Virtualizing HPC and Technical Computing with VMware vSphere

CASE STUDY: JOHNS HOPKINS UNIVERSITY APPLIED PHYSICS LABORATORY

TECHNICAL WHITE PAPER

vmware®

Table of Contents

Introduction

Over the last decade and a half, virtualization has grown from a niche technology confined for use in test/dev environments to the de facto platform on which practically all enterprise data center workloads are run. It is also the foundation on which cloud computing is built. Better hardware utilization, reduced CapEx and OpEx, improved business agility, enhanced business continuity and security—these benefits are by now well understood and enjoyed widely.

In recent years, virtualization has started making major inroads into the realm of High Performance Computing (HPC), an area that was previously considered off-limits. In application areas such as life sciences, electronic design automation, financial services, Big Data, and digital media, people are discovering that there are benefits to running a virtualized infrastructure that are similar to those experienced by enterprise applications, but also unique to HPC.

The Johns Hopkins University Applied Physics Laboratory is one institution that has benefitted from adopting a virtualized infrastructure for its weapons simulation program. The results of using VMware vSphere® as the platform for their computer-intensive applications include:

- More than triple the average utilization of the hardware
- Reduced cost due to more effective resource sharing
- Ability to run a more diverse set of applications

It should be noted that this was achieved without compromising the manner in which the scientists manage and run their applications—in some cases, the performance of the applications was actually improved when virtualized.

Before describing the details of this case study, it is worth first reviewing the specific ways in which virtualization can significantly improve the realm of HPC. We will then see how Johns Hopkins University Applied Physics Laboratory arrived at the infrastructure design that met their particular needs.

Why Virtualize HPC

In this section, we discuss the particular aspects of virtualization that can greatly enhance the productivity of an HPC environment. We focus on the capabilities provided by VMware vSphere, as this platform provides a number of unique features above and beyond a generic virtualization platform. We specifically look at how virtualization addresses an important high-level goal: improved scientific productivity. Although the impact of virtualization on the runtime of a particular simulation or calculation can vary, the improvement in the overall effectiveness of a compute deployment is almost always extremely positive.

A Brief Overview of VMware vSphere

Although VMware produces virtualization products that are aimed at personal use on individual computers or laptops, the main platform for running workloads in data centers is vSphere. This product has a long history, and has evolved over time to optimize both hardware utilization as well as performance for a wide variety of application types. It is based upon a bare-metal virtualization architecture. The hypervisor, known as ESXi, is deployed directly onto server-class hardware. All virtual machines (VMs) run within carefully controlled and managed containers on top of this hypervisor layer. The hypervisor manages all access to the server hardware, including compute, memory, storage, and networking, and no guest VM has any privileged status.

When a VM is created, you specify the number of virtual CPUs and the amount of virtual memory. When the VM is running, the virtual CPUs are assigned to one physical core each on the physical CPU. The virtual memory is mapped to physical memory by the hypervisor. Oversubscription of resources describes a condition where either or both of the following occur:

- The total number of virtual CPUs across all running VMs exceeds the total number of cores across all physical CPUs on the host
- The total amount of virtual memory assigned to all running VMs exceeds the total amount of physical memory on the host

The ESXi hypervisor is designed to handle oversubscription very well, through various techniques for sharing memory and scheduling CPUs. For enterprise workloads, oversubscription is quite common and is a key part of the cost savings that virtualization provides. For HPC workloads, however, it is often more appropriate to avoid oversubscription and aim for a one-to-one mapping of CPU and memory between physical and virtual. We will see in the case study how this was approached for this particular environment.

As with a physical x86 server, a VM typically requires a boot device, such as a hard disk, and may also have additional disks for storing applications and data. In a virtual environment, these are provided as virtual disks. Each virtual disk is represented by one or more files that reside on an underlying file system, referred to as a datastore. The datastore can exist on a local hard disk of a physical host, or on a shared storage array. Every virtual disk file is associated with only one single VM at a time, and a VM cannot access the contents of another VM's virtual disk(s).

Another important concept is that of a virtual switch. A virtual switch behaves like a physical switch, except it exists entirely in software on an ESXi host. The host can have one or more virtual switches, and each switch can have one or more VMs connected to it. A VM itself can have one or more virtual network interface cards (NICs), and these can be connected to any of the virtual switches on the host. The virtual switches in vSphere support layer 2 features such as VLANs, thus allowing VLAN-based isolation between VMs on the same virtual switch. A virtual switch can use one or more network ports on the ESXi host as an uplink to a physical switch, which allows traffic to flow out to the rest of the datacenter. vSphere virtual switches are only layer-2 devices, meaning that if there are two virtual switches on one host that are connected to different physical subnets, traffic would need to go from one switch out to a physical router and back to the second switch in order for any communication to occur. It is also possible to configure a virtual switch without an uplink; this would mean that VMs on that virtual switch can only communicate with each other (provided that they are assigned to the same VLAN).

