Configurable Virtualized System Environments for High Performance Computing*

Christian Engelmann[1,2], Stephen L. Scott[1], Hong Ong[1],
Geoffroy Vallée[1], and Thomas Naughton[1,2]
[1]Computer Science and Mathematics Division,
Oak Ridge National Laboratory, Oak Ridge, TN 37831, USA
[2]Department of Computer Science,
The University of Reading, Reading, RG6 6AH, UK
[1]{engelmannc,scottsl,hongong,valleegr,naughtont}@ornl.gov
[2]{c.engelmann,t.naughton}@reading.ac.uk

ABSTRACT
Existing challenges for current terascale high performance computing (HPC) systems are increasingly hampering the development and deployment efforts of system software and scientific applications for next-generation petascale systems. The expected rapid system upgrade interval toward petascale scientific computing demands an incremental strategy for the development and deployment of legacy and new large-scale scientific applications that avoids excessive porting. Furthermore, system software developers as well as scientific application developers require access to large-scale testbed environments in order to test individual solutions at scale. This paper proposes to address these issues at the system software level through the development of a virtualized system environment (VSE) for scientific computing. The proposed VSE approach enables "plug-and-play" supercomputing through desktop-to-cluster-to-petaflop computer system-level virtualization based on recent advances in hypervisor virtualization technologies. This paper describes the VSE system architecture in detail, discusses needed tools for VSE system management and configuration, and presents respective VSE use case scenarios.

Keywords
Virtualization, Hypervisor, High Performance Computing, On-demand Computing, Virtualized System Environment

1. INTRODUCTION
The U.S. Department of Energy (DOE) plans to deploy a 1 petaflop (quadrillion of calculations per second, or Pflop/s) scientific high performance computing (HPC) system by the 2008/9 time frame. A similar system deployment effort is currently being undertaken by the U.S. National Science Foundation (NSF). In order for these systems to run "out-of-the-box", several challenges in petascale system software and application runtime environments have to be addressed to assure day-one operation capability. Efficiently exploiting tens-to-hundreds of thousands of processor cores using tens-to-hundreds of thousands of interdependent computational tasks requires appropriate scalability, manageability, and ease-of-use at the system software and application runtime environment (RTE) level. Furthermore, the expected rapid system upgrade interval demands an incremental strategy for scientific application development and deployment that avoids excessive porting.

Current efforts in operating system (OS) research and development at vendors, such as Cray and IBM, and within DOE's Forum to Address Scalable Technology for Runtime and Operating Systems (FAST-OS) [7] concentrate on numerous varying approaches ranging from custom lightweight solutions, such as Catamount on the Cray XT3/4 [3] and the Compute Node Kernel (CNK) on the IBM Blue Gene/L system [18], to scalable Linux variants, like ZeptoOS [1]. From recent HPC system deployment experience, it has become clear that there is no one-size-fits-all OS solution. Each OS has its own design and performance advantages, supported platforms, and targeted scientific applications. Furthermore, it is not clear at this point which OS will be better suited or even available for each of the planned petascale class systems and for the targeted set of scientific applications.

A petaflop class computer with tens-to-hundreds of thousands of processors raises many challenges for both, its users and system administrators. A management approach that is consistent with small- to large-scale systems is crucial in order to easily port and run the next-generation large-scale scientific applications developed today for the targeted platform. The potential problem lies in porting and successfully executing legacy and new large-scale scientific applications. It is exacerbated by the fact that many scientific applications are developed on desktop systems or small-scale clusters, as resources and system knowledge are gained, the codes are enhanced to exploit the targeted supercomputer environment. Consequently, scientific application deployment often takes significantly more time than anticipated.

In addition, access to large-scale testbed environments is needed for system software developers as well as for scientific application developers in order to test individual software solutions at scale. Both groups directly compete with each other for access to a limited set of testbed environ-

*Research sponsored by the Laboratory Directed Research and Development Program of Oak Ridge National Laboratory, managed by UT-Battelle, LLC for the U.S. Department of Energy under Contract No. DE-AC05-00OR22725.

ments. Furthermore, system software tests often require to modify or replace the currently installed system software suite, which often contradicts with production system use policies of HPC centers.

These challenges are by no means new, but their impact increases in magnitude as HPC systems dramatically scale up in processor count, while supporting only one specific system software suite. Furthermore, with today's emerging multi-core processor architectures and with next-generation heterogeneous accelerator-supported computing environments, even more development time will be incurred as scientific application as well as system software developers try to discover the methods of efficiently exploiting the power of these new technologies.

This paper proposes to address these issues at the system software level through the development of a virtualized system environment (VSE) for scientific computing. In addition to providing a scalable and reliable "sandbox" environment for scientific application development on desktops and clusters, the VSE will offer an identical production environment for scientific application deployment on existing terascale and future petascale HPC systems. The proposed VSE concept enables "plug-and-play" supercomputing through desktop-to-cluster-to-petaflop computer system-level virtualization based on recent advances in hypervisor virtualization technologies. The overall goal of the proposed effort is to advance the race for scientific discovery through computation by enabling day-one operation capability of newly installed systems and by improving productivity of system software and scientific application development and deployment.

