AC 2011-1231: EXPLORING THE USE OF VIRTUAL MACHINES AND VIRTUAL CLUSTERS FOR HIGH PERFORMANCE COMPUTING EDUCATION.

Thomas J. Hacker, Purdue University, West Lafayette

Exploring the Use of Virtual Machines and Virtual Clusters for High Performance Computing Education.

Abstract

High performance computing systems have been based on commodity computing hardware since the introduction of Beowulf systems in the mid-1990's. The emergence of virtualization and cloud computing technologies now make it possible to build high performance computing cluster systems across a collection of virtual machines. This paper will explore the pedagogical and technological issues involved in the use of virtualization and cloud computing technologies for HPC education, focusing on: mixed use of physical and virtual computing environments; high performance networking fabrics; pedagogical limitations of virtual and cloud computing; the development of an effective teaching laboratory for virtual clustering; and the performance and reliability constraints of a mixed virtual cluster environment. The paper will describe the use of virtualization software, specifically Xen, OpenVZ, and VMware, and an assessment of the viability of the Eucalyptus, NIMBUS, and OpenNebula cloud computing systems for use for virtual clusters for HPC education.

1. Introduction

The emergence of virtualization technology over the past five years is sparking a new revolution in computing. This revolution has led to greater efficiencies in the use of computer servers and is one of the driving technologies behind the development of cloud computing. A similar trend a decade ago exploited the availability of low cost and high performance commodity computer hardware and open source software to power the development of commodity component based high performance computing systems. Today, these systems dominate the list of the 500 most powerful supercomputers in the world [1].

The recent availability of virtualization technologies motivates the exploration of several questions about the applicability of these technologies to high performance computing. First, how can high performance computing systems make the best use of virtualization and cloud computing technologies? Second, what new capabilities, services, and pedagogical approaches do these new technologies now make possible in the area of high performance computing education?

This paper explores these questions, and discusses the issues involved in the adoption of virtualization and cloud computing technologies for HPC education, and describes experiences and lessons learned from the use of four virtualization and cloud computing systems: Eucalyptus, OpenNebula, NIMBUS, and VMware.

2. Virtualization and Cloud Computing for HPC

High performance computing systems in use today are predominately based on the Linux operating system and commodity hardware components. Virtual machines provide the same types of components used in high performance computing systems and support many types of operating systems. In this section, we provide an overview of virtualization and cloud computing technologies. The section following this section explores the technological and pedagogical issues involved in the use of virtualization and cloud computing for high performance computing.

2.1 Virtualization Technologies

There are two basic types of virtualization technologies in use today that are available as commercial and open source packages. The first type, hypervisor based virtualization systems, operates on the "bare metal" computer hardware, and is designed to provide a operating system environment that can host one or more guest operating systems within virtual machines. A *Type 1 hypervisor* is an operating system image designed to only run guest VMs. A *Type 2 hypervisor* is a software layer that runs within the operating system image to allow users to both use the base operating system as well as run virtual machines [2]. The second type of virtualization is *paravirtualization,* in which the guest operating system has been explicitly modified to run as an application within the host operating system, rather than running on a virtual machine monitor without awareness of the underlying virtualization system.

The three most popular hypervisors in use today are VMware, Xen, and KVM. VMWare is a commercial virtualization system [3] that consists of a hypervisor (named *ESX*) that runs on physical nodes, a management system (named *vSphere*), and a suite of products for VM and virtual network configuration and management. Xen [4] is an open source hypervisor system that has been in use for nearly a decade. Xen can operate as a hypervisor on Intel or AMD processors, and provides a simple command line interface to create, stop, restart, suspend, and migrate virtual machines on a physical host. Another more recent open source hypervisor system is the *Kernel-based Virtual Machine* (KVM) [5]. KVM is supplanting Xen as a frequently used virtualization system, due in part to its inclusion in the Red Hat 6 distribution. Similar to Xen, KVM provides functionalities to create, stop, restart, suspend, and migrate virtual machines. A major difference between KVM and Xen is that KVM relies on a graphical GUI, the *virt-manager*, for managing virtual machines. Another difference is that KVM is integrated with an open source processor emulator, *QEMU*, [6] to provide support for a broad range of processors, and relies on *libvirt* to provide a comprehensive command line interface (CLI) [7]. *Libvirt* is a veneer layer virtualization application programming interface (API) and CLI that provides a common control library and management system that can interoperate with several virtualization systems. Libvert currently supports Xen, QEMU, VMware, Virtual Box, and several other virtualization systems. To provide virtual networking support, Xen and KVM rely on a virtual ethernet bridge running on the Linux system that is managed using *brctl* [8].