The operation of the virtual data center is managed by the component know as VMware vCenter Server™, an application that is itself typically deployed as one or more VMs. vCenter Server acts as the central point for tasks such as host configuration, VM deployment and operation, monitoring and alerting for events, and collection of statistical information.

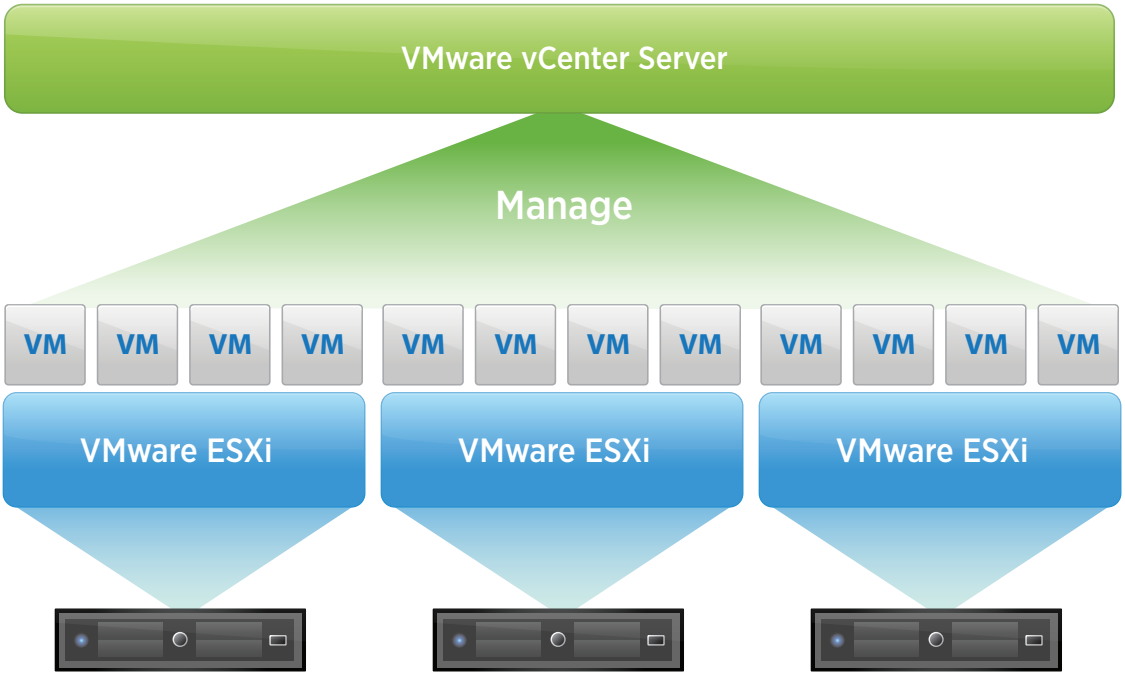

Figure 1: High-level architecture of VMware vSphere.

The two most fundamental characteristics of virtualization are:

1) Encapsulation: A VM is represented by a set of files that capture information such as boot disk and other data disks, the memory swap file, and virtual hardware configuration. Because vSphere provides a faithful representation of a standard x86 computer, the same files can be used to execute the virtual machine on any hardware. In addition, vSphere provides extensive backwards compatibility to older versions of virtual hardware, so that a VM created years prior can still be run on brand-new hardware, and it will behave exactly the same.

2) Isolation: Each VM operates totally independently of other VMs. The only way one VM can communicate with another is through whatever networking interface has been configured for it. Furthermore, the resources used by the VM are subject to strict control by the ESXi scheduler, which strives to allocate resources as efficiently as possible while adhering to consumption policies that will be described below.

Resource and Data Control

One of the challenges of any shared compute environment is how to prevent the various users from adversely affecting each other. Virtualization solves this problem for a number of specific concerns.

Resource Isolation

Because each VM's consumption of host resources is strictly governed by the hypervisor, it is not possible for one VM to take over a host and start stealing resources from other VMs. The hypervisor decides, for example, when a particular VM can use the CPU to execute instructions, and it maintains control over which memory pages are used by it. Furthermore, since each VM is executing in its own private container with its own virtual disk devices, any faults, such as the crashing of a daemon or system process, or filling up of temporary disk space, will affect only that VM and no other.

