacted by virtualization is now outdated and often incorrect. We refer here specifically to single-process, throughput-oriented workloads as are often found in Financial Services, Electronic Design Automation, Digital Content Creation, and Life Sciences.

The situation with respect to MPI applications is different. Based on experimentation in cooperation with Purdue University, we have seen that I/O (storage, interconnect, networking) performance presents a larger challenge for virtualized HPC workloads, especially latency-sensitive I/O. We are using these preliminary results to guide further, more detailed performance experiments to uncover specific issues to motivate future product improvements. Some of these issues are discussed in more detail in Section 6 and others will be addressed in future work.

Table 1: Native memory bandwidth (MB/s)

	Threads				
	1	2	4	8	16
Copy	6388	12163	20473	26957	26312
Scalar	5231	10068	17208	25932	26530
Add	7070	13274	21481	29081	29622
Triad	6617	12505	21058	29328	29889

Because we have only recently started our analysis of virtualization for HPC, we report below on just two preliminary benchmark results comparing native and virtualized performance on STREAM and NAMD, two of the Purdue benchmarks. We will be expanding our testing to include additional applications and benchmarks and will be examining compute, storage, and interconnect performance in detail for a range of HPC workloads.

5.1 Benchmark: STREAM

STREAM [8] is a benchmark that measures system memory bandwidth, which is critical to a large class of HPC workloads. Our test compares STREAM performance in a virtual environment against performance achieved on a native system using identical hardware and operating system. The OpenMP version of STREAM (V5.9) was used.

All tests in this section and the next were performed on an HP DL380 with two Intel X5570 processors, 48 GB memory (12×4 GB DIMMs) with 24 GB connected directly to each CPU socket, and four 1-GbE NICs (Intel Pro/1000 PT Quad Port Server Adapter) connected to a switch. RHEL 5.5 x86_64 was used in both the virtual and native cases. Hyper-threading is enabled in the BIOS unless otherwise noted, so 16 logical processors are available. A pre-release version of ESX 4.1 was used, build 254859. We expect the performance on ESX 4.1, which has subsequently been released, to be the same.

The array size (`N`) and number of iterations (`NTIMES`) are hard-wired in the code as `N=`10^8 (for a single machine) and `NTIMES=40`. The large array size ensures that the processor cache provides little or no benefit and allows measurement of bandwidth to main memory. STREAM reports maximum memory bandwidth performance in MB/s for four tests: copy, scale, add, and triad. Table 1 shows the native results as a function of the number of threads. Note that the scaling starts to fall off after two threads and the memory links are essentially saturated at 8 threads. This is one reason why HPC applications often do not see much benefit from enabling hyperthreading.

To achieve the maximum aggregate memory bandwidth in a virtualized environment, two virtual machines (VMs), each with 8 virtual CPUs, were used. One instance of STREAM with `N=`5×10^7 was run in each VM simultaneously so the total amount of memory accessed was the same as in the native test. The advanced ESX configuration option `preferHT=1` is used (see below). Bandwidths reported by the VMs are summed with the results shown in Table 2: The two VMs combined achieve a slightly higher bandwidth than the corresponding native case.

Table 2: Virtualized total memory bandwidth (MB/s), 2 VMs, `preferHT=1`

	Total threads			
	2	4	8	16
Copy	12535	22526	27606	27104
Scalar	10294	18824	26781	26537
Add	13578	24182	30676	30537
Triad	13070	23476	30449	30010

It is apparent that the OpenMP runtime together with the simplicity of the STREAM algorithm are enough to ensure that nearly all memory accesses in the native tests are "local" (that is, each execution thread accesses only memory local to its socket.) In ESX 4.1, NUMA information is not passed to the guest OS and (by default) 8-vCPU VMs are scheduled across CPU sockets in order to take advantage of more physical cores. This means that about half of memory accesses will be "remote" and that in the default configuration one or two VMs will achieve significantly lower bandwidth than the native tests. Setting `preferHT=1` causes the ESX scheduler to count logical processors (hardware threads) instead of cores when determining if a given VM can fit on a socket. In this case, both memory and CPU of an 8-vCPU VM are forced onto a single CPU socket. This guarantees all memory accesses are local and the aggregate bandwidth of two VMs can then equal or exceed native bandwidth.

Note, however, that a single VM cannot currently match the maximum achieved native bandwidth. The single VM will achieve either half the peak native rate (when running on one socket) or about 70% of native peak (when running across both sockets). In both native and virtual environments, the maximum bandwidth of purely remote memory accesses is about half that of purely local. On machines with more CPU sockets, the importance of memory locality increases since the fraction of memory that is remote from a given socket increases significantly.