The differences among these virtualization systems can be compared in several areas: stability and ease of use; level of technical support; integration with the Linux kernel and system; and programmability within Linux. VMware is the most mature and stable, and is easy to use through a system and laboratory level graphical user interface based management system. The ESX hypervisor is stable and is based on a Linux kernel. However, controlling a VMware hypervisor through the command line interface is complex, and the ESX hypervisor programming environment is limited, which makes it difficult to develop a rich and comprehensive set of customizations for the VMware environment. In contrast, the Xen hypervisor operates within the familiar Linux environment, since Xen is layered on the kernel as an addition to Linux. The hypervisor environment, which runs in Xen *Domain-0*, can be easily customized, and access to Xen virtualization functionalities is straightforward. The downside to Xen is that it can be unstable, especially when performing complex operations such as live migration among physical hosts. KVM, the newest of the three, has greater stability and is more up-to-date than Xen, due to the integration of KVM into Red Hat Linux Version 6. KVM lacks a native management interface, but can be controlled through the libvirt *virsh* command line interface. Adopting a virsh based management approach may be the wisest choice in the long run, because libvrirt seeks to develop and support a common veneer interface over many different virtualization systems.

As an alternative to hypervisors, there are several popular *paravirtualization* systems available today. OpenVZ [9] is a lightweight virtualization system that allows an operating system image to run several guest operating system instances (named *containers*) that share the operating system kernel with the host operating system. Rather than storing the guest VM image as a large virtual disk file, the file system within an OpenVZ container is a subtree within the host operating system file system. This greatly simplifies the process of customizing and managing OpenVZ images on a system. OpenVZ provides a set of management commands though a command line interface, and supports the creation, stopping, starting, checkpointing, and live migration of OpenVZ containers. The live migration functionality of OpenVZ is much faster than the live migration operation for hypervisor based systems, since only a fraction of the file system and memory must be transferred between physical hosts, rather than the entire virtual disk image as is the case for hypervisor based systems. OpenVZ is used to provide software-as-a-service (SaaS) for the popular HubZero [10] science gateway and is used in the NEEShub for the National Science Foundation NEES project.

Another popular virtualization package is *VirtualBox*[11], which is a paravirtualization system that runs on a wide variety of platforms, and provides a simple GUI based management interface. Virtual Box is especially useful for creating base virtual machine images for import into Xen or KVM. The ability of VirtualBox to quickly create and export a VM image is especially useful for creating VM images for use in Xen or KVM. This is due in part to the cumbersome process necessary in Xen and KVM to create an initial VM image, which requires placing the DVD image on a network attached NFS server or web server.

One major difference between VirtualBox and OpenVZ is that live migration in OpenVZ is much simpler than VirtualBox, due to the need to provision a target VM container for VirtualBox prior to live migration. Also, VirtualBox provides an administration GUI, where OpenVZ for Linux only provides a command line management interface.

2.2 Cloud computing technologies

Cloud computing is emerging as the latest generation of distributed computing technology for this decade, and is an extension of the utility computing concept first developed in the 1960's [12] to a broader class of applications beyond scientific computing. Cloud computing is a broad term that encompasses several technologies: Infrastructure-as-a-Service (Iaas), which involves the use of virtualization to create and manage computing infrastructure; Platform-as-a-Service (PaaS), which builds on a virtualization platform to provide a software stack (e.g. web server, middleware, and authentication) in addition to a virtualization service; and Software-as-a-Service (SaaS), which allows users to run software on a remote server through a graphical interface without the need to download and install the software on their local computer. The high performance computing community uses one or more of these types of cloud computing today in various forms. For example, the *NEEShub*, a science gateway for the NSF NEES earthquake engineering community, uses SaaS to allow users to run tools within the hub environment, with interaction controller through an embedded VNC window. The NEES project also uses IaaS in the form of VMware virtual machines that provide a secure development sandbox for NEES software engineering and users.

