WHITE PAPER – **APRIL 2018**

VIRTUALIZING HPC THROUGHPUT COMPUTING ENVIRONMENTS

Contents

Introduction

High Performance Computing (HPC) workloads have traditionally been run only on bare-metal, unvirtualized hardware. However, performance of these highly parallel technical workloads has increased dramatically over the last decade with the introduction of increasingly sophisticated hardware support for virtualization, enabling organizations to begin to embrace the numerous benefits that a virtualization platform can offer.

To demonstrate the results of this continuing performance improvement, this paper explores the application of virtualization to HPC and evaluates the performance of HPC workloads in a virtualized, multitenant computing environment.

With parallelism being the key to achieving high performance, HPC usually involves a large number of processes that run simultaneously. Depending on the amount of communication among the parallel processes, HPC workloads can be broadly divided into throughput workloads and parallel distributed workloads. This pioneering study focuses primarily on the virtual performance of throughput workloads. Detailed information related to parallel distributed workloads—primarily MPI workloads—can be found in “References” [2,3].

Virtualizing HPC Workloads

Although HPC workloads are most often run on bare-metal systems, this has started to change over the last several years as organizations have come to understand that many of the benefits that virtualization offers to the enterprise can often also add value in HPC environments. The following are among those benefits:

- Supports a more diverse end-user population with differing software requirements – By using virtual machines (VMs), each user or group can run the operating system (OS) and other software that are most effective for their needs, and these different software environments can be freely mixed on the same hardware. In addition, that mix can be changed dynamically as user requirements change, which enables IT departments to increase overall agility and to help decrease time to solution for researchers, scientists, and engineers.
- Provides data security and compliance by isolating user workloads into separate VMs, including running within different software-defined virtual networks – This ensures that projects (for example, studies involving clinical data) maintain control over their data and that the data is not shared inappropriately with other users. In addition, data security can be achieved while allowing projects to share underlying hardware, increasing overall utilization of physical resources.
- Provides fault isolation, root access, and other capabilities not available in traditional HPC environments through the use of VMs – By running users’ jobs within dedicated VMs, each user can be protected from problems caused by other users, a common issue in bare-metal HPC environments in which jobs from multiple users are frequently run within the same OS instance. In addition, the isolating nature of the VM abstraction means that root access can be granted to those users who require it, because that privilege is granted only within the VM and it does not compromise the security of other users or their data.

- Creates a more dynamic IT environment in which VMs and their encapsulated workloads can be live-migrated across the cluster for load balancing, for maintenance, for fault avoidance, and so on – This is a considerable advance over the traditional approach of statically scheduling jobs onto a bare-metal cluster without any ability to reassess those placement decisions after the fact. Such dynamic workload migration can increase overall cluster efficiency and resilience.

HPC Workloads

In HPC, performance is of paramount importance, and delivering high performance in the VMware virtual environment has been a key part of the work required to make virtualization viable for these workloads.

HPC workloads can be broadly divided into two categories: parallel distributed workloads and throughput workloads. Parallel distributed applications—often these are MPI applications, referring to the most popular messaging library for building such applications—consist of many simultaneously running processes that communicate with each other, often with extremely high intensity. Because this communication is almost always in the critical performance path, the HPC community has adopted specialized hardware and software to achieve the lowest possible latency and highest bandwidth to support running these applications efficiently.

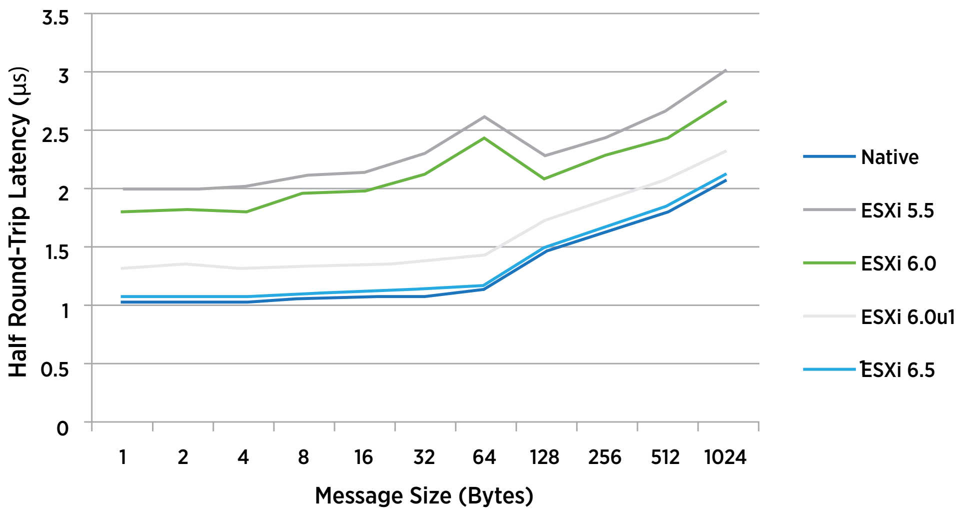

Figure 1. Improvement in InfiniBand Latency over Several Versions of the ESXi Hypervisor

NOTE: Lower is better.