Preliminary tests have shown that if the NUMA aspects of the underlying hardware can be exposed to the guest OS, then a single VM running a single instance of STREAM can match the results seen in the native case.

5.2 Benchmark: NAMD

NAMD [9] is a commonly-used molecular dynamics application from the University of Illinois. It is used by researchers to investigate the properties of large molecules. NAMD includes a mix of computation and communication, allowing us to evaluate both. The main goal here was to explore the effect of network latency in the context of a single, distributed application. NAMD supports both CHARM++ and AMPI on parallel and multi-machine systems. For our tests, we used the multi-process CHARM++ version of NAMD 2.7b2 running over TCP/IP.

The hardware is the same as that used for the STREAM tests, but with the four 1-GbE NICs connected to a switch. RHEL 5.5 x86_64 was used in both the virtual and native cases. Hyper-threading is enabled in the BIOS unless otherwise noted, so 16 logical processors are available. A pre-release version of ESX 4.1 was used, build 254859. As with

Table 3: NAMD elapsed time (seconds), STMV molecule, HT disabled

	Total processes	
	4	8
Native	1748	915
1 VM	1768	926

STREAM, we expect the performance of this test on ESX 4.1 to be the same.

The native results use up to 16 processes on a single machine ("local" mode). Future work will use multiple machines, but some idea of the performance issues involved can be obtained by running multiple VMs in various configurations on the same physical host using physical networking between VMs.

The benchmark consists of running 500 steps of the Satellite Tobacco Mosaic Virus. "STMV" consists of slightly over one million atoms, which is large enough to enable good scaling on fairly large clusters. Shown in the following tables are elapsed time measurements for various configurations. Each is an average of three runs and the repeatability is good. The virtual NIC is e1000 for all the virtualized cases.

A comparison between native and virtual is obtained by disabling hyperthreading in the machine BIOS and using a single 8 vCPU VM, configured with 12 GB and using default ESX parameters. With no networking, the virtual overhead is just 1% as shown in Table 3.

The effect of splitting the application across multiple machines and using different network configurations can be tested in a virtual environment. For these tests HT is enabled to get the full performance of the machine. The single VM case is configured as above. The 2-VM cases are configured with 12 GB, 8 vCPUs, and `preferHT=1` (so each VM can be scheduled to a different CPU socket). The 4-VM cases have 6 GB, 4 vCPUs, and `preferHT=0`. For this application, the exact amount of memory in each VM is not important, as long as neither the host nor the VMs experience memory pressure. ESX virtualizes physical Ethernet switches much as it virtualizes memory, processors, and storage. The result is called a vSwitch [12]. A vSwitch detects which virtual machines are logically connected to each of its virtual ports and uses that information to forward traffic to the correct virtual machines. Virtual networks can be joined to physical networks by associating one or more physical Ethernet adapters to a vSwitch and connecting the adapters to physical switches. Physical NICs and switches are not always needed. A simple example is our "1 vSwitch" case, where multiple VMs on the same host communicate using the same vSwitch. No physical networking components are used, and ESX handles all the network traffic in memory. This configuration allows for various optimizations that result in low latency connections. For the multiple vSwitch cases, each vSwitch is associated with a physical NIC which is connected to a physical switch. Since all networking traffic must go through this switch, this configuration will be the same as using multiple hosts in terms of inter-VM communication latencies. The elapsed time for each of these configurations is shown in Table 4.

Table 4: NAMD elapsed time (seconds), STMV molecule, HT enabled

	Total processes		
	4	8	16
Native	1761	1020	796
1 VM	1766	923	-
2 VMs, 1 vSwitch	1779	928	787
2 VMs, 2 vSwitches	1800	965	806
4 VMs, 1 vSwitch	1774	940	810
4 VMs, 4 vSwitches	1885	1113	903

The single VM case shows that HT has little effect on ESX performance when the extra logical processors are not used. However HT does slow down the native 8 process case significantly. This appears to be due to Linux not scheduling one process per core when it has the opportunity, which the ESX scheduler does by default. Scalability from 4 to 8 processes for the single vSwitch cases is close to 1.9X, and from 8 to 16 processes (using the same number of cores, but taking advantage of HT) it is 1.17X. This is excellent scaling. Networking over the switch reduces the performance somewhat, especially for four vSwitches. Scaling for native is reduced because the application does not manage NUMA resources itself, and Linux is limited by how well it can do this. This allows one of the 16-process virtualized cases to be slightly faster than native, despite the virtualization and multiple-machine overheads. The 16-process cases have the best absolute performance, and therefore correspond to how NAMD would actually be configured in practice. Here, the performance of all the virtualized cases is very close to native, except for the 4-vSwitch case where the extra overhead of networking has a significant effect. This is expected and should not be compared to the native case since the virtual case models four hosts. We plan to investigate multiple-host scaling soon to enable a direct comparison. A useful simulation needs up to ten million steps, which would only be practical on a large cluster.