In the context of developing virtual clusters, Infrastructure-as-a-Service is the most relevant type of cloud computing. Generally, IaaS systems provide a higher level of management above the individual system level for collections of physical systems and virtual machines running in a finite set of physical computer nodes. There are several types of IaaS cloud computing packages available today that can be used to develop a virtual high performance computing cluster. As is the case in virtualization technology, there are commercial and open source cloud computing packages available to develop IaaS: Eucalyptus, Nimbus and OpenNebula.

In the commercial realm, Eucalyptus has been available for several years [13] in both a commercial version and as an open source package. Eucalyptus is an IaaS system that allows users to create and manage virtual machine instances among a group of physical computing systems. Eucalyptus can manage virtual machine images across Xen and KVM hypervisor based virtualized systems. Eucalyptus components include a *node controller*, to manage VM instances; a *cloud controller*, which manages sets of nodes and clusters; and a *cluster controller*, which manages the hypervisor and virtual networking among the physical nodes. Another commercial IaaS is Amazon EC2 [14], which is a completely commercial service for which users can purchase metered IaaS and PaaS services. The major difference between these two commercial approaches is that when using Eucalyptus, the user provides the computing hardware

and purchases the cloud computing software; and when using Amazon EC2 services users pay for both computing infrastructure and software.

As an alternative to potentially expensive commercial cloud computing services, there are several open source cloud computing packages that can be used with a site's existing physical computing, storage, and networking infrastructure. The first package, described above, is Eucalyptus. In contrast to the commercial version, the open source version of Eucalyptus is not as well supported and does not support VMware, Windows, and several other features needed to run Eucalyptus within an enterprise. The second open source package is Nimbus [15], developed by Argonne National Laboratory in 2008. Nimbus is an IaaS package that is an open source toolkit that manages the allocation and deployment of VM images across a set of computational nodes. Nimbus is designed to interoperate with Globus based grid computing systems [16], and is focused on aiding science communities in the deployment of customized VM images to support scientific applications [17]. Nimbus supports Xen and KVM, but does not support VMware or OpenVZ. The third package is OpenNebula [18], developed at the Universidad Complutense de Madrid in Spain. OpenNebula is an open source set of tools for managing a set of hypervisors, virtual machine images, and networks to create an IaaS service. In contrast to Eucalyptus and Nimbus, OpenNebula is based on common Linux services and utilities such as NFS, scp, and rsync; and supports the Xen, KVM, and VMware virtualization systems by using the libvirt libraries and utilities.

In practice, the differences between Eucalyptus, Nimbus, and OpenNebula are in: ease of installation, troubleshooting, and use; degree to which users and administrators can control and customize the environment; and stability. Of the three, Nimbus is most difficult to install and configure, due to its integration with grid computing technologies. The process of installing and configuring OpenNebula is the simplest of the three, since it can leverage existing familiarity with Linux services and troubleshooting practices common to Linux. In terms of configurability, OpenNebula is easily customized – users and administrators can easily modify OpenNebula scripts and tools. Finally, in the area of stability, our personal experience with testing all three systems is that OpenNebula proved to be remarkably resilient and persistent despite frequent system outages and network configuration changes. We found Nimbus to be the most brittle and intolerant of variations in virtual machine images, and although Eucalyptus worked well with certified VM images, it was difficult to configure and keep Eucalyptus running as we changed VM images and the network configuration.

3. Technological and pedagogical issues involved in the use of virtualization and cloud computing technologies focusing on HPC education

The availability of virtualization and cloud computing technologies and the adoption of these technologies for general practice motivate the investigation of their use for high performance computing. To effectively use these technologies, several fundamental issues must be addressed. First, what are the effects of the mixed use of physical and virtual computing environments on

the development and use of virtual clusters based on this technology? Second, what changes in approach are needed to introduce virtualization and cloud computing technology into HPC education? Third, what are the pedagogical limitations of virtualization and cloud computing that must be taken into account in these changes? Finally, how can we best evolve a physical computing laboratory to include these technologies without negatively impacting the learning experience?

3.1 Mixed use of physical and virtual computing environments

The use of physical computer hardware, in the form of desktop computers or server class systems, makes it straightforward to build on the students' knowledge of computer architecture and operating systems to introduce topics related to the design of commodity based high performance computing clusters. For example, a lecture on deducing the I/O bottlenecks in a motherboard design and its effects on data intensive applications can build on students' personal experiences with their own computers and video game systems. The use of virtual machines, however, adds many new differences in the behavior and performance characteristics of operating systems and applications running within a virtual machine environment.