InfiniBand and RDMA are the two most widely used hardware and software approaches for HPC message passing, and they can be used in a virtual environment as well. Figure 1 shows how InfiniBand latencies under a VMware vSphere® platform using VMware vSphere DirectPath I/O™ have improved over the last several releases of VMware ESXi™, the vSphere hypervisor. Latencies now approach those achievable without virtualization, and Figure 2 shows performance results for a variety of popular open-source and commercial MPI applications. These tests were run on a 16-node EDR InfiniBand cluster running one large VM per node. As can be seen when compared to Figure 3, the degradations can be higher with this workload class than with throughput workloads. Because overheads depend on the specific application, the model being used, and the scale at which the application is run, a proof-of-concept deployment is often recommended to determine achievable acceptable performance. Additional information about MPI performance can be found on the Dell Community site [3].

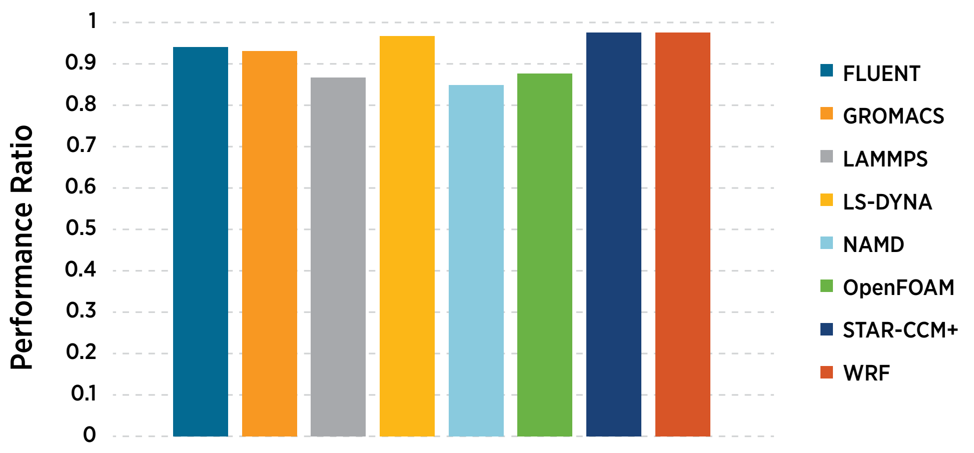

Figure 2. Performance of a Variety of MPI Applications Running on a 16-Node Cluster Using 320 MPI Processes

NOTE: Results are shown as the ratios of unvirtualized and virtualized performance. Higher is better.

Throughput workloads often require a large number of tasks to be run to complete a job, with each task running independently with no communication between the tasks. Rendering the frames of a digital movie is a good example of such a throughput workload: Each frame can be computed independently and in parallel; when all frames have been computed, the overall job has been completed. Throughput workloads currently run with very little degradation on vSphere, typically either a percentage point or two of degradation, and in some circumstances they can run slightly faster when virtualized. Figure 3 shows performance comparisons for a popular set of life sciences throughput benchmarks, illustrating that virtual performance can be very similar to unvirtualized for this workload class. In these tests, we compare the runtime of each program in the benchmark suite, running each within a VM and on bare metal, using identical hardware and OSs for each test.

For these tests, we run just a single instance of a benchmark at a time and study how well the performance of that benchmark compares when run in a virtual environment as opposed to a bare-metal environment. In later sections, we move beyond such simple evaluations of individual application performance and focus instead on the more typical case in which a large number of such tasks are scheduled and run on the nodes of an HPC cluster. In this scenario, the metric of interest is cluster throughput—the time needed to complete a specified set of tasks on the cluster.

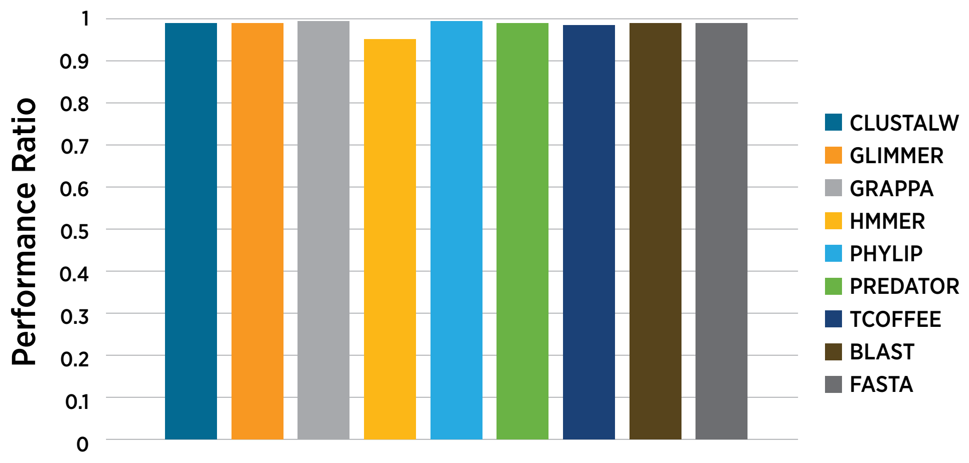

Figure 3. Performance of Typical HPC Throughput Applications

NOTE: Results are shown as the ratios of unvirtualized and virtualized performance. Higher is better.