Data Isolation

When a VM starts, it typically uses a boot disk that is represented by a single file on some datastore accessible to the ESXi host that it's running on. It can also mount other virtual disk drives represented by other files. However, each VM is only able to mount the virtual disk files that have been designated for it. The VM is not able to see any other virtual disks, especially those belonging to other VMs. The result is that users from totally different groups are able to run jobs on the same cluster without worrying about data being snooped or leaked.

Security Isolation

One significant consequence of VM independence is much stronger security. The ESXi hypervisor does not treat any VM guest as privileged; furthermore, it is designed to not trust any privileged operations that the guest tries to perform. All such attempts are either intercepted and executed by the hypervisor on behalf of the guest according to the guest's configured rights, or else rejected. In its early days, this was done entirely through software using a technique known as binary translation. In later years, CPUs started including features that support virtualization by enforcing isolation and enabling memory management all in hardware. (Non-privileged instructions are allowed to execute directly on the hardware, which is the key to achieving performance comparable to running without virtualization.) The result is that it is extremely unlikely for an attacker with access to one VM to take control or cause damage to other VMs or the host itself, even if the attacker has root access. Malware that infects one VM (typically through the network) cannot spread within the host to other VMs. In addition, the virtual networking layer onto which VMs are attached can be configured with other security controls, such as virtual firewalls and layer-2 controls.

More Efficient Infrastructure Utilization

The ESXi hypervisor was initially designed to perform the core tasks of VM isolation and efficient x86 instruction execution. As it evolved, more resource management features were added, and in the current vSphere 6.0 release there is a very sophisticated set of resource management capabilities that allow fine-grained and effective management of compute, memory, storage, and networking resources. The relative share of these resources that a VM receives can be set differently for different VMs based on policy settings. With this prioritization ability, different groups can share the same compute cluster with fair-share allocation of resources according to their relative importance.

Another aspect of ESXi is its ability to efficiently schedule multiple VMs on the same host. The scheduler in ESXi has been shown to be very effective at this task, in many cases doing a better job than a single native OS instance managing the host. Benchmarks have shown that the superior CPU scheduling capability of ESXi can result in better-than-native performance of certain kinds of throughput applications, enabling batches of jobs to be completed more quickly than running native on the same hardware.

Beyond the management of resources on one host, one of the most powerful capabilities that revolutionized the IT industry, first introduced by VMware, is live migration of VMs from host to host. In addition to allowing hardware host management (such as patching, repairing, or even replacing) without requiring application downtime, live migration also enables dynamic load balancing of VMs across a cluster. VMware's version of live migration, called VMware vSphere® vMotion®, combined with VMware vSphere® Distributed Resource Scheduler™ (DRS), provides a powerful capability to ensure that all VMs get as much of the resources they are entitled to. For example, if the VMs running on one host start to experience too much CPU or memory contention, then DRS automatically decides which VMs to move to other hosts in order to relieve the pressure.

Built-In Application Services

Because a VM runs in software, it is possible to provide various services to the application that take advantage of this encapsulation. VMware vSphere® High Availability, one of the most popular features in vSphere, can be configured to automatically restart VMs on other hosts in a cluster if the host they are running on experiences a failure and goes down. For short-lived simulations or calculations, this provides a way to ensure that failed runs are automatically restarted without user intervention, thus allowing them to eventually be completed. For runs that are longer in duration, if the application is set up to checkpoint its state periodically, then the restart would simply pick up from where it left off, again allowing the run to be completed without requiring a user to restart it manually. This feature can greatly save time and administrator or user effort.

Another consequence of running in software is that VMs can easily be paused or suspended, and then resumed at a later time. When a VM is suspended, the entire contents of its memory is written to a file, so while it remains suspended it doesn't actual consume any physical memory (nor, of course, does it use the physical CPU). This capability can be used as a type of universal snapshot, permitting a job to be paused at any time and restarted right from where it left off. There is also a specific snapshot feature in vSphere that saves the state of the VM at a particular point in time and then continues running it. At any future point, the VM can be "rolled back" to the point of this snapshot, essentially enabling it to go back in time and run from that point as if nothing further had happened. There are many ways this feature can be utilized; for example, a simulation can be run up to a point, snapshotted, and then can be run repeatedly from that point with different input parameters or even different datasets.

Because VMs are encapsulated as files and virtual hardware behaves in a known way, the behavior of an application run is reproducible. This means that VMs can be archived for compliance purposes and rerun if needed. Even if the hardware or version of vSphere is different, it is guaranteed to run the same, because the virtual hardware representation would be the same.