Physical and virtual computing environments are fundamentally different in many ways. First, there are obvious differences in the number of CPUs and amount of memory for physical systems versus a virtual machine. There are also much more subtle differences. For example, when deciding which physical RAM to use for an HPC cluster node, it is important to match the memory striping factor and DIMM clock speed with the processor front side bus speed to ensure that you don't over invest in memory that is faster than the front side bus speed, but also not under invest in high speed memory and consequently create a structural performance bottleneck in the system. Virtualization systems provide no context for this information when creating a new VM – all of the specific architectural details are abstracted away and hidden from the user. There are also many subtle differences between the physical and virtual machine environment. For example, modern AMD and Intel processors support a fault reporting feature called *Machine Check Exception* (MCE) that provides an interrupt to the operating system when the processor encounters a soft correctable error or a hard error. Virtualization systems tend to explicitly mask this information from guest operating systems. In VMware, for example, the hypervisor will confirm the existence of MCE banks to the guest VM when it boots and queries the processor capabilities, but indicates that it supports no MCE banks when the kernel tries to access them. Subtle differences such as this example can complicate the process of exploring and using hardware features of the physical system.

Another fundamental difference between virtual and physical systems is in the approach to systems management. For physical systems without virtualization, the mapping of host to service and operating system is clear, and hardware features such as network attached Baseboard Management Controllers (BMC) makes it simple to remotely perform administrative tasks. In contrast, virtualization technologies make it possible to operate many VM images on a physical

system. In this environment, managing the mapping between service and hardware becomes much more complex, and significantly more effort is needed to configure and maintain operating system versions and systems security. Managing network connectivity and security is also greatly complicated by virtualization. The ability to create virtual switches that can be linked among systems using VPNs and VLANs, as well as the ability to create multiple virtual network adapters within VMs that can attach to separate networks creates a tremendous opportunity for novel approaches to connectivity among VMs, but also can quickly become a significant security vulnerability.

These inherent differences between physical and virtual affect the basic approach in which an HPC cluster is designed and loaded.

In terms of design, the cluster designer needs to ensure that the mapping of cluster node VMs to the physical hardware does not overload the physical CPUs, and to avoid under utilizing the physical resources – finding the appropriate balance among these depends on the characteristics of the application. I/O intensive applications encounter frequent CPU stalls, so it will be possible to put more virtual machines on a physical server than would be advisable when supporting a CPU intensive parallel application. The memory footprint used by a VM is another factor – the amount of memory needed by a parallel application depends on the individual characteristics of the application and the problems size the user seeks to solve. Determining the memory size needed for an application requires an assessment of application performance and memory use as the size of memory in the VM is varied. Over-provisioning memory in a VM could potentially waste memory, and under provisioning could cause page swapping that may slow the entire system. Finally, network latency and bandwidth is another critical factor. The performance of some types of applications, such as molecular dynamics (MD) and computational fluid dynamics (CFD), strongly depend on network performance. The additional latencies added among physical systems could seriously degrade performance. Conversely, significant gains in performance possible when communicating among VMs running on a single physical system through a virtual ethernet switch can provide an excellent testbed to assess and demonstrate the effect of latency and bandwidth on the performance of different categories of parallel applications.

The price/performance considerations traditionally used for designing cluster from physical computer nodes also fundamentally changes. In the past, the best price/performance ratio was usually 1 or 2 rack unit servers with 2 processors. Higher density architectures, such as blade centers and larger symmetric multiprocessor systems usually carried a significant price premium that precluded their use for commodity clusters. The recent availability of 1U and 2U servers that contain 2 to 4 processors with 12 cores each are fundamentally changing the analysis. At our institution, the most recent cluster purchased contained 24 processor cores and 48 GB of memory for each 1U server. Although the price per system increased, surprisingly the price per CPU core has remained constant over the past few acquisition cycles. Moreover, since a single power supply per system now serves more cores, the power consumption factor and its

relationship to cluster design is also fundamentally changing. This trend is making the use of virtualization and virtual clusters much more viable and attractive, and the savings from reduced physical space use and power consumption has the potential to more than offset the staff and license costs needed to deploy virtual clusters. In terms of space, at our institution, the clusters purchased during the past three acquisition cycles have each taken less floor space due to the increased core count per system. Thus, although we are purchasing roughly the same number of cores at roughly the same cost, the physical space occupied by each successive cluster is half of the space of the previous cluster, which has effectively eliminated the need to construct a new data center. The final technology factor is the emergence of graphics processor unit (GPU) technology, which is now used on the Tianhe-1A, the fastest computer in the world [19]. It is not clear how virtualization systems will integrate GPUs as a co-processor into a virtual machine architecture.