HPC Environments

HPC infrastructure typically consists of a cluster of nodes, ranging from tens to hundreds of thousands, to support a large degree of parallelism. The nodes in such a complex system are often split into multiple partitions based on their roles, such as login nodes, management nodes, and compute nodes. An illustration of an HPC platform can be seen in Figure 4. Rather than directly accessing the compute resources, users submit and manage jobs via login nodes, which are sometimes duplicated for load balancing and fault tolerance. To efficiently share the resources among multiple users while being able to enforce specific rules for fairness and quality of service, most production HPC systems execute user jobs on a pool of compute nodes in a batch mode. That is, each user-submitted job is first put into a job queue, waiting until a job scheduler acquires the requested resources from a resource manager. After the resources are obtained, they are then allocated to the job. The job scheduler and resource manager are management services that run on dedicated management nodes.

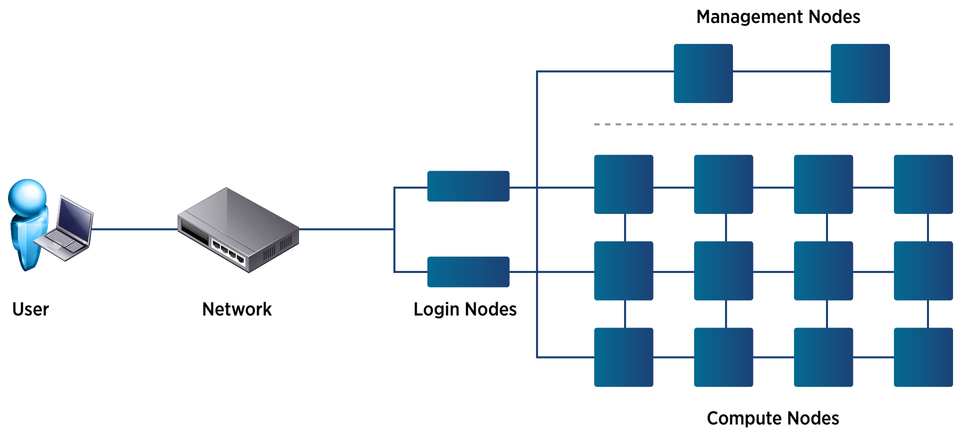

Figure 4. Illustration of an HPC Platform

Virtualization Considerations

Virtualization brings new flexibility to HPC. With that flexibility, however, one must be careful to configure the virtual environment to simultaneously achieve both agility and high performance. This section provides some guidance relative to throughput computing environments—that is, environments in which most or all of the tasks are throughput computing tasks.

Although it is possible to create a single virtual cluster that spans an entire physical cluster with one maximally sized VM per node, this approach misses the opportunity to enable several important virtualization benefits. Among these are the ability to support per-user or per-project software stacks as well as security and fault separation between user workloads. In the more typical case, multiple virtual clusters should be hosted simultaneously on the physical cluster.

In a situation in which a project is using resources very intensively—when they are essentially driving their assigned physical resources to 100 percent utilization continuously—it might make most sense to assign the project to its own dedicated subset of hardware and to configure VMs on those nodes appropriately for the project. With two such projects, one can either place their VMs on a nonoverlapping set of nodes or place them on the same nodes, being careful to size their VMs to avoid any overcommitment. Typically, the size of each such VM is set to the number of cores available in each underlying physical CPU to take advantage of the hardware NUMA topology. This approach works well if the virtual clusters are so heavily loaded that few cycles are left unused on each node. However, it is often the case that such projects do not in actuality show this degree of resource intensity. Although they might be very busy and resource constrained in some time periods, they can also be less busy and even idle in other periods. In such cases, sizing VMs as described—so that they partition the available physical cores—can lead to commensurate losses in throughput because the idle CPUs serving one VM are not available to the other busy VM.

The key to avoiding this potential issue is CPU overcommitment. By creating two VMs on a host, each configured to the full number of physical cores on the node, it is now possible for each VM to potentially use all available CPU resources when the other VM is idle. This paper explores this scenario in detail to evaluate whether such an approach can deliver similar job throughput rates to that achievable with a bare-metal cluster.

The experiments are conducted in two phases: First, the evaluation is done with the assumption that each virtual cluster is given equal access to the underlying physical resources. In the second phase, we introduce the concept of CPU shares, which enable assigning relative scheduling weights to the VMs of a cluster so that one cluster can be scheduled preferentially relative to another. Customers have found that this approach brings significant flexibility to their virtualized HPC environments by adding a quality-of-service capability.