In the following, we describe the VSE research background and approach in more detail. We discuss the proposed VSE system architecture and the needed tools for VSE system management and configuration. We continue with a presentation of use case scenarios for the VSE concept, and a review of related work in this area. We conclude with a short summary of the presented research and a brief outlook on future work.

2. VIRTUALIZED SYSTEM ENVIRONMENTS

The proposed VSE approach is derived from the virtual environment (VE) concept for scientific application development and deployment, which is presently being studied in the Harness Workbench project [23] at Oak Ridge National Laboratory, the University of Tennessee, and Emory University.

In a traditional HPC application development and deployment model, system administrators are solely responsible for system-wide installation of supporting software and scientific libraries. Scientific application developers write, compile, and run their codes utilizing these resources. While this model makes perfect sense for general-purpose software components, like optimized MPI [25, 24, 15] implementations or BLAS/LAPACK [20, 21] packages, it is less appropriate for libraries that are unconventional and more problems specific. In situations when only a few scientists in an organization need some (usually recent) package, it might be more reasonable to enable them to perform user-specific, local (home directory) installation, rather than placing that burden on site administrators. Even though such local, user-level installations may be technically possible (by careful setup of appropriate environment variables, and appropriate use of configuration options) the technicalities of the procedure and potential for conflicts render it fraught with pitfalls to scientific application developers in practice.

The Harness Workbench project addresses this problem by allowing the scientific application developer to deploy VEs for scientific application development and deployment. The XML-based VE configuration description contains the necessary modifications to be applied to the base system, which are needed by an application in order to compile, link, and run. The current Harness Workbench approach for VEs focuses on the `chroot` mechanism of a Unix-type OS in conjunction with linking/copying files and directories to/from the original base system root directory to the VE root directory, and/or mounting the original base system root directory underneath VE root directory using features of the UnionFS [27] stackable file system, such as copy-on-write.

The limitation to the `chroot` mechanism and its system security implications are a major concern for deployment on production-type HPC systems. In order to alleviate this issue and to further extend the VE concept, the VSE approach is based on system-level hypervisor virtualization technology that provides better VE isolation in form of a "sandbox" environment for scientific application development and deployment. Additionally, the VSE concept is not limited to the application space as system software development and deployment efforts also can profit from hypervisor virtualization technology. While the VE concept allows specifying the RTE requirements of a scientific application in form of a XML configuration file, the VSE approach is extending this idea to the entire software suite installed on a HPC system, including OS (kernel, libraries, and services), RTE(s) (libraries and services), and access policies for external resources, *e.g.*, for a parallel file system.

2.1 System Architecture

The proposed software infrastructure (Figure 1) utilizes system-level hypervisor virtualization technology in combination with configuration mechanisms for virtual machines in order to provide a powerful abstraction for portability, isolation, and customization of the entire software suite of a HPC system.

At its core, a virtual machine monitor (VMM), *i.e.*, hypervisor, offers a low-level abstraction layer to support a wide variety of operating systems running inside a virtual machine (VM). A low-level protection mechanism isolates VM instances and the host OS residing on the same processor from each other, which enables on-demand VM instantiation on development systems, such as desktops or HPC system service nodes, and permits oversubscription of processors on compute nodes in case of large-scale emulation on a smaller scale system.

Inside the VM resides the entire software suite running on a HPC system customized based on a VSE configuration description to the needs of the running job, which may be a

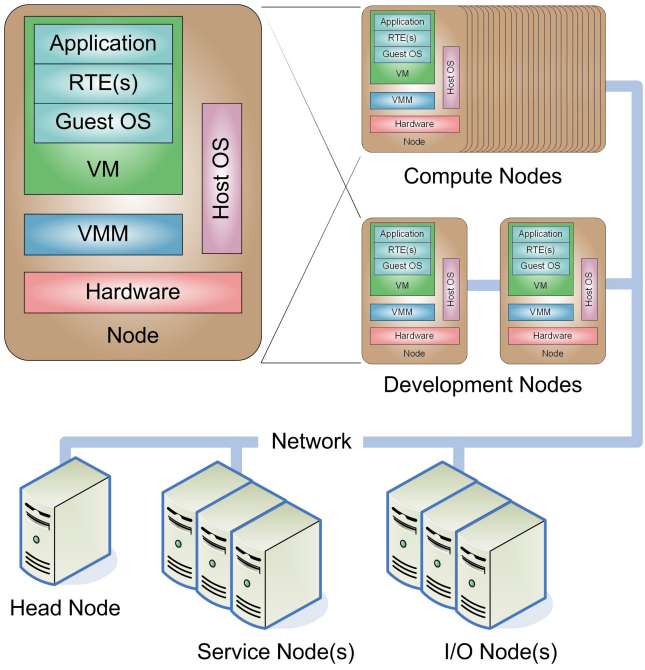

Figure 1: The virtual system environment software architecture for scientific high performance computing systems utilizes system-level virtualization on compute nodes and development environment service nodes to provide "sandbox" environments for system software and scientific application development and deployment.

production run or a test run of an application or of a system software component. The software suite running inside the VM may access external resources, such as a parallel file system. The VM is configured to access external resources based on the VSE configuration description, *e.g.*, for accessing a different root file system.