The use of virtualization to create virtual clusters now makes it possible to support the use of customized operating system images that are tuned for a specific application. Without the use of virtualization, the cluster must be loaded with a uniform operating system image across the entire cluster, which is a "one size fits all" approach that is a trade-off between optimal application performance and ease of systems administration. The capability to support customized VM images opens a new world of possibilities for the use of light-weight kernels, low noise operating systems, and ultimately a highly tuned operating system that provides maximum application performance for a specific parallel application.

The systems administration effort needed to manage a large virtualized cluster is fundamentally different than the effort needed for a traditional cluster. Within the VM image, the knowledge and skills needed remain the same. At the underlying virtualization level, however, a significant amount of new knowledge and skills are required. Systems administrators need to learn to manage commercial (e.g. VMware) as well as open source (e.g. Xen, KVM, OpenVZ) virtualization systems for the physical nodes. In addition to this new knowledge, they will also need to learn to manage the IaaS, PaaS, and SaaS cloud computing layers needed to create and manage the virtual images deployed to the virtual cluster system. The cloud computing systems they would need to learn at a minimum include one of the popular cloud computing systems, such as Eucalyptus or OpenNebula. An additional layer of management that systems administrators would need to learn is how to manage the virtual networking layer and the VPNs and VLANs needed to provide private networking space for partitions of a virtual cluster, as well as the security infrastructure needed to keep the systems stable and secure.

Designing and operating the high performance network fabric needed to operate a high performance computing system is a critical element of operating a successful virtual cluster. If the network latency and bandwidth are significantly reduced, the collection of VMs will not adequately perform as a commodity cluster, which would invalidate the virtual cluster approach. On a physical system, high performance network interface cards (NIC) traditionally used for clusters (such as Myrinet and InfiniBand) are designed to work with custom device drivers that

reduce latency in the operating system by exploiting direct access to the system hardware and operating system kernel. On a virtual system, this is much more complicated. Hypervisor based systems as such Xen and VMware generally don't support specialized high performance network interface cards unless the NIC manufacture develops a device driver specifically for the hypervisor. Without such a driver, many hypervisor systems provide a "direct pass through" mode between a physical device and a VM to allow a device driver in a VM to directly control a device. The limitation of this approach is that only one VM can control and use the device, which eliminates the ability of the hypervisor to share the device among VMs. On the positive side, virtualization systems such as VMware and Xen use virtual network switches in the hypervisor to create a internal network for VMs running on the physical system. By using the virtual switch, it is possible to create a very high speed and low latency network among VMs on the same physical system. This feature of hypervisors will become more useful as core counts on multicore systems increase over time.

In the area of disk storage, the sharing of a limited set of physical disks by VMs is a significant bottleneck for virtual machines. Although virtualization technology can take advantage of multiprocessor/multicore systems by subdividing the processors and cores into partitions and efficient use large memory systems by partitioning memory, it is much more difficult to partition a limited number of local disk devices. The I/O bottleneck created by many VMs going through the same physical I/O system can severely hamper file system and virtual disk performance. The approach commonly used to overcome this problem is to use a SAN or other high speed storage network to create a shared storage environment, which can become expensive as the system scales up. Although it is possible to spread out VM I/O among several physical disks, the typical server hardware used for VMs have a limited number (4 to 6) of physical disks, which ultimately puts an upper bound on the number of I/O intensive VMs that can be effectively used on the physical hardware. This is a difficult issue for courses that require benchmarking and measurement of local file system of disk performance.

3.2 Pedagogical limitations of virtual and cloud computing for HPC

The fundamental differences in physical and virtual computing environments lead to a number of issues in determining the most relevant material to teach to students and how to teach it. Students greatly benefit from the tactile experience of configuring and changing the hardware in a physical system and wiring the network for their systems. They learn how the CPUs, memory, disks, and adapters are configured and placed in the system, as well as the system design choices and the effects of these choices on space used by the system and the cooling of the system.