Test Bed Configuration

To evaluate performance in a realistic HPC throughput computing environment, we built an 18-node test bed with Dell PowerEdge servers as a part of a collaboration with the Dell EMC HPC and AI Innovation Lab in Austin, Texas. In this test bed, one server is dedicated as the login node, one as the management node, and the remaining 16 as the compute nodes. Each node is configured with dual 10-core processors and 128GB of memory. More details can be found in Table 1. To achieve the best performance, BIOS settings are tuned, as shown in Table 2.

Cluster	18 nodes
Server	Dell PowerEdge C6320
Processor	Dual 10-core Intel Xeon Processor E5-2660 v3 @ 2.6GHz (Haswell)
Memory	128GB DDR4
Interconnect	1Gb Ethernet

Table 1. Hardware Configuration

System Profile	PerfOptimized
Logical Processor	Enabled
Virtualization Technology	Enabled
Turbo Boost Technology	Enabled

Table 2. BIOS Settings

For our performance evaluation, each node is installed with two execution environments, that is, a native OS and a virtualized guest OS on top of an ESXi hypervisor. For fairness, CentOS 7.2 with kernel version of 3.10.0-327.el7.x86_64 is used in both environments. Throughout the experiments, the TORQUE Resource Manager is used with its default job scheduler. The software stack is shown in Table 3. Seven HPC throughput benchmarks from the PolyBench/C version 3.1 and BioPerf benchmark suite are selected, as shown in Table 4. When running throughput tests in various configurations, each of these seven benchmarks is run 464 times in a random but consistent order to generate an hour-long job sequence.

ESXi Hypervisor	6.5.0
Native/Guest OS	CentOS 7.2 (3.10.0-327.el7.x86_64)
Resource Manager/Job Scheduler	TORQUE 6.1.0/pbs_sched
Benchmarks	Seven benchmarks from PolyBench/C 3.1 and BioPerf

Table 3. Software Stack

3mm	Three matrix multiplications
gemm	Matrix-multiply
symm	Symmetric matrix-multiply
syr2k	Symmetric rank-2k operations
syrk	Symmetric rank-k operations
trmm	Triangular matrix-multiply
T-Coffee	Sequential multiple sequence alignment

Table 4. Benchmarks

CPU Overcommitment

With the test bed described in the previous section, a fair performance comparison between bare-metal and virtual clusters is to contrast the completion time for a fixed sequence of jobs. For this test, we first boot Linux on each node and time how long it takes to run the job stream through this physical TORQUE cluster. We then reboot the machines with the ESXi hypervisor and run the same throughput test on a virtual TORQUE cluster built using one VM per node. However, in a virtualized HPC environment, there is an extra configuration parameter—the number of VMs on each host—an exclusive advantage of virtualization that supports multitenancy and resource sharing. As has been confirmed in many enterprise use cases, resource consolidation can improve utilization and thereby increase overall throughput. To verify if this also applies to HPC throughput computing, we experimented with CPU overcommitment in this work.

While keeping each VM configured with the same number of cores as its physical host, CPU overcommitment is achieved through simultaneously running multiple VMs on each host. Specifically, one, two, and four VM(s) per host have been tested to achieve up to 4X CPU overcommitment. When multiple VMs execute on an ESXi host, a work-conserving, share-based scheduler allocates CPU time to VMs based on the shares of each VM [1]. In this first set of tests, all VMs are configured with equal shares. The performance with unequal shares will be discussed in the next section.

To mimic a real user-private, overcommitted execution environment, four TORQUE clusters are built to create four separate virtual clusters. Each virtual cluster consists of one TORQUE head node and 16 TORQUE MOMs,[1] all of which run as VMs spread across the 18 hosts of the underlying physical cluster. This is illustrated in Figure 5. In each experiment, the number of virtual clusters that are powered on depends on the overcommitment factor. For example, when 2X CPU overcommitment is chosen, only two virtual clusters will be powered on, while the other two will not consume any resource on the physical cluster.

To study solely the effect of CPU overcommitment and avoid memory overcommitment, in the virtual environment 28GB of memory on each node is always reserved for the ESXi hypervisor instance, and the remaining 100GB is evenly split among the running VMs. For example, when four VMs are running on each host, each VM is given a reservation of 25GB memory.

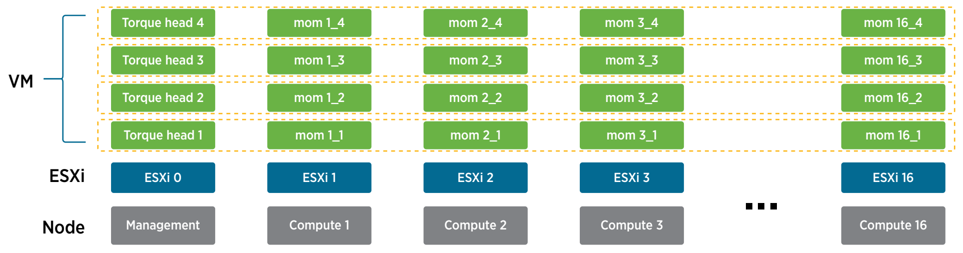

Figure 5. Four Virtual Clusters Share a Single Physical Cluster to Achieve 4X CPU Overcommitment

NOTE: Each dotted-line enclosure designates a separate virtual cluster.