The flexibility of virtualization enables workload to be configured in multiple ways. When this multi-process scientific application is run as a set of four virtual machines interconnected with virtual networking on a single host, we find virtualization adds little or no overhead. If instead we more closely simulate a physical cluster configuration by connecting those VMs with actual networking hardware, we see a slowdown of about 12%. Four physical hosts are needed to determine the analogous slowdown for a native configuration, which we leave for a future test. The negligible overhead of virtual networking for this application shows that when the number of physical processors is larger than the allowed number of virtual CPUs per VM, large machines can be divided into two or more virtual machines with little performance penalty.

6. CHALLENGES

We have described the potential benefits of virtualization for HPC and we have presented evidence that some aspects of HPC workload run well now in a virtualized environment. However, there is more work to be done to deliver acceptable performance for a broader array of HPC applications and to fully enable the benefits described in Section 4. In this section we outline some of the major challenges that must be addressed.

6.1 Workload Management

We have discussed the differences between HPC application requirements and those of traditional enterprise applications. There is also a significant difference in the workloads – the mix of applications – running in these two environments. In an enterprise setting, application resource requirements, the relative priority of applications, and policies about application co-location are all generally planned in advance. With policies and resource reservations in place, applications are then run under control of the Distributed Resource Scheduler (DRS), which dynamically balances load across machines using vMotion while enforcing fairness and guaranteeing resource reservations are satisfied.

In an HPC environment, the workload is more dynamic with users presenting a potentially unbounded set of applications to a distributed resource manager (DRM) to be scheduled across the cluster. The DRM will generally make static placement decisions based on currently available cluster resources and on the resources requested for the current job. Once placed, applications either run to completion, fail, or are terminated. There is no mechanism to dynamically migrate workload after initial placement.

Mechanisms are needed to support running and managing this dynamic HPC job stream on virtualized infrastructure which has been designed primarily to support enterprise workloads and their characteristics. Broadly, there are two approaches to achieve this.

The first option, which we label *evolutionary*, offers an end-user experience identical to that of native (non-virtualized) HPC clusters. In this approach, an HPC cluster is run on a virtualized infrastructure as a set of persistent virtual machines, each running an OS instance and the usual DRM daemons. While the virtualized nature of this cluster is visible to the site administrator, it is hidden from both the DRM and the end-user and thus appears as a native HPC cluster, though the site administrator may still perform vMotion operations on these VMs for load balancing, power management, etc. In addition, if supported by the DRM, the site administrator can easily add or remove virtual machines to dynamically grow or shrink the HPC cluster as demand fluctuates.

Hadoop clusters could be virtualized in a similar way, though there are additional issues to be addressed. For example, HDFS uses a local storage model whereas virtualized environments most often use a shared storage approach. In addition, HDFS is designed to be aware of the underlying datacenter topology to avoid single points of failure by (for example) placing data replicas on systems in different chassis. In a virtualized environment, the physical arrangement of hosts is hidden from guest operating systems.

As both the Dynamic Resource Scheduler (DRS) and the Distributed Resource Manager (DRM) may be active simultaneously in this evolutionary approach, it will be important to determine how these two resource management layers should interact to avoid conflicts and to increase overall resource utilization.

The second, or *revolutionary* option, represents a significant shift in the way applications are packaged, provisioned, and executed. With this approach, applications are delivered as pre-built and pre-configured VMs – as virtual appliances. These VMs are transient in that they are only active for the time needed to run their embedded application.

To implement the revolutionary approach, a DRM or DRM-like capability must be integrated with the virtualization infrastructure to support the rapid provisioning and management of HPC-style applications, both single- and multi-process. In addition, the dynamic nature of the HPC workload must be handled and at a scale appropriate for typical HPC cluster deployments.

Each model has its strengths and weaknesses. Of the two models, the revolutionary approach is needed to deliver all of the values described in Section 4. The evolutionary approach offers a subset of those values, but with the benefit that initial adoption can be made transparent to end-users, a significant advantage.