The downside of virtual machines is that they completely lack this physical interaction between the student and the computer and networking hardware. Within virtualization systems, components, such as CPUs, memory, and disk become abstractions without a clear physical context and definition. As mentioned earlier, the observable performance of the virtual components will be affected by the behaviors of other virtual machines running on the physical

system. This situation doesn't help clarify student understanding of the performance effects of components and upgrades. There are also some positives to the use of virtual components. Students can easily add devices, e.g. many network interfaces and processor cores, as well as tune the mapping from virtual to physical on the hypervisor. These capabilities make it possible to explore scaling and tuning as well as investigation into novel cluster architectures.

Another major problem with virtualization is that is difficult to conduct repeatable performance measurements of parallel applications running on a shared physical infrastructure in which other VMs are busy.

A final problem is to develop an effective approach to manage virtual machine images for students using VMs for assignments and research, and how to manage the assignment of physical resources to students.

There are advantages to the use of virtualization technology for high performance computing and cloud computing. First, students can easily observe and measure the effects of changing the amount of memory and CPU cores for parallel applications and systems. Second, students could submit a virtual machine image as their submission for a project or an assignment. This makes it possible to archive VMs over time and to compare VM submissions to ensure the originality of students' work. Finally, the use of virtualization technology enables a greatly increased efficiency in the use limited laboratory hardware and networking resources.

3.3 Developing a hybrid laboratory infrastructure

An approach to address the shortcomings of virtualization technology while maintaining the advantages is to develop a hybrid laboratory infrastructure that includes a collection of physical desktop and server computing systems in conjunction with a set of virtualized systems. Designing systems to link the physical and virtual into a collective computing system will allow students to learn both physical and virtualization technologies, and to learn how to effectively integrate these two types of systems. The integration of physical and virtual helps students understand the differences and similarities between the two. Moreover, the ability to link physical with virtual network adapters and switches provide an excellent test bed for experimenting with networking architectures and understanding the reliability and performance effects of virtualization.

4. Experiences deploying and using cloud computing technologies for high performance computing

To develop a hands-on understanding of cloud computing technologies and their suitability for high performance computing, we installed, configured, and tested several cloud computing packages: Eucalyptus [13], Nimbus [15], and OpenNebula [18]. We investigated the use of these systems with Xen and VMware virtualization systems, as well as the ability of these systems to support the use of an additional paravirtualization layer using OpenVZ [9].

The Eucalyptus system is available as a set of yum and tar format downloadable packages, and is also available as an installation option for Ubuntu. To provide an illustrative example, the installation of Eucalyptus for CentOS first requires the installation of Xen and the Eucalyptus packages using the *yum* package utility. Once installation and configuration and complete, the Eucalyptus system is brought up by starting Eucalyptus service daemons, and then registering the head node and other physical nodes. Administration and configuration of Eucalyptus is performed through a web based administration interface. There are four networking options available in Eucalyptus that use dynamic IP addresses and VLANs to create a secure networking environment, which can make setting up networking in Eucalyptus complex and difficult to troubleshoot. The default VM images provided with Eucalyptus are easy to use, but the process involved in creating and then adding a custom VM image to Eucalyptus is complex. The problem is that the internal configuration of the VM image must match Eucalyptus' expectations, and must also first correctly function on the underlying virtualization system Eucalyptus is managing. Once the VM images are loaded and working, the web based management system works well. However, the CLI based management interface is cumbersome. Debugging and troubleshooting Eucalyptus is difficult, due to its heavy reliance on web services.

There are several advantages and disadvantages to Eucalyptus. The advantages are that Eucalyptus: can work well in a distributed internet environment, due to its heavy reliance on web services; is mature, since it is one of the first cloud computing systems; has a commercial version that is supported; has an excellent web based management interface. The disadvantages to Eucalyptus are: a heavy reliance on java, customized tools, and web services that makes it difficult to debug and trouble shoot problems with the system; uses web services, which does not scale well with the magnitude and size of the multi-gigabyte virtual disk images that it must move between physical systems; and that it has a complex and non-intuitive administration command line interface, which makes it difficult to write scripts and other commands that use Eucalyptus.