Job Execution Time

To perform fair comparisons between test scenarios with differing numbers of active TORQUE clusters, we must ensure that the sequence of jobs that runs on the hardware in each scenario is approximately the same. To achieve this, we adopt a bottom-up approach by first generating a randomized job stream with each of the seven benchmarks repeated 116 times, for a total of 812 jobs. In the 4X overcommitment case in which four virtual TORQUE clusters execute jobs simultaneously, each of the clusters is fed a copy of this job stream, resulting in a total of 3,248 jobs being run over the course of this test. For the 2X overcommitment case with two active clusters, two copies of the 812-job stream are combined in an

[1] The TORQUE term for compute node manager.

interleaved manner to produce a new job stream containing 1,624 jobs. This 1,624-job stream is fed to each of the two active clusters, resulting in the same number of jobs run as in the 4X test case (3,248) and with jobs launched in approximately the same order. Finally, two copies of the 1,624-job stream are combined again to create a third job stream with 3,248 jobs, which is fed to the single active cluster for bare-metal and single–virtual-cluster tests. This approach ensures that the same job sequence is used in all test cases to ensure fairness.

Using the job streams previously described, the average execution time of three runs in different environments is shown in Figure 6. Because each node in our test bed has 20 cores, in the first experiment we configured each TORQUE MOM with 20 job slots. As reflected in Figure 6, the execution with one virtual cluster is very close to that of bare metal (first column), with only a 2.2 percent overhead. Furthermore, when multiple virtual clusters are used to achieve CPU overcommitment, the execution time is reduced, implying an improved throughput. Through careful analysis, we identified that the throughput improvement can mainly be attributed to increased CPU utilization when more jobs are concurrently scheduled to execute. This has been verified by modifying each TORQUE MOM in the bare-metal environment to use 40 job slots, after which the throughput is improved to the same level as the overcommitted virtual environment. This can be seen in the second column of Figure 6. We will analyze CPU utilization in the following section.

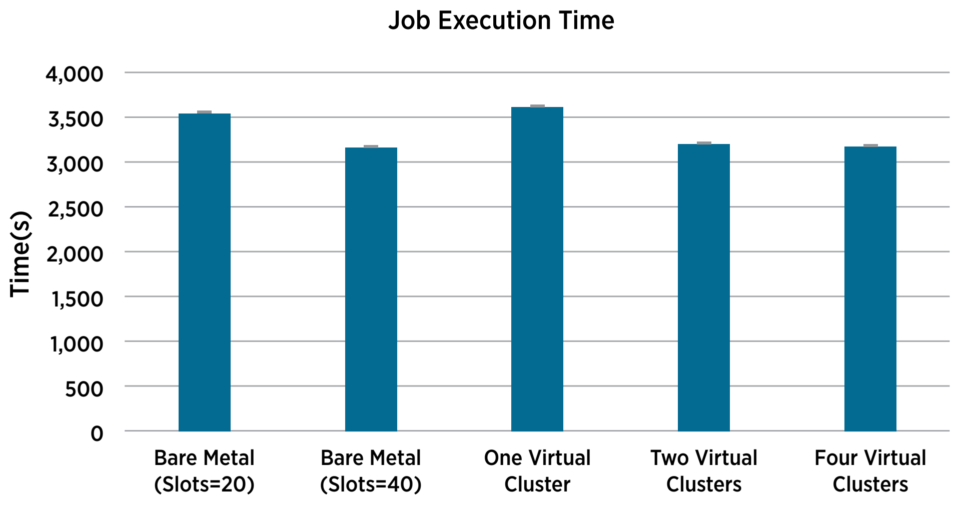

Figure 6. Comparison of Job Execution Time Between Bare Metal and Virtual

NOTE: Lower is better.

CPU Utilization

Besides execution time, another performance metric that we monitored is the total CPU utilization across the whole physical cluster. In an ESXi ssh session, the esxtop tool can gather a variety of metrics at fine granularity to give a detailed view of system state. We used esxtop running on each compute node to sample CPU utilization of all VMs at 5-second intervals, and the results for one, two, and four virtual clusters are shown in Figure 7. With two and four virtual clusters, the decreasing trend at the end is due to job completion.