Greater Flexibility in Infrastructure Usage

Perhaps the greatest benefit of a virtualized cluster to HPC is the flexibility for the same infrastructure to be used in many different ways. A typical physical HPC cluster runs a single OS version, and has only one set of application versions installed. It is not easy to rebuild the hosts with a different OS version, so this operation is rarely done. The users also have to worry about potential conflict among different applications running on the host, and if the user requires an alternate version of an application, they usually have no choice but to use a different infrastructure. The hosts themselves are sized with a certain chosen CPU and memory, and although there might be a mix of host sizes in a cluster, this mix is static and might not meet the needs of many researchers.

Contrast this with a virtual environment. A catalog of VMs can be kept on hand; each one can contain totally different software stacks, from the operating system to the application. The VMs can be right-sized to the specific simulation being run, with different virtual CPUs and virtual memory. A user can deploy whichever VM suits their needs, without worrying about conflicts. The user can even be given root access within the VM, without any worry of harming any other user. If a VM does somehow become corrupted, it's a simple matter of deleting it and starting with a fresh copy.

In one sense, this greater flexibility is so far ahead of physical HPC clusters in terms of capabilities that HPC resource management tools have yet to catch up with their enterprise workload management tools. For example, a typical HPC job scheduler assumes that the pool of compute nodes is immutable, and if no host matches the requirements of a job (e.g. CPU and memory size), it does its best to pick the closest match rather than request the provisioning of a new VM that exactly matches the requirements. There are various ways in which this can be addressed today, and we will see one approach in the Johns Hopkins Applied Physics Lab case study.

Performance of Virtualized HPC

If virtualization is to be used in HPC or Technical Computing environments, applications must run well when virtualized. Due to advances in the sophistication of virtualization software as well as significant and continued development of hardware support for virtualization, performance degradations due to virtualization have been much reduced from earlier versions of vSphere.

Throughput applications—single-process, possibly multi-threaded jobs—have been found to run on vSphere with well under 5% degradation, often just 1-2%. When many instances are run in parallel on a cluster or grid, job throughput can sometimes be higher in the virtualized environment. These results are consistent across a range of disciplines, including life sciences, digital content creation, electronic design automation, and finance. Figure 2 shows performance results for a representative throughput workload, the BioPerf benchmark suite.

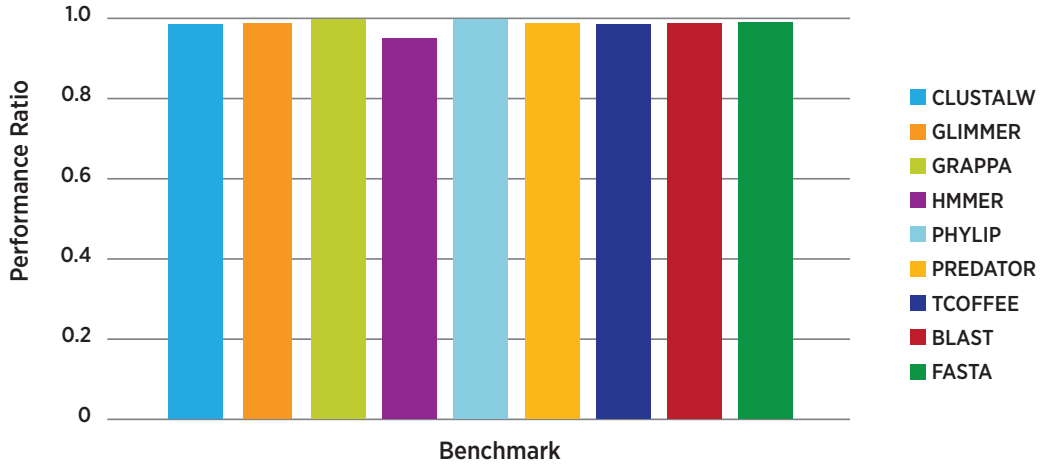

Figure 2: Performance of the BioPerf benchmark suite, showing the ratio of virtual to native performance. Higher is better, with 1.0 indicating that virtual performance is the same as native. HP DL380p G8 server, dual Intel Xeon E5-2667v2 processors, 128GB, VMware ESX 6.0.

Message Passing Interface (MPI) applications are more challenging due to their requirements for extremely low latency. With continued engineering improvements driving latency overheads relative to native performance toward zero (see figure 3), vSphere is able to deliver near-native performance for an increasing range of MPI applications. Testing with a variety of molecular dynamics, chemisty, and weather codes, yielded overheads under 10% (and very often under 5%) for 128-way MPI configurations running across eight InfiniBand-connected nodes.