We also assessed the Nimbus cloud computing system by using an instance of Nimbus installed at Purdue called Wispy. The Wispy group provided a virtual box VM image for use on a local desktop computing system that contained all of the software and authentication keys needed to submit a VM to the Nimbus system. As of Summer 2010, only Xen images were supported in Nimbus, and the only VMs supported were preconfigured images. The remote nature of the system and reliance on X.509 certificates greatly complicated the debugging and troubleshooting process, and lack of access to debugging logs on the central cloud computing system made it very difficult to use the system.

The advantages to Nimbus are that it is supported by Argonne National Laboratory, and that it has some limited use today for specific applications within the scientific community in a trial mode. The disadvantages are that the strong legacy connection with Globus and X.509 based authentication makes installation and configuration complex; the undocumented requirements for the internal structure of the virtual machine images makes the system brittle; and finally that the

focus of Nimbus is on the use of clouds for scientific computing rather than as the basis for a general computing utility.

The third cloud computing system we investigated was OpenNebula, which was developed at the the Universidad Complutense de Madrid. OpenNebula is based on standard Linux technologies, such as NSF, ssh, and rsync. To manage the virtualization layer, OpenNebula uses the libvirt system, which provides a uniform management veneer layer over VMware, Xen, and KVM. This combination of standard Linux technologies with the uniform libvirt management interface greatly simplifies troubleshooting and debugging. The management of nodes, virtual machines, and networks is performed using an OpenNebula management command line interface, which is very logical and simple to use.

The advantages of OpenNebula are: it is based on standard Linux system utilities and services; is easy to troubleshoot and debug compared with Eucalyptus and Nimbus; and it provides a clear and logical management CLI. The disadvantage of OpenNebula is that the development is managed by a small group in Europe, which may make it difficult to get timely answers to any questions or problems.

One cloud computing system that we did not assess was Microsoft Hyper-V and Azure. These systems are not broadly used today for High Performance Computing, and consequently are not included in our HPC courses today.

4.1 Practical considerations in deploying cloud technologies for high performance computing education.

Our department provides courses in designing, building, and developing a high performance computing system cluster and parallel file systems. There are several practical considerations to be considered in using cloud computing technology for these courses. First, it is difficult to provision sufficient physical machines for scaling tests and performance comparisons among several parallel file systems simultaneously. Second, the amount of disk space needed for potentially hundreds of virtual machine disk images would require a tremendous amount of disk space as well as networking infrastructure to quickly transfer images among physical servers. Moreover, using older computer systems can be problematic, since the failure of a single physical server can render several virtual machines inoperable.

Based on the assessment of VMware, Eucalyptus, NIMBUS, we found that a pure hypervisor based virtualization approach is not scalable for several reasons. First, a large disk image is required for each virtual machine. There is not opportunistic sharing of *any* content among virtual machine images that could significantly reduce the amount of storage needed for virtual disks. Moreover, transferring and cloning large virtual disk images is difficult and time consuming. Second, the current state and stability of the management tools for Nimbus and Eucalyptus makes it difficult to easily and reliably manage VM images. VMware is the notable exception in this regard – the management GUI provided by VMware is exceptionally easy to

use and is reliable. Third, we found that Nimbus and Eucalyptus (the open source version) was unstable. Additionally, the community is migrating from Xen to KVM, so there is limited pedagogical value learn Xen instead of KVM.

Overall, we found that the combination of OpenNebula, KVM, and OpenVZ is the most promising practical approach for provisioning cloud computing systems. OpenNebula provides a straightforward and simple approach for deployment and managing Xen and KVM images that uses a suite of standard Linux technologies. On top of KVM virtual machine images, OpenVZ can deploy several virtual machine containers within each VM images that shares a kernel with the KVM VM image. For the parallel data systems class in which students design, build, and test a parallel file system, we are planning to have students build parallel file systems within OpenVZ containers. The major constraint in this approach is the I/O performance bottleneck for the I/O server components of parallel file systems running in virtual machines that must share physical disks on the services on which they operate. The approach we are considering to overcome this constraint is to schedule final performance runs for students running on physical servers. Based on our assessment of the cloud computing and virtualization technologies, we believe that we can scale up to at least 40 virtual machine containers using a combination of Open Nebula, KVM, and OpenVZ on five physical servers.

