Virtualized Environments for the Harness High Performance Computing Workbench*

B. Könning[1,2], C. Engelmann[1,2], S. L. Scott[1], and G. A. Geist[1]
[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
{koenningb,engelmannc,scottsl,gst}@ornl.gov
{b.koenning,c.engelmann}@reading.ac.uk

Abstract

This paper describes recent accomplishments in providing a virtualized environment concept and prototype for scientific application development and deployment as part of the Harness High-Performance Computing (HPC) Workbench research effort. The presented work focuses on tools and mechanisms that simplify scientific application development and deployment tasks, such that only minimal adaptation is needed when moving from one HPC system to another or after HPC system upgrades. The overall technical approach focuses on the concept of adapting the HPC system environment to the actual needs of individual scientific applications instead of the traditional scheme of adapting scientific applications to individual HPC system environment properties. The presented prototype implementation is based on the mature and lightweight `chroot` *virtualization approach for Unix-type systems with a focus on virtualized file system structure and virtualized shell environment variables utilizing virtualized environment configuration descriptions in Extensible Markup Language (XML) format. The presented work can be easily extended to other virtualization technologies, such as system-level virtualization solutions using hypervisors.*

1. Introduction

While hardware, system software, and scientific applications for high-performance computing (HPC) continue to rapidly improve in performance and efficiency from current terascale to next-generation petascale machines, scientific application development and deployment activities performed by application scientists are hampered by frequent HPC system upgrades and new installations, constantly changing hardware and software environments, and a significant diversity in HPC system environments.

HPC system hardware upgrades and new HPC system installations have become annual or even semi-annual events for many HPC centers. Similarly, HPC system software upgrades have become monthly or even semi-monthly events. There is a constant need to port, recompile, and/or retune scientific applications