6.2 Scalability

From the perspective of a single physical machine, the profile of an HPC workload is very different from that found on a machine in an enterprise environment. In an enterprise, where massive consolidation is a primary value for virtualization, each machine typically hosts a large number of VMs – possibly 40-50 or more on a modest-sized multi-socket system. This over-subscription of resources is possible due to the relatively light resource requirements of typical enterprise applications. In contrast, the total number of virtual CPUs allocated across all VMs scheduled on a single machine in an HPC environment would not typically exceed the number of available physical cores in the system. This smaller number of virtual machines and the absence of resource over-subscription should allow the hypervisor scheduler to more easily deliver high performance to each VM.

While the number of VMs per machine in an HPC environment would be considerably lower than that found in an enterprise environment, each HPC VM would in many cases have more memory and virtual CPUs (vCPUs) allocated than would be found in a typical enterprise VM. This is because relatively few enterprise applications are parallelized to take advantage of more than a few vCPUs: the majority of enterprise VMs are currently single or dual-vCPU virtual machines, well below the current vSphere limit of eight vCPUs per VM. There are many current OpenMP and other threaded HPC applications that are parallelized beyond this limit. This is an important aspect of scalability that must be addressed to fully support HPC workloads in a virtualized environment.

MPI application requirements differ substantially from those of enterprise applications. Starting a virtualized MPI application will require many identical VMs be launched simultaneously. As an MPI application runs, it is important that the VMs that compose the application be scheduled simultaneously to minimize jitter and maintain high performance. In addition, checkpointing or migrating MPI applications will place high load on the virtualization infrastructure and in a manner not typical of enterprise environments, potentially stressing current platform limits.

Scalability is one aspect of the MPI challenge to be addressed. In the next section we discuss interconnect-related issues that are equally important for delivering good MPI performance in a virtualized environment.

6.3 Interconnect & Accelerators

Perhaps the most difficult challenge to address to enable virtualization for HPC is that of allowing application access to specialized acceleration hardware, most notably InfiniBand and GPUs. Without access to InfiniBand's high bandwidth and low latency, the performance of many MPI applications will be unacceptably degraded when virtualized. Similarly, as the use of GPU for general purpose computation becomes more common in HPC, lack of support for this model will also unacceptably impact application performance.

The vSphere platform does currently support a capability called DirectPath that allows hardware devices to be made visible to a guest operating system running in a virtual machine environment. The use of this so-called passthrough mode is problematic, however. Much of the value of virtualization for both the enterprise and for HPC are linked to vMotion and the ability to live-migrate workloads from machine to machine. Because DirectPath allows real hardware to break the virtual machine abstraction, we lose the ability to migrate such virtual machines with vMotion. The challenge then is to engineer a capability that allows unfettered access to these accelerator technologies while also maintaining the value of vMotion.

These are not insurmountable problems. For example, internal experiments have demonstrated that GPGPU[3] access which supports an ability to use vMotion is feasible. We believe similar approaches could be used to enable access to high-speed, low latency interconnects as well. Other research in this area [14, 6] is in agreement.

6.4 Network Latency

Network latency is another area of concern. In cases where small message transfers are important, whether for NAS-based storage or for inter-node communication, the additional processing currently required to handle device interrupts in a virtual environment can adversely affect application performance. We are working to characterize this issue more precisely to better understand the magnitude of the effect, but preliminary indications are that this is a significant issue that will need to be addressed. It is also clear that this is not an HPC-specific concern given the emerging importance of distributed middleware services in the enterprise; for example, Gemstone GemFire, Oracle Coherence and others.

6.5 Storage Requirements

Enterprise virtualization deployments are built on a shared storage infrastructure to enable capabilities like vMotion. This shared infrastructure is often SAN-based and can be heavily loaded depending on a site's application workload profile and its use of vMotion, snaphots, and other capabilities of the virtualized infrastructure.

[3]General Purpose computation on Graphics Processing Unit (GPU).

Many low-end and midrange HPC sites use NFS, often over 1 Gb Ethernet, as shared storage infrastructure. Virtualizing such an HPC environment will require careful consideration of the additional load placed on NAS storage and networking by the virtualization infrastructure. In some cases, network upgrades may be advisable to handle the larger load and these additional expenses must be considered as part of the cost-benefit analysis of a shift to a virtualized HPC approach.

The cluster file systems (e.g., Sun Lustre, IBM GPFS) often used by high-end customers may be able to more easily handle the additional load generated by the virtualization infrastructure. The challenge in this case will be to support these HPC-specific file systems as underlying storage options for the virtual infrastructure layer which currently supports only SAN and NFS-based storage.

For sites with application dependencies on local disk, shifting to shared storage may incur unacceptable performance penalties. In such cases, it is possible to use a physical machine's local disk as storage for a virtual machine's virtual disk, but currently with loss of the ability to live-migrate virtual machines to other physical machines.