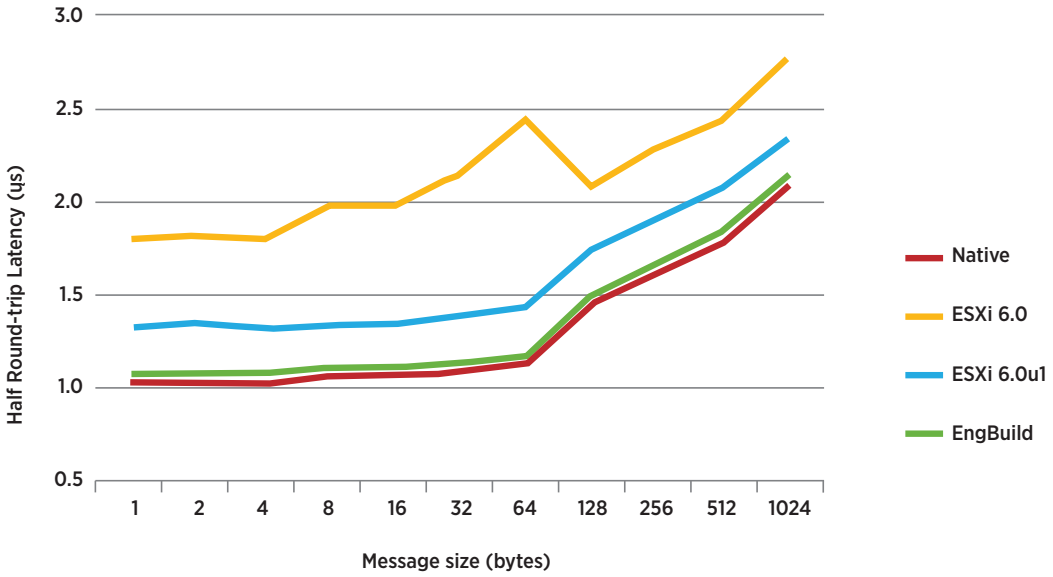

Figure 3: MPI small-message latencies over FDR InfiniBand, comparing native performance to that of several versions of VMware ESXi hypervisor. Lower is better. ESXi 6.0 and ESXi 6.0u1 are currently shipping versions of ESXi. EngBuild is an internal engineering build.

Figure 4 shows performance results for eight-node runs of Weather Research and Forecasting (WRF), a popular weather forecasting and research code, which demonstrate the benefits of removing latency overheads from the virtual platform.[1] As higher-scale testing and testing with even more demanding applications progresses, we will continue to identify and remove platform overheads to embrace an even wider array of MPI applications in the future. (See https://blogs.vmware.com/cto/tag/hpc/ for more details.)

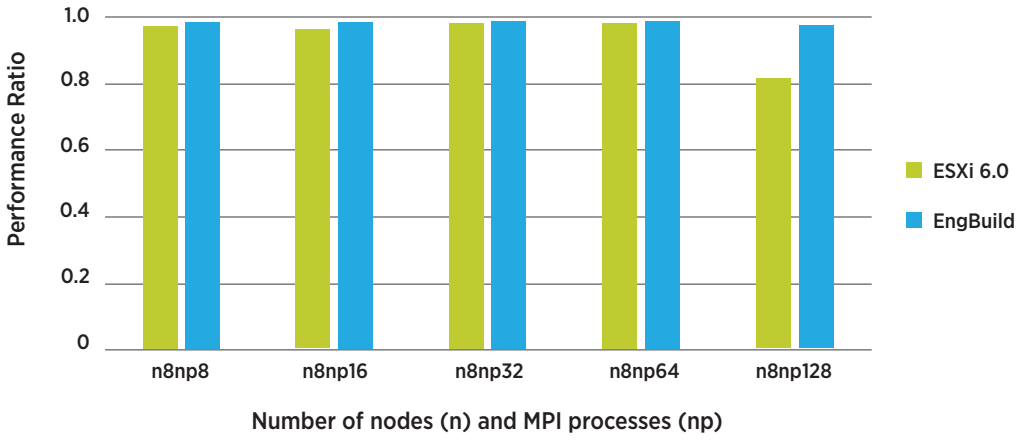

Figure 4: Comparison of the relative performance of WRF using ESXi 6.0 and an internal engineering build. Performance ratios are shown, and higher is better, with 1.0 indicating equal virtual and native performance.

1. The benchmark problem (conus 2.5km) is run five times, each test on an eight-node, FDR-connected cluster of HP DL380p G8 systems, varying the number of MPI processes from 8 to 128.