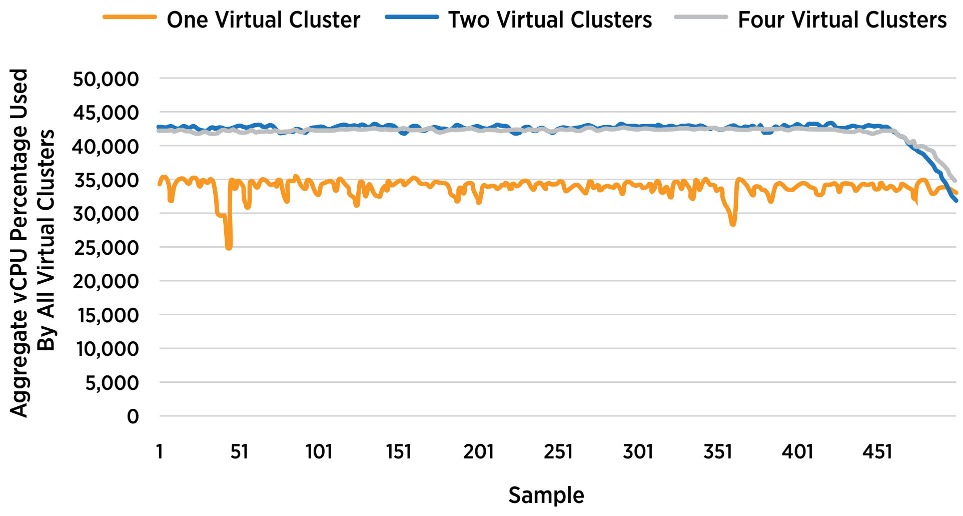

Figure 7. Total CPU Utilization Across 16 Nodes

Theoretically, the peak CPU utilization across the whole cluster can be calculated as 16 * 20 * 100 percent = 32,000 percent, given that there are 16 nodes and each node has 20 cores. However, Figure 7 shows that in all cases, the actual CPU utilization is well above the theoretical peak. Two factors contribute to the extra gains: Intel Hyper-Threading and Turbo Boost technologies. Because Hyper-Threading on average improves CPU utilization by about 25 percent, utilization values measured by the ESXi hypervisor are multiplied by 1.25 when reported by esxtop. Therefore, displayed peak utilization is increased from 32,000 percent to 40,000 percent. Furthermore, Turbo Boost frequently increases the CPU clock rate. In the case of two and four virtual clusters, Turbo Boost can increase CPU utilization to approximately 42,500 percent. The Turbo Boost effect has been verified by checking the esxtop power management panel, in which the %A/MPERF value indicates whether Turbo Boost is being used [4].

Clearly, the job execution times in Figure 6 are consistent with the CPU utilizations in Figure 7 for all the virtual cases. For example, the lowest CPU utilization case—one virtual cluster—matches the longest job execution time, and higher CPU utilization with two and four virtual clusters corresponds to shorter execution time. It is straightforward that the higher CPU utilization with two and four virtual clusters is due to resource consolidation brought about by virtualization. That is, when multiple virtual clusters share a physical cluster, the CPU scheduler on each ESXi host has more jobs to schedule and is therefore able to make better scheduling decisions by taking advantage of Hyper-Threading and eliminating idle cycles. At the same time, improved utilization comes with better consistency, where the utilization with two and four virtual clusters is much smoother than in the single-cluster case, as can be seen in Figure 7.

Per-Cluster CPU Utilization

In a real production HPC system, an important principle is fairness when multiple users are sharing the computing resources. This is also true in a virtualized environment. In particular, the previously mentioned CPU overcommitment configurations are representative of a future virtualized HPC environment where each user or group is given a virtual cluster during resource allocation. We can delve a little deeper into the two– and four–virtual cluster aggregate results in the previous section and examine the per-cluster CPU utilization in each case as shown in Figure 8. It is clear in both cases that the ESXi scheduler effectively maintains fairness so that each virtual cluster gets the same amount of CPU resources.