7. CONCLUSIONS

We believe virtualization technologies can offer significant value for High Performance Computing, from access to cloud resources to proactive application resilience and efficient resource use via dynamic workload migration.

Performance, however, is always of primary concern in HPC. While some types of HPC workload will run well now in a virtualized environment, there is much future work needed to enable a broader array of HPC applications to run well when virtualized. We have outlined some of the major challenge areas that must be addressed.

8. REFERENCES

[1] F. Cappello. Fault tolerance in petascale/ exascale systems: Current knowledge, challenges and research opportunities. *Int. J. High Perform. Comput. Appl.*, 23(3):212–226, 2009.

[2] Y. Chen, D. Pavlov, and J. F. Canny. Large-scale behavioral targeting. In *KDD '09: Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 209–218, New York, NY, USA, 2009. ACM.

[3] E. Gabriel, G. E. Fagg, G. Bosilca, T. Angskun, J. J. Dongarra, J. M. Squyres, V. Sahay, P. Kambadur, B. Barrett, A. Lumsdaine, R. H. Castain, D. J. Daniel, R. L. Graham, and T. S. Woodall. Open MPI: Goals, concept, and design of a next generation MPI implementation. In *Proceedings, 11th European PVM/MPI Users' Group Meeting*, pages 97–104, Budapest, Hungary, September 2004.

[4] W. Gropp, E. Lusk, T. Sterling, and J. Hall. *Beowulf Cluster Computing with Linux, 2nd Edition*. The MIT

Press, Cambridge, Massachusetts, 2003.

[5] J. Hursey, T. I. Mattox, and A. Lumsdaine. Interconnect agnostic checkpoint/restart in Open MPI. In *HPDC '09: Proceedings of the 18th ACM international symposium on High Performance Distributed Computing*, pages 49–58, New York, NY, USA, 2009. ACM.

[6] J. Liu, W. Huang, B. Abali, and D. K. Panda. High performance VMM-bypass I/O in virtual machines. In *ATEC '06: Proceedings of the annual conference on USENIX '06 Annual Technical Conference*, pages 3–3, Berkeley, CA, USA, 2006. USENIX Association.

[7] J. C. Martinez, L. Wang, M. Zhao, and S. M. Sadjadi. Experimental study of large-scale computing on virtualized resources. In *VTDC '09: Proceedings of the 3rd international workshop on Virtualization technologies in distributed computing*, pages 35–42, New York, NY, USA, 2009. ACM.

[8] J. D. McCalpin. Memory bandwidth and machine balance in current high performance computers. *IEEE Computer Society Technical Committee on Computer Architecture (TCCA) Newsletter*, pages 19–25, December 1995.

[9] J. C. Phillips, R. Braun, W. Wang, J. Gumbart, E. Tajkhorshid, E. Villa, C. Chipot, R. D. Skeel, L. Kale, and K. Schulten. Scalable moleclar dynamics with NAMD. *Journal of Computational Chemistry*, 26:1781–1802, 2005.

[10] A. Ranadive, M. Kesavan, A. Gavrilovska, and K. Schwan. Performance implications of virtualizing multicore cluster machines. In *HPCVirt '08: Proceedings of the 2nd workshop on System-level virtualization for high performance computing*, pages 1–8, New York, NY, USA, 2008. ACM.

[11] A. Tikotekar, G. Vallée, T. Naughton, H. Ong, C. Engelmann, S. L. Scott, and A. M. Filippi. Effects of virtualization on a scientific application running a hyperspectral radiative transfer code on virtual machines. In *HPCVirt '08: Proceedings of the 2nd workshop on System-level virtualization for high performance computing*, pages 16–23, New York, NY, USA, 2008. ACM.

[12] VMware, Inc. VMware Virtual Networking Concepts. `http://www.vmware.com/files/pdf/virtual_networking_concepts.pdf`.

[13] L. Youseff, K. Seymour, H. You, J. Dongarra, and R. Wolski. The impact of paravirtualized memory hierarchy on linear algebra computational kernels and software. In *HPDC '08: Proceedings of the 17th international symposium on High performance distributed computing*, pages 141–152, New York, NY, USA, 2008. ACM.

[14] L. Youseff, R. Wolski, B. Gorda, and C. Krintz. Evaluating the performance impact of Xen on MPI and process execution for HPC systems. In *VTDC '06: Proceedings of the 2nd International Workshop on Virtualization Technology in Distributed Computing*, page 1, Washington, DC, USA, 2006. IEEE Computer Society.