Due to the utilization of type-I system-level virtualization technology [8, 11] a host OS is required to perform certain basic functionality, such as providing VM management, *e.g.*, VM creation and destruction. The host OS is identical to the original OS on development systems, while on compute nodes it also may be a customized lightweight OS solution. In some cases, the host OS on compute nodes may be eliminated completely if its functionality can be offloaded to service nodes.

Current type-I virtualization solutions, like Xen [2, 32], use the host OS also for hardware drivers. With the development of OS-bypass technologies and hardware-level virtualization support in processors and devices, OS-level hardware drivers and user-level libraries will be able to access virtualized hardware from within a VM without requiring a host OS for driver support.

2.2 System Management

A VSE, like traditional systems, is composed of different elements: OS (kernel, drivers, and low-level system libraries), RTE(s) (libraries, compilers, and scientific runtimes), and user applications. The characteristics of a VSE do not differ much from non-virtualized system environments. Therefore, traditional tools can be reused for performing routine system management. However, they need to be adapted in order to make them aware of the virtualized nature of a VSE.

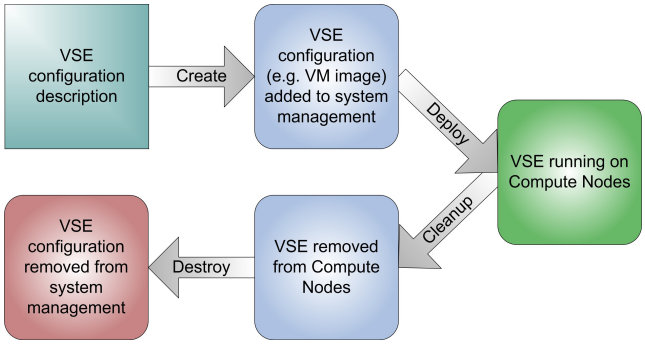

Figure 2: The life cycle of a virtual system environment encompasses the creation of a configuration from a configuration description, the deployment of the configuration on compute nodes and development environment service nodes as virtual machines, the cleanup of deployed virtual machines, and the destruction of the virtual system environment configuration.

VSE instances are administered using a set of system management tools and configuration files that create, deploy, cleanup, and destroy VMs belonging to a VSE instance in form of a customized parallel virtual system (Figure 2).

VSE configuration tools translate individual VSE configuration descriptions to respective system dependent VSE configurations, *e.g.*, VM images to be loaded on compute nodes for job runs or on service nodes for compilation runs. These VSE configuration tools create and destroy VSEs in the form of system dependent VSE configurations, which can be then used by system management tools for VSE deployment and cleanup.

In order to enable the VSE concept in HPC environments, existing system management tools, *e.g.*, HPC system resource managers, need to be enhanced to support scalable management of lightweight system-level hypervisor virtualization technology on compute nodes. Furthermore, respective OS deployment mechanisms for compute nodes, *e.g.*, HPC system installation suites, need to be improved to allow for scalable on-demand deployment of VMs on service and compute nodes.

In order to enable the VSE concept directly on the compute nodes and on single system environments, such as the desktop of an application software developer or the development environment server of a HPC system, existing system management tools for VM deployment and cleanup need to be adapted in order to interface with system resource management tools and software development environment solutions.

2.3 Configuration Management

VSEs are preferably configured offline, *i.e.*, VM configurations are modified without actually deploying VMs. However, special circumstances may exist in which a VM configuration needs to be modified while it is is running. In this case, the VM to be modified is deployed to the desktop of the application software developer or to the development environment server of the HPC system for modification. Configuring a VSE may also include to compile, link, and install an application or system software component into a VM configuration depending on the VSE deployment mechanism. This may require online VM configuration, *e.g.*, when using the system probing features of the GNU autotools [29].

A VSE configuration description specification is needed that allows defining the properties of a VSE, such as installed OS, RTE(s) and services, and access policies for external resources. Respective VSE configuration tools are needed as well with support for offline and online VM configuration.

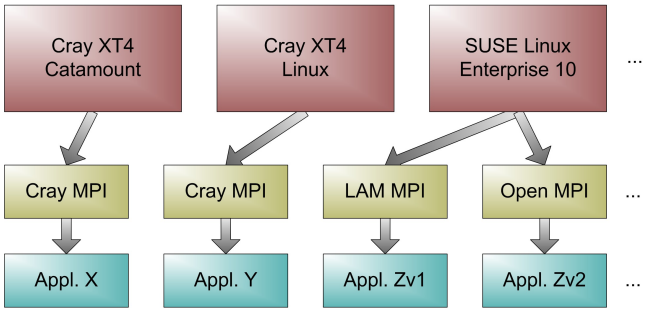

Figure 3: The hierarchical virtual system environment configuration description scheme enables users to override, remove, or add certain configuration options using vendor or system operator configuration descriptions as a base configuration description.