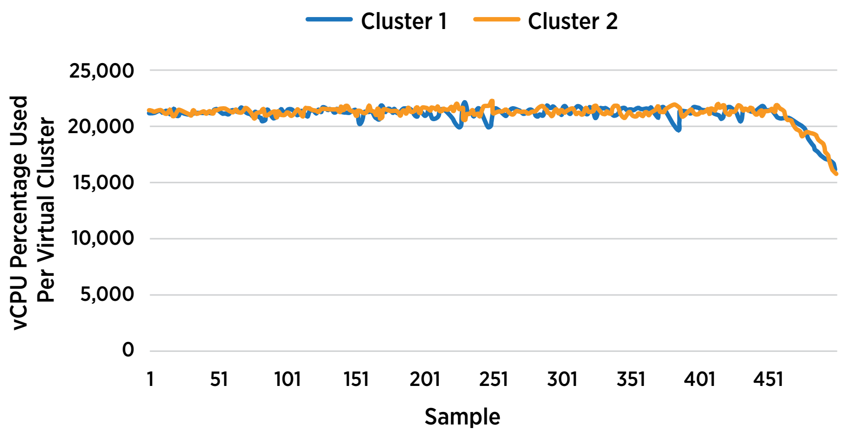

Figure 8a. Per-Cluster CPU Utilization: Two Virtual Clusters

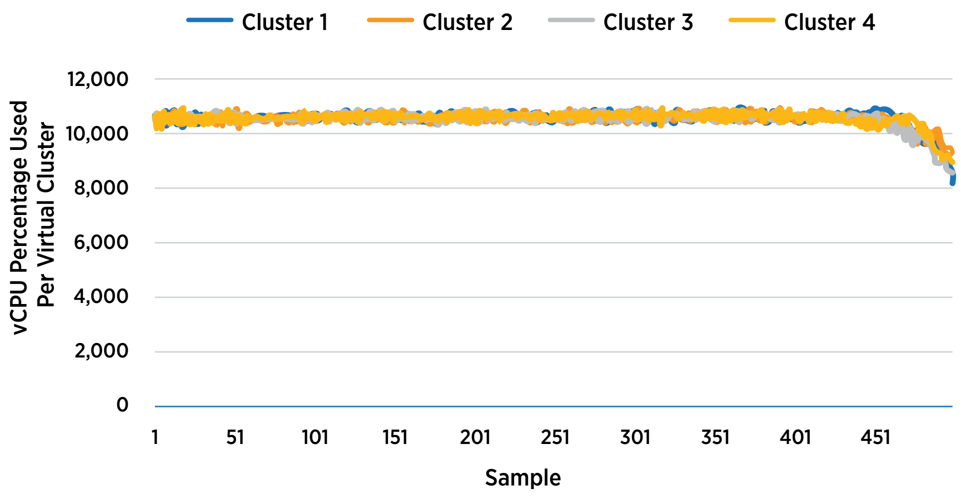

Figure 8b. Per-Cluster CPU Utilization: Four Virtual Clusters

CPU Overcommitment with Shares

In addition to the basic benefits of virtualization previously described, the proportional, share-based scheduler of the ESXi hypervisor offers another very useful degree of flexibility. This section continues the CPU overcommitment study by configuring virtual clusters with different shares, as can be done when creating a multitenant environment with quality-of-service guarantees.

With the use of CPU resource shares, each VM can be given a particular guaranteed share of CPU resources, which is difficult to achieve in a bare-metal system. When there are multiple VMs running on an ESXi host, the ESXi scheduler allocates CPU resources based on the ratio of shares among all the running VMs. For example, when there are two VMs, with shares of 30,000 and 10,000, the ESXi scheduler sees a 3:1 ratio and guarantees that the two VMs get 75 percent and 25 percent of CPU time, respectively. However, the ESXi scheduler is also work conserving in that if any VM does not fully use its shares, the other VMs are allowed to use more than their configured shares. This feature extends to a virtualized HPC system when it is composed of a cluster of ESXi hosts. Specifically, each user or group can be given an appropriate share of the physical system when their virtual cluster is allocated.

As a first experiment, we set the shares for the VMs in two virtual clusters to achieve a 3:1 ratio between the two VMs on each compute node. In this case, the average CPU utilization across three runs is 36,945 percent, which lies between the utilization level of one virtual cluster and that of two virtual clusters with equal shares. The total CPU utilization for both clusters as well as the per-cluster CPU utilizations from one run are shown in Figure 9. Recall that two virtual clusters with equal shares achieves the maximum CPU utilization. The lower CPU utilization in this unequal-share case is expected, because from time to time the scheduler must allocate exclusive CPU access to the higher-share virtual cluster to guarantee that it receives its configured CPU share. The exclusive access allocation essentially temporarily disables the effect of Hyper-Threading and therefore reduces the total CPU utilization. As a result of this, the actual measured CPU utilization ratio between the two virtual clusters is 2.8:1.