Johns Hopkins University Applied Physics Laboratory Case Study

For more than 70 years, the Johns Hopkins University Applied Physics Laboratory has provided critical contributions to critical challenges with systems engineering and integration, technology research and development, and analysis. Their scientists, engineers, and analysts serve as trusted advisors and technical experts to the government, ensuring the reliability of complex technologies that safeguard our nation's security and advance the frontiers of space. They also maintain independent research and development programs that pioneer and explore emerging technologies and concepts to address future national priorities. Their four main sponsored areas of work include air and missile defense, asymmetric operations, force projection, and space science.

Status Quo Prior to Virtualization

At the time when this case study takes place, the application landscape at the lab consisted of 16 unique Monte Carlo simulations. These jobs are mostly CPU intensive and embarrassingly parallel, with about 2 GB RAM per task and only moderate networking needs. The applications are split between Linux and Windows environments.

In this environment, the lab experienced the most typical problem of physical HPC clusters. The environment was sized to meet peak demands, which was expensive and led to many idle cycles. In addition, the hosts were statically provisioned and evenly split between Linux and Windows. If the demand for one type of operating system was greater, there was no way to borrow resources from the other half. There was also a problem specific to the Windows hosts. Often, the configuration of the hosts would drift, and so this lack of consistency meant the jobs did not run reliably.

The Solution: vGrid

After researching the capabilities of vSphere, the IT staff of the lab decided to try it out in their environment. The way they approached this was very methodical, and can be considered a model for how to adopt virtualization in an HPC environment. The resulting environment was given the name "vGrid."

The staff was concerned about right-sizing the VMs. As is the usual approach for HPC, they wanted to avoid oversubscription of CPU and memory. However, they needed to choose between the extremes of many small VMs on a host versus a few larger VMs. To determine the best choice for their environment, they ran a series of benchmarks. What they discovered was that VMs performed best when the number of virtual CPUs could fit exactly into all the cores on a single socket (i.e. single CPU). In fact, the simulations ran on average 4% faster than if they were run natively on the hardware. Figure 5 shows a comparison of run times running virtual versus running native on physical hardware. This result is sometimes seen when benchmarks compare performance on vSphere with native (non-virtual) for embarrassingly parallel, throughput applications. Because the ESXi scheduler is optimized to take advantage of NUMA locality, it can frequently do a better job of scheduling VMs than an operating system scheduling an application natively.

CONFIGURATION	CORES PER VM	NUMBER OF VMS	GB OF RAM PER VM	RUNS	TIME IN MINUTES	PERCENT FASTER
VM Configuration	16	2	32	32	20:38	**4.32%**
Bare Metal	32	N/A	128	32	21:28	

Figure 5: Comparison of run times for jobs running native on physical hardware versus running virtualized on vSphere.

The lab settled on the use of VMs with 16 virtual CPUs each. For Windows VMs, each was assigned 32 GB of virtual memory, while Linux VMs were assigned 16 GB. These sizes were chosen so that the VMs would fit within the configuration of the physical hosts that were used for this environment. When it came to deploying the VMs the lab decided that, for this phase of the project, they would statically provision a set of VMs that would be permanently up and running, able to run jobs at any time. They provisioned two Windows and two Linux VMs per host. Although they didn't oversubscribe memory, they did oversubscribe the CPUs at a 2:1 ratio. The reason for this is that, in their operating model, the Linux and Windows VMs are never used at the same time. For example, when a batch of Linux simulations is run, they occupy all the Linux VMs, while the Windows VMs simply remain idle and do not consume any CPU cycles.

Because all VMs are provided exactly as much memory as they are assigned, ESXi memory management techniques don't have to take effect and handle memory oversubscription.

For the virtual disks, the lab decided to assign one physical hard drive per VM to avoid contention. Each host has four hard drives, and each hard drive is formatted as a single datastore. Each VM is then given its own dedicated datastore, and each has one virtual disk on the datastore that is used primarily for the operating system, applications, and temporary storage. The VMs are able to access simulation data on a network file share, exactly how it would be done with a physical compute node.

For networking, each VM was configured with three virtual NICs, connected to three virtual switches. These three networks handled the following traffic:

1. Access to the network storage
2. Communication with the Grid Engine scheduler
3. All other network traffic

The three virtual switches each had their own 1 gigabit ethernet physical NIC on the host, which was connected to the appropriate physical switch. A fourth physical NIC is used by the ESXi host management network, which is how it communicates with the vCenter Server management virtual machine.