A hierarchical VSE configuration description scheme (Figure 3) enables users to override, remove, or add certain configuration options, such as libraries and services, using vendor or system operator configuration descriptions as a base configuration description. Certain configuration options may be reserved for system administrators to avoid unintentional or malicious configurations, and to maintain site-specific policies. For example, if an application is supposed to run in a specific Linux distribution, it derives its VSE configuration description from a VSE configuration description for this Linux distribution. A different base VSE configuration description for this Linux distribution may be supplied by system administrators of HPC centers to adapt to system-specific VSE configuration and deployment mechanisms, to enable system-specific hardware and software support, and to enforce site-specific policies.

VSE configuration tools translate individual VSE configuration descriptions to respective system dependent VSE configurations. Depending on the granularity of the VSE configuration description scheme, the following VSE configuration mechanisms may exist: copying files or directories into a VM configuration; copying, linking, and/or deleting files or directories within a VM configuration; compiling, linking, and installing software into/within a VM configuration; changing existing (configuration) file content within a VM configuration; installing/removing software packages into/from a VM configuration; and supplying boot options to the OS of the VM configuration. Existing tools for software and package management may be adapted to or reused for VSE configuration. The VE configuration and management tools currently being developed by the earlier mentioned Harness Workbench project may be reused for fine-grain VM configuration.

3. USE CASE SCENARIOS

The main motivation behind this research effort in VSEs is to improve the productivity of system software and scientific application development and deployment for current terascale and next-generation petascale scientific HPC systems. The proposed VSE concept enables "plug-and-play" supercomputing through desktop-to-cluster-to-petaflop computer system-level virtualization by enabling developers with a seamless development, test, and deployment process for system software and scientific applications across the various systems involved in these activities.

3.1 Application Development and Deployment

Scientific application developers typically implement early prototypes on their desktops using standard OSs and RTE(s), *e.g.*, Linux and Open-MPI [24]. Depending on the application and on the developers preference, different programming languages, *e.g.*, C, Fortran and/or Python, and application programming interfaces (APIs), *e.g.*, SysV sockets and/or POSIX file I/O, are utilized. Initial tests are performed on the developers desktop, and as the application matures, more extensive tests are performed on small-scale scientific computing systems, such as on Linux clusters.

Moving an application prototype from a desktop to a small-scale scientific computing system already implies a certain adaptation effort as OSs and RTE(s) may not be the same, some APIs may not be supported, and different compiler and linker flags are needed. In other words, the system environment changes, therefore the application needs to be adapted, *i.e.*, ported.

Once a certain maturity threshold is reached, the scientific application is deployed on large-scale production-type HPC systems, which typically involves more extensive adaptation efforts as these systems employ even different environments. At this initial deployment stage, applications are commonly ported to several HPC system platforms at the same time in order to fully utilize provided resource allocations. Furthermore, application porting is performed with each HPC system upgrade and with each newly deployed HPC system. This results in a continuous porting effort for scientific applications, which diverts resources away from developing new and improving legacy scientific applications.

The VSE concept changes the way scientific applications are developed and deployed (Figure 4) by adapting the system environment required by an application to the system the application is running on, instead of trying to adapt the application to each system environment. Due to the use of system-level virtualization technology, the adaptation of

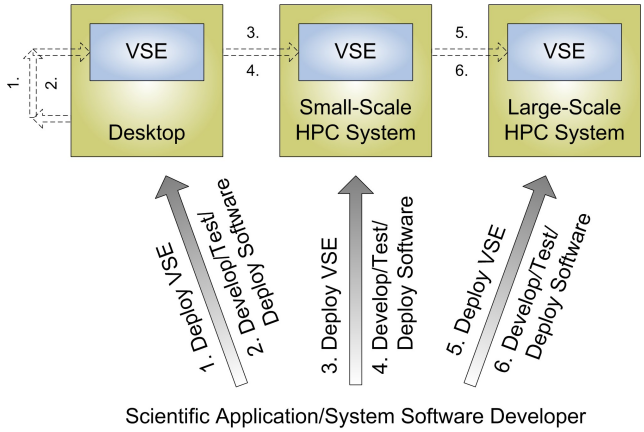

Figure 4: The virtual system environment use-case scenario for software development and deployment provides a seamless development, test, and deployment process for system software and scientific applications across the various involved systems.

system environments to systems is much simpler and transparent to the application.