5. Conclusions

Based on our experiences with these systems, as well as VMware, we found that VMware is the most effective commercial system. For developing virtual clusters, we found that OpenNebula is the most effective open source cloud computing system. In related work, Sempolinski assessed Eucalyptus, OpenNebula, and Nimbus [17], and found that Eucalyptus is geared toward private companies, Nimbus is targeted for specific scientific communities with broad customization requirements, and OpenNebula is geared toward groups interested in cloud and VM technology. Our experiences are similar to Sempolinski's observations.

In terms of design, we concluded that although the use of virtualization masks the physical nature of the hardware, virtualization and cloud computing technologies has the potential to open new forms of exploration and teaching in high performance computing. This new capability provides the ability to explore larger scale clusters than would be possible with physical hardware alone. Also, it will allow users and students to better understand the effects of memory, CPU, and disk on cluster design and the performance of parallel applications. Moreover, it will greatly ease the process of understanding how networking configuration and architecture will affect performance and design of clusters and applications.

There are no functional open source turn-key virtual cluster systems available today. OSCAR-V [20] attempted to create a version of the OSCAR toolkit for clusters, but the software doesn't currently work and there are no plans by the developers to fix the software in the near future. Our assessment of the cloud computing systems described in this paper leads us to conclude that OpenNebula is the best potential system for developing virtual clusters for high performance

computing and cloud computing education. On the hypervisor level, our observation is that due to increasing support for KVM, KVM is a first choice for virtualization, followed closely by VMware and Xen.

REFERNECES

1. Feitelson, D.G., *The supercomputer industry in light of the Top500 data.* Computing in Science & Engineering [see also IEEE Computational Science and Engineering], 2005. **7**(1): p. 42-47.
2. Goldberg, R., *Architectural Principles for Virtual Computer Systems*. 1973, Storming Media.
3. Vmware, I., *VMware.* Inc., VMware products, VMware, Inc., Palo Alto, CA, USA (2008)< http://www. vmware. com/products/>[accessed 01.03. 08], 2008.
4. Barham, P., et al., *Xen and the art of virtualization*, in *Proceedings of the nineteenth ACM symposium on Operating systems principles*. 2003, ACM Press: Bolton Landing, NY, USA.
5. Kivity, A., et al. *kvm: the Linux virtual machine monitor*. 2007.
6. Bellard, F. *QEMU, a fast and portable dynamic translator*. 2005: USENIX.
7. Victoria, B., *Creating and Controlling KVM Guests using libvirt*. 2009, University of Victoria.
8. Yu, J. *Performance Evaluation on Linux Bridge*. 2004.
9. SWSoft. *OpenVZ User's Guide*. 2005 July 1, 2009; Available from: http://download.openvz.org/doc/OpenVZ-Users-Guide.pdf.
10. McLennan, M. and R. Kennell, *HUBzero: A Platform for Dissemination and Collaboration in Computational Science and Engineering.* Computing in Science & Engineering, 2010. **12**(2): p. 48-53.
11. Watson, J., *Virtualbox: bits and bytes masquerading as machines.* Linux Journal, 2008. **2008**(166): p. 1.
12. Corbato, F. and V. Vyssotsky. *Introduction and overview of the Multics system*. 1965: ACM.
13. Nurmi, D., et al. *The eucalyptus open-source cloud-computing system*. 2009: IEEE Computer Society.
14. Amazon, E., *Amazon elastic compute cloud.* Retrieved Feb, 2009. **10**.
15. Keahey, K., et al., *Science clouds: Early experiences in cloud computing for scientific applications.* Cloud Computing and Applications, 2008. **2008**.
16. Foster, I., *Globus Toolkit Version 4: Software for Service-Oriented Systems.*, in *IFIP International Conference on Network and Parallel Computing*, Springer-Verlag. p. 2-13.
17. Sempolinski, P. and D. Thain, *A Comparison and Critique of Eucalyptus, OpenNebula and Nimbus.*
18. Sotomayor, B., et al., *Capacity leasing in cloud systems using the opennebula engine.* Cloud Computing and Applications, 2008. **2008**.
19. Chen, G., *Petaflop supercomputers of China*. Frontiers of Computer Science in China, 2010. **4**(4): p. 427-427.
20. Vallée, G., T. Naughton, and S. Scott. *System management software for virtual environments*. 2007: ACM.