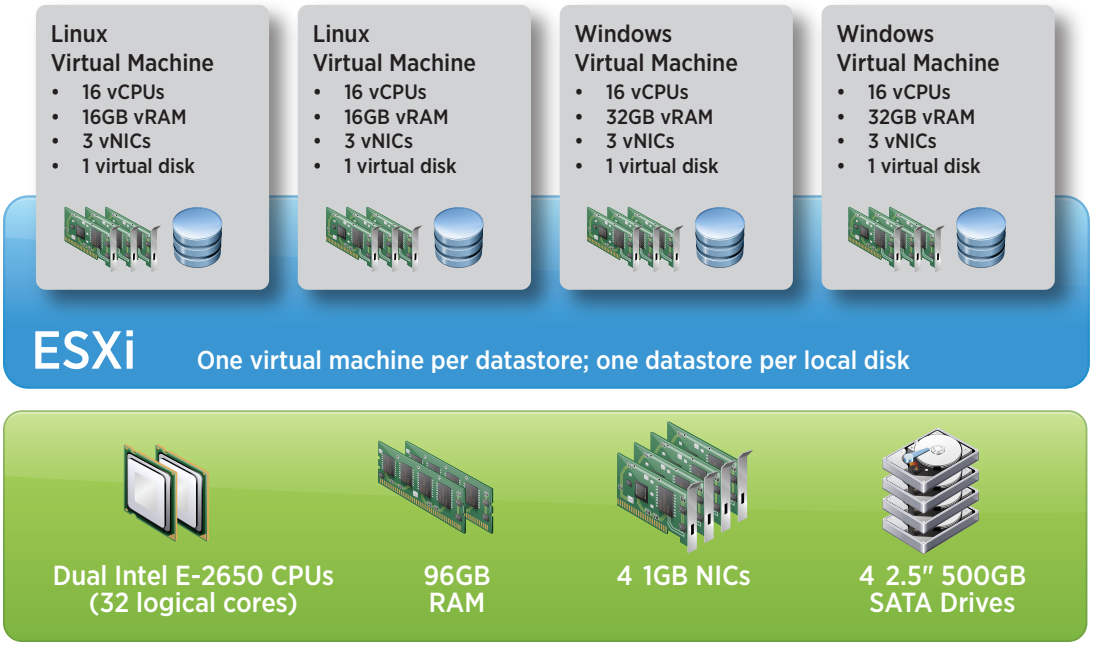

Figure 6: Configuration of physical hosts and VMs.

Each project has its own network-attached storage array. The types of simulation dictate the storage access patterns. For some simulations, there are millions of short-lived tasks, and results are written at the end of each task. For this type of job, the data is first copied from the array to the local virtual disk of the VM, and all intermediate results are read and written from this local disk. This is done to avoid oversaturating the storage network. At the end of the job, the final results are written back to the storage array in a batch. Other simulations run much longer, and they write intermediate results much less frequently. For these, the data can be read and written directly from and to the storage array during the course of the simulation, as this I/O occurs less frequently and hence does not risk oversaturating the storage network.