Using the VSE approach, scientific application developers continue to implement early prototypes on their desktops. However, by deploying VSEs on these desktops, scientific application developers are able to develop in the production-type system environment utilizing the features of a particular HPC system. Using VSEs on application developer desktops is an added optional benefit of the VSE concept and by no means a mandatory requirement for the scientific application developer.

Once a certain maturity threshold is reached, instead of porting the scientific application to a small-scale scientific computing system, the developer describes the scientific application requirements in form of a VSE configuration description by deriving existing VSE configuration descriptions of HPC systems provided by system administrators. Although interaction with system administrators and scientific application support groups of HPC centers is needed, it is by far less than the amount of time spend for traditional scientific application porting efforts.

The initial VSE configuration description of a scientific application defines the best-fit requirements. Application deployment to small-scale scientific computing systems or to large-scale production-type HPC systems is seamlessly provided by deploying this best-fit VSE configuration. Often, the-best fit requirements of a scientific application depend on certain system parameters, such as scale or available memory. Individual application tuning may be supported by the VSE concept by using tuned VSE configuration descriptions. For example, if the application requires a Linux-like OS due to the use of SysV and POSIX features, VSE configuration description should be changed to use a lightweight Linux variant on supported large-scale production-type HPC systems.

The VSE deployment mechanisms for scientific application development and deployment rely on the system and configuration management tools described earlier. Additionally, automatic VSE deployment for scientific application development and testing may be supported by interfacing with integrated development environments (IDEs), such as Eclipse [5].

3.2 System Software Development and Deployment

System software development and deployment efforts often require to modify or replace a currently installed system software suite for testing purposes. This procedure often violates use policies of large-scale production-type HPC systems or even of entire HPC centers. As a result, system software is mostly developed on desktops and small-scale scientific computing systems. Adaptation of a developed system software suite for testing and deployment purposes to different HPC system architectures is a necessity due to different hardware and software environments. Large-scale development and testing of system software is typically performed in conjunction with its deployment, *i.e.*, when a scientific HPC system is newly installed or upgraded.

The VSE concept changes the way system software is developed and deployed by providing *virtual testbeds* on desktops, small-scale scientific computing systems, and large-scale HPC systems. Instead of developing and testing system software on the bare hardware by replacing the original system software, system-level virtualization technology allows to encapsulate system software to be tested in "sandbox" testbed environments in the form of VMs on top of a low-level hardware abstraction and isolation provided by VMMs. In case of a serious failure of the tested system software, the VMM protects the host OS and other VMs from unintended corruption. Current type-I virtualization solutions, like Xen, are already being used by system software developers outside the HPC community. The VSE approach extends this idea to parallel and distributed systems for scientific computing.

In contrast to using system-level virtualization technology for developing system software for desktops and servers, the parallel and distributed nature of scientific computing systems requires the deployment of VMs on multiple compute nodes at various scale in form of a parallel virtual testbed system. Large-scale HPC system emulation may be achieved by oversubscribing compute nodes, *i.e.*, by deploying more than one VM per compute node.

Since the system software test run becomes an actual application test run to be performed on a desktop, small-scale scientific computing system, or large-scale HPC system, the VSE configuration description of a system software defines the system software, RTE, and test application requirements.

The VSE deployment mechanisms for system software development and deployment are similar to those of scientific applications (Figure 4). They rely on the same system and configuration management tools described earlier. Automated testing of system software components or entire system software suites may be supported by interfacing

with IDEs, and/or, due to the rather collaborative nature of system software development, with management tools for source code repositories.

4. RELATED WORK

Related research and development efforts focus on system-level virtualization technologies, virtual machine configuration mechanisms, and virtual system management tools. The VSE approach presented in this paper relies on solutions in each of these areas. However, system-level virtualization technologies typically focus on the desktop and server market, while virtual machine configuration mechanisms and virtual system management tools have also been recently developed for HPC systems.

4.1 System-level Virtualization

The recently increased world-wide interest by researchers, developers, and enterprise businesses in system-level virtualization was sparked a few years ago with the development of lightweight hypervisor technology using type-I virtualization. Over the last years, this technology matured up to the point that processor manufacturers, like Intel and AMD, very recently incorporated hardware virtualization support into their products. Soon, device manufacturers, like network card vendors, will follow and virtualization will be supported by most computer hardware components.

4.1.0.1 Xen

Xen [2, 32] is an open source VMM for IA-32, x86-64, IA-64, and PowerPC architectures. Its type-I system-level virtualization allows one to run several VMs (guest OSs) in a unprivileged domain (DomU) on top of the VMM on the same computer hardware at the same time using a host OS running in a privileged domain (Dom0) for VM management and hardware drivers. Several modified OSs, such as FreeBSD, Linux, NetBSD, and Plan 9, may be employed as guest systems using paravirtualization, *i.e.*, by modifying the guest OS for adaptation to the VMM interface. Using hardware support for virtualization in processors, such as Intel VT and AMD-V, the most recent release of Xen is able to run unmodified guest OSs inside VMs.