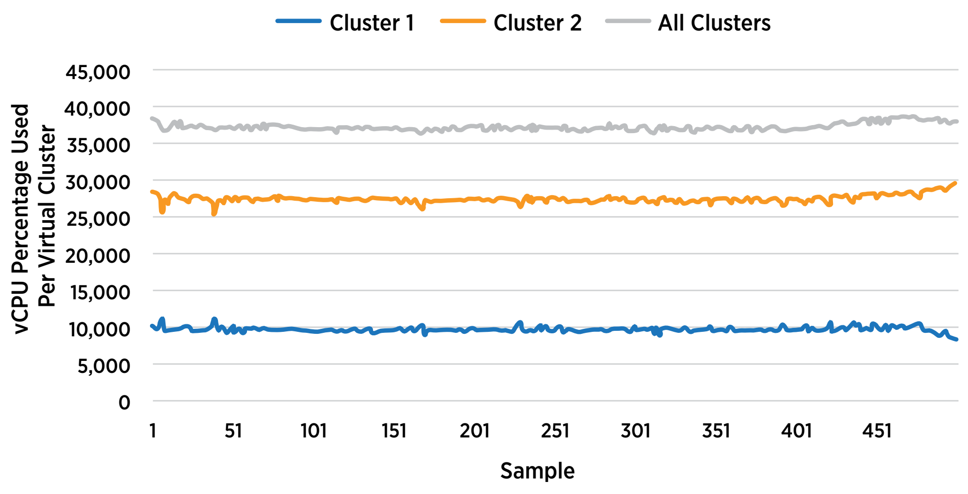

Figure 9. CPU Utilization for Two Virtual Clusters with 3:1 Shares

Lowered CPU utilization is undesirable and, by eliminating the need for the CPU scheduler to occasionally provide exclusive CPU access, it can be avoided while still allowing unequal shares to be used. To do this, consider that when two VMs with equal shares are each running a thread on the same physical core, they each use 50 percent of the core (reported as 62.5 percent due to the 1.25X utilization factor in effect when Hyper-Threading is enabled). When these two VMs are given shares such that the higher-share VM is entitled to more than 62.5 percent of the CPU resources, the scheduler occasionally will give that VM exclusive access to the CPU to ensure that it is receiving its promised share of CPU resources. In the case of the 3:1 ratio previously mentioned, one VM is promised 75 percent of the resources, which is greater than the 62.5 percent threshold. Therefore, occasional exclusive access is granted, resulting in slightly lower overall utilization of the physical core because only one hyper-thread is active during this time. However, if ratios are chosen such that no VM is given more than 62.5 percent of the CPU resources, exclusive access need not be granted. For example, when increasing the number of virtual clusters to four and giving them 2:1:1:1 shares, their respective CPU entitlements—considering the 1.25X utilization factor—are 50, 25, 25, and 25 percent. Because all of these entitlements are less than 62.5 percent, maximum overall CPU utilization is achieved, as is shown in Figure 10. Figure 10 further shows that the measured CPU utilization ratio among the virtual clusters is the same as the specified shares ratio.

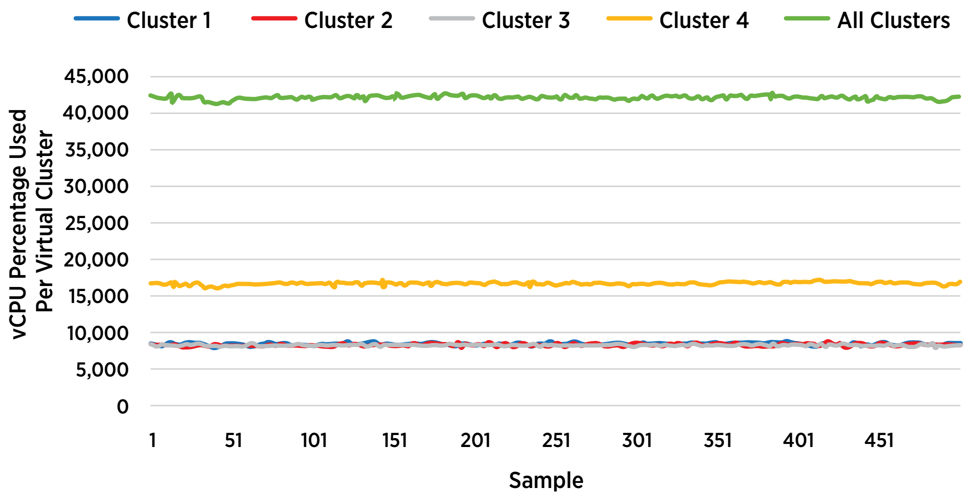

Figure 10. CPU Utilization for Four Virtual Clusters With 2:1:1:1 Shares

Summary

A rising trend toward virtualizing HPC environments is being driven by a motivation to take advantage of the increased flexibility and agility that virtualization offers. This paper explores the concepts of virtual throughput clusters and CPU overcommitment with VMware vSphere to create multitenant and agile virtual HPC computing environments that offer the ability to deliver quality-of-service guarantees between HPC users with good aggregate performance. Results from this work demonstrate that HPC users can expect similar to native performance for HPC throughput workloads while enjoying the various benefits of virtualization.

References

[1] VMware, Inc. *The CPU Scheduler in VMware vSphere 5.1* technical white paper. 2013.

[2] VMware, Inc. *Performance of RDMA and HPC Applications in Virtual Machines Using FDR InfiniBand on VMware vSphere* technical white paper. Na Zhang and Josh Simons. 2016.

[3] Dell, Inc. “Virtualized HPC Performance with VMware vSphere 6.5 on a Dell PowerEdge C6320 Cluster” blog. 2017.

[4] Yellow Bricks. “ESXTOP” blog. Duncan Epping.

Contributors

Michael Cui is a member of technical staff in the VMware Office of the CTO, working on virtualizing High Performance Computing. Previously, he was a research assistant and part-time instructor at the University of Pittsburgh, where his research was supported by multiple grants from the National Science Foundation and the U.S. Department of Energy. He holds both a PhD and a master's degree in computer science from the University of Pittsburgh.

Josh Simons is chief technologist for High Performance Computing, working in the VMware Office of the CTO, where he currently leads an effort to bring the value of virtualization to HPC. Previously, he was a distinguished engineer at Sun Microsystems, working on HPC; prior to Sun, he worked at Thinking Machines Corporation. Josh has a degree in engineering from Harvard College and a master's in computer science from Harvard University. He is currently serving on the OpenMP Board of Directors.