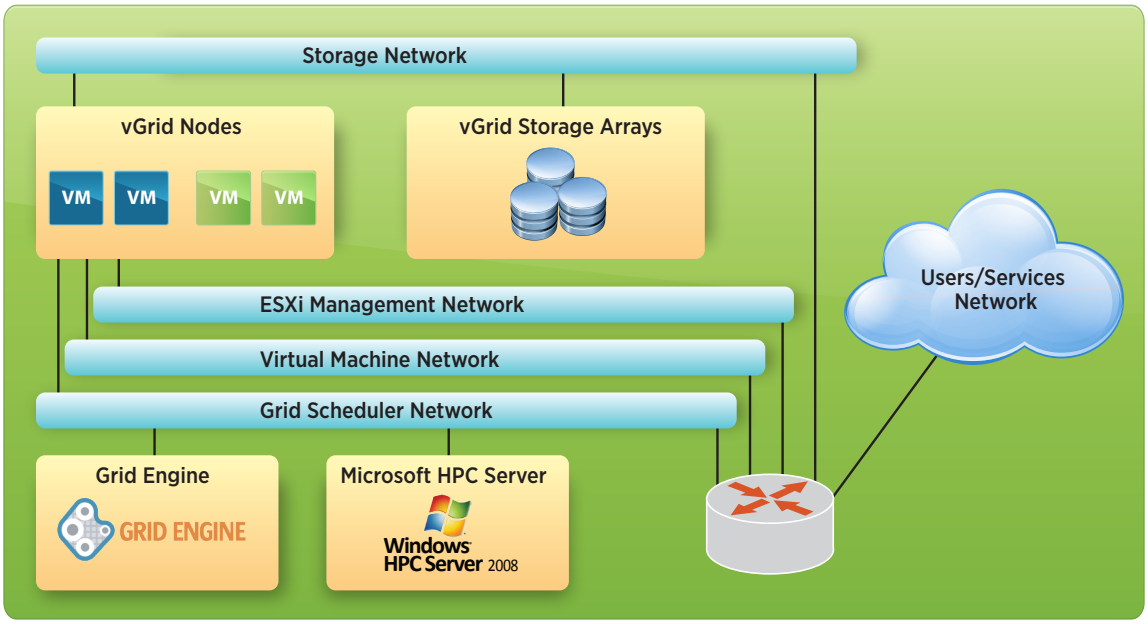

Figure 7: Architecture of compute environment.

Results of Implementing vGrid

The final cluster deployment, as described in figures 6 and 7, had a total of 3728 cores dedicated for computational tasks. By consolidating all servers into a shared cluster and converting it to a vSphere environment, the Applied Physics Lab saw the following benefits:

- Overall utilization of the hardware more than tripled, going from 9.1% to 29.2%
- Simulations ran on average 4% faster compared to running natively on the hardware
- The operating system and application versions could be changed easily

Grid Engine was used for Linux job scheduling prior to implementing vGrid, and continued to be used after virtualizing the environment. The only difference is that jobs are scheduled onto virtual compute nodes, and this happens totally transparently to Grid Engine. This meant that the scientists didn't have to learn a new way of submitting and managing their jobs. For Windows job scheduling, the laboratory implemented Microsoft HPC Server, which works similarly to Grid Engine for Linux, i.e. the fact that the compute nodes are VMs is transparent to the scheduler.

The Applied Physics Lab is considering ways to expand upon the success of vGrid. The addition of a service catalog of VM images along with a self-service capability would greatly enhance the scope of possible applications that can be run in the cluster. It would allow the simultaneous use of as many software stacks as are pre-created, and users would be able to select whichever one they want without requiring any intervention from the IT administrators.

Conclusion

Virtualization can be applied to greatly improve the productivity of HPC environments. The features and capabilities of VMware vSphere have the potential to reduce costs, improve resource control, increase utilization, and enable wider usage of the compute environment by providing much greater flexibility in usage. The Johns Hopkins University Applied Physics Laboratory was able to successfully implement virtualization in their HPC compute cluster. Their case study provides a good example of how a methodical approach to designing and architecting a virtual compute grid can be used to gain significant and measurable benefits.

Contributors

Josh Simons currently leads an effort within VMware's Office of the CTO to bring the full value of virtualization to HPC. He has over 20 years of experience in High Performance Computing. Previously, he was a Distinguished Engineer at Sun Microsystems with broad responsibilities for HPC direction and strategy. He joined Sun in 1996 from Thinking Machines Corporation, a pioneering company in the area of Massively Parallel Processors (MPPs), where he held a variety of technical positions. Josh has worked on developer tools for distributed parallel computing, including language and compiler design, scalable parallel debugger design and development, and MPI. He has also worked in the areas of 3D graphics, image processing, and real-time device control. Josh has an undergraduate degree in Engineering from Harvard College and a Masters in Computer Science from Harvard University. He has served as a member of the OpenMP ARB Board of Directors since 2002.

Edmond DeMattia is a multi-disciplined Senior Systems Engineer at Johns Hopkins University Applied Physics Laboratory. His background is in virtualization architecture and high-performance scientific computing. Edmond is currently the technical lead for virtualized cyber modeling and simulation projects that are designed to ensure the security and effectiveness of the nation's nuclear assets against cyber vulnerabilities in the Nuclear Command and Control Communications (NC3) systems.

Charu Chaubal is the Director of Technical Marketing in the Cloud Platform business unit at VMware. He has been at the company for over nine years, and has managed teams responsible for customer education and sales enablement for various technologies such as hypervisor security, virtual networking, hyperconverged storage, and Big Data. Previously, he worked at Sun Microsystems, where he had over seven years of experience with architecting distributed resource management and grid infrastructure software solutions. He holds several patents in the fields of datacenter automation and numerical price optimization. Charu received a Bachelor of Science in Engineering from the University of Pennsylvania, and a Ph.D. from the University of California at Santa Barbara, where he studied theoretical models of complex fluids.

VMware, Inc. 3401 Hillview Avenue Palo Alto CA 94304 USA Tel 877-486-9273 Fax 650-427-5001 www.vmware.com