Xen originated as a research project at the University of Cambridge, led by Ian Pratt, senior lecturer at Cambridge and founder of XenSource, Inc. This company now supports the development of the open source project and also sells enterprise versions of the software. The first public release of Xen was made available in 2003.

In the context of this paper, Xen provides a perfect starting point as a system-level virtualization solution for VSEs in HPC environments. However, the Xen development effort is targeted toward the desktop and server market, where use case, such as single VM vs. multiple VMs per processor, and system characteristics, like processor scale, are much different from scientific HPC.

4.1.0.2 VMware

VMware Inc. [31] offers a wide range of system-level virtualization solutions, including the free VMware player and VMware Server (formerly VMware GSX Server). While the mentioned free products and the non-free VMware Workstation employ type-II virtualization, *i.e.*, VMs are actual processes inside the host OS, VMware Inc. also provides a non-free type-I system-level virtualization solution, VMware ESX Server, based on hypervisor technology. The company further distributes an infrastructure solution (VMware Infrastructure) and various data-center products for deployment of system-level virtualization in enterprise businesses, such as for server consolidation.

Similar to Xen, VMware products also target the desktop and server market. However, VMware is not open source and type-I products are non-free. This makes it rather hard to use VMware in an HPC environment. It is our intend to use the free VMware products only for proof-of-concept prototypes, if at all.

4.1.0.3 L4

L4 is a microkernel originally designed and implemented by Jochen Liedtke [17]. The L4 microkernel has complete control over all hardware access and is the only component in the system running in privileged mode. It offers minimal abstractions and mechanisms to support isolation and communication for all OS services. In order to support virtualization of OS instances, L4 provides abstractions for timers and interrupts as well as virtual memory management, and encapsulates these as IPC messages. L4Linux [10] is a port of the Linux 2.6 kernel to run on top of the L4 API. Hardware accesses in LinuxL4 are replaced by IPC calls to the underlying L4 microkernel.

While the proposed VSE approach does not focus on using a microkernel as a virtualization layer, there are certain similarities between microkernels, such as L4, and hypervisors, like Xen. As hypervisors evolve in the next few years, researchers may profit from existing microkernel technology.

Moreover, microkernel technology plays an important role in custom lightweight OS solutions for HPC, such as Catamount on the Cray XT3/4 [3] and the Compute Node Kernel (CNK) on the IBM Blue Gene/L system [18]. Using these existing lightweight OS solutions as a host OS requires integrating a microkernel with hypervisor technology.

4.2 VM Configuration and Virtual System Management

Appropriate mechanisms for VM configuration are a necessity for efficiently using virtualization technology. Preconfigured system images, *e.g.*, boot disk partition image files, are typically used to avoid the system software and application installation process for every instantiation of a VM. Additionally, virtual system management tools are a necessity as well in order to create, destroy, migrate, suspend, resume, and load balance VMs.

4.2.0.4 OSCAR-V

The Open Source Cluster Application Resources (OSCAR) toolkit is used to build and maintain HPC clusters [19]. The toolkit has been recently extended to support system-level virtualization technology, *e.g.*, Xen and QEMU. This virtualization-enhanced version, OSCAR-V [28], includes additional tools to create and manage VMs atop a standard

OSCAR cluster. The OSCAR-V solution combines existing OSCAR facilities with a new VM Management (V2M) tool that provides a high-level abstraction for the interaction with underlying VM implementations.

The OSCAR-V enhancements were developed recently by our team at Oak Ridge National Laboratory to help exploring virtualization technology in HPC environments, and as a first step toward a VSE configuration and system management suite.

4.2.0.5 Virtual Workspaces

The Virtual Workspaces project [12, 30] is an effort to exploit virtualization technology for the Grid [13]. The goal is to capture the requirements for an execution environment in the Grid in form of a virtual workspace definition, and then use automated tools to find, configure, and provide an environment best matching those requirements. Virtualization technology, such as Xen, is being used in conjunction with the Globus Toolkit for dynamic provisioning of customized and controllable remote execution environments.

While the Virtual Workspaces approach seems to be similar to the VSE concept proposed in this paper, the major difference between both research efforts is the resource oriented aspect of the Virtual Workspaces approach and its focus on computing environments with distributed resources. The VSE concept focuses on tightly coupled large-scale production-type HPC systems, where highly scalable, lightweight solutions are needed and performance is the ruling metric. The Virtual Workspaces project focuses on more loosely coupled small-scale distributed cooperative computing environments, where interoperability is needed most.

4.2.0.6 Virtuoso

Virtuoso [22] is a resource management system build on top of VMware for managing VMs in a distributed Grid computing environment. The Virtuoso solution performs admission control of VMs, and provides the ability for the system to adapt when the user cannot state his resource requirements. It also offers the ability to support a mode of operation in which VMs and other processes compete for resources.

The Virtuoso project further explores network virtualization issues, such as a layer 2 virtual network system (VNET) [16, 26] and the virtual traffic and topology inference framework (VTTIF) [9]. VNET creates virtual network overlays of VMs residing on distributed hosts. It provides a layer-2 Ethernet that connects remote VMs to a local physical LAN. On the other hand, the VTTIF observes every packet sent by a VM and infers from this traffic a global communication topology and traffic load matrix among a collection of VMs.

Similar to the Virtual Workspaces project, the Virtuoso project targets distributed Grid computing environments. However, its research in VM management and in network virtualization may be reused for VSEs in HPC environments.

4.2.0.7 Cluster On-Demand

In addition to the mentioned projects, there have been recent research and development efforts in using virtualization technology for dynamic virtual clusters or cluster on-demand (COD) provision [14, 4, 6]. These approaches target capacity HPC as well as distributed Grid computing environments for dynamic (on-demand) allocation of resources in form of encapsulated virtual cluster computing environments. Research in this area focuses on matching resources with COD provision requirements, deploying virtual clusters on resources, and isolating virtual cluster instances from each other and from the host system(s).

The VM configuration and system management tools developed by these projects will highly influence the development of appropriate tools for the VSE approach.

5. CONCLUSIONS

We have presented the virtualized system environment (VSE) concept for scientific high performance computing (HPC). The proposed approach utilizes system-level virtualization for improving the development and deployment processes of system software and applications in scientific HPC environments. Based on recent advances in hypervisor virtualization technologies, the solution presented in this paper enables "plug-and-play" supercomputing through desktop-to-cluster-to-petaflop computer system-level virtualization. It enables software developers with a seamless development, test, and deployment process across the various systems involved in these activities.

The overall goal of the proposed effort is to advance the race for scientific discovery through computation by enabling day-one operation capability of newly installed systems and by improving productivity of system software and scientific application development and deployment. To this end, the VSE approach provides an identical, scalable, and reliable "sandbox" environment for software development and deployment on desktops, small-scale scientific computing systems, and large-scale production-type HPC systems.

In this paper, we discussed the VSE research background and approach, the proposed VSE system architecture, the needed tools for VSE system management and configuration, and the use case scenarios for the VSE concept.

For the implementation of the proposed approach, several software research and development efforts need to be undertaken. First, a hierarchical VSE configuration description scheme needs to be devised and appropriate tools are required to allow for VSE configuration. Second, existing system-level virtualization technologies and virtual system management tools need to be adapted to allow for VSE deployment. Third, existing system-level virtualization technologies have to be modified or new solutions must be developed to provide scalable hypervisors for HPC with low-to-zero performance impact.

While the implementation of the VSE concept seems to be straightforward, many open research questions remain, such as the deployment of scalable HPC hypervisors. Future work of our research team will focus on these open research questions, while implementing the proposed VSE approach in collaboration with university and industry partners.

6. REFERENCES

[1] Argonne National Laboratory, Argonne, IL, USA. ZeptoOS: The Small Linux for Big Computers.

Available at http://www-unix.mcs.anl.gov/zeptoos, 2007.

[2] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield. Xen and the art of virtualization. In *Proceedings of the* 19th *ACM Symposium on Operating Systems Principles (SOSP) 2003*, pages 164–177, Bolton Landing, NY, USA, 2003.

[3] R. Brightwell, S. M. Kelly, and J. P. VanDyke. Catamount software architecture with dual core extensions. In *Proceedings of* 48th *Cray User Group (CUG) Conference 2006*, Lugano, Ticino, Switzerland, May 8-11, 2006.

[4] J. S. Chase, D. E. Irwin, L. E. Grit, J. D. Moore, and S. E. Sprenkle. Dynamic virtual clusters in a Grid site manager. In *Proceedings of the* 12th *IEEE International Symposium on High Performance Distributed Computing (HPDC) 2003*, page 90, Seattle, WA, USA, June 24-27, 2003.

[5] Eclipse Integrated Development Environment, 2007. Available at http://www.eclipse.org.

[6] W. Emeneker, D. Jackson, J. Butikofer, and D. Stanzione. Dynamic virtual clustering with Xen and Moab. In *Lecture Notes in Computer Science: International Symposium on Parallel and Distributed Processing and Application (ISPA) Workshops 2006*, pages 440–451, Sorrento, Italy, Dec. 4-7, 2006.

[7] Forum to Address Scalable Technology for Runtime and Operating Systems (FAST-OS). Available at http://www.fastos.org, 2007.

[8] R. P. Goldberg. Architecture of virtual machines. In *Proceedings of Workshop on Virtual Computer Systems*, pages 74–112, Cambridge, MA, USA, 1973.

[9] A. Gupta and P. A. Dinda. Inferring the topology and traffic load of parallel programs running in a virtual machine environment. In *Proceedings of* 10th *Workshop on Job Scheduling Strategies for Parallel Processing (JSPPS) 2004*, pages 125–143, New York, NY, USA, June 13, 2004.

[10] H. Hartig, M. Hohmuth, J. Liedtke, and S. Schonberg. The performance of μ-kernel based systems. In *Proceedings of* 16th *Symposium on Operating System Principles (SOSP) 1997*, pages 66–77, Saint Malo, France, Oct. 5-8, 1997.

[11] IBM Corporation. IBM Systems Virtualization Version 2 Release 1 (2005). Available at http://publib.boulder.ibm.com/infocenter/eserver/v1r2/topic/eicay/eicay.pdf, 2007.

[12] K. Keahey, I. Foster, T. Freeman, and X. Zhang. Virtual workspaces: Achieving quality of service and quality of life on the Grid. *Scientific Programming*, 13(4):265–276, 2005.

[13] C. Kesselman and I. Foster. *The Grid: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann Publishers, San Francisco, CA, USA, 1998.

[14] N. Kiyanclar, G. A. Koenig, and W. Yurcik. Maestro-VC: A paravirtualized execution environment for secure on-demand cluster computing. In *Proceedings of* 6th *IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGrid) (CCGRID) 2006*, page 28, Singapore, May 16-19, 2006.

[15] LAM-MPI Project at Indiana University, Bloomington, IN, USA, 2007. Available at http://www.lam-mpi.org.

[16] J. Lange, A. Sundararaj, and P. Dinda. Automatic dynamic run-time optical network reservations. In *Proceedings of* 14th *IEEE International Symposium on High-Performance Distributed Computing (HPDC) 2005*, pages 255–264, Research Triangle Park, NC, USA, July 24 27, 2005.

[17] J. Liedtke. μ-kernels must and can be small. In *Proceedings of* 5th *IEEE International Workshop on Object-Orientation in Operating Systems (IWOOOS) 1996*, pages 66–77, Seattle, WA, Oct. 27-28, 1996.

[18] J. Moreira, M. Brutman, J. Castanos, T. Gooding, T. Inglett, D. Lieber, P. McCarthy, M. Mundy, J. Parker, B. Wallenfelt, M. Giampapa, T. Engelsiepen, and R. Haskin. Designing a highly-scalable operating system: The Blue Gene/L story. In *Proceedings of International Conference on High Performance Computing, Networking, Storage and Analysis (SC) 2006*, Tampa, FL, USA, Nov. 11-17, 2006.

[19] J. Mugler, T. Naughton, S. L. Scott, B. Barrett, A. Lumsdaine, J. M. Squyres, B. des Ligneris, F. Giraldeau, and C. Leangsuksun. OSCAR clusters. In *Proceedings of* 5th *Annual Ottawa Linux Symposium (OLS) 2003*, Ottawa, Canada, July 23-26, 2003.

[20] Netlib Repository at University of Tennessee, Knoxville, and Oak Ridge National Laboratory, TN, USA. Basic Linear Algebra Subprograms (BLAS). Available at http://www.netlib.org/blas, 2007.

[21] Netlib Repository at University of Tennessee, Knoxville, and Oak Ridge National Laboratory, TN, USA. Linear Algebra PACKage (LAPACK). Available at http://www.netlib.org/lapack, 2007.

[22] Northwestern University, Evanston, IL. Virtuoso: Resource Management and Prediction for Distributed Computing Using Virtual Machines. Available at http://virtuoso.cs.northwestern.edu, 2007.

[23] Oak Ridge National Laboratory, Oak Ridge, TN, USA. Harness Workbench Project. Available at http://www.csm.ornl.gov/harness, 2007.

[24] Open MPI Project, 2007. Available at http://www.open-mpi.org.

[25] M. Snir, S. Otto, S. Huss-Lederman, D. Walker, and J. Dongarra. *MPI: The Complete Reference*. MIT Press, Cambridge, MA, USA, 1996.

[26] A. I. Sundararaj and P. A. Dinda. Towards virtual networks for virtual machine Grid computing. In *Proceedings of* 3rd *Virtual Machine Research and Technology Symposium (VM) 2004*, pages 177–190, San Jose, CA, USA, May 6-7, 2004.

[27] UnionFS: A Stackable Unification File System, 2007. Available at http://www.am-utils.org/project-unionfs.html.

[28] G. Vallée, T. Naughton, and S. L. Scott. System management software for virtual environments. In *Proceedings of ACM International Conference on Computing Frontiers (CF) 2007*, Ischia, Italy, May 7-9, 2007.

[29] G. V. Vaughan, B. Elliston, T. Tromey, and I. L. Taylor. *GNU Autoconf, Automake and Libtool*. New Riders Publishing, Thousand Oaks, CA, USA, 2000. Available at http://sourceware.org/autobook.

[30] Virtual Workspaces Project, 2007. Available at http://workspace.globus.org/.

[31] VMware, Inc., Palo Alto, CA, USA. VMware virtualization products. Available at http://www.vmware.com/, 2007.

[32] XenSource, Inc., Palo Alto, CA, USA. Open Source Xen Hypervisor Technology. Available at http://www.xensource.com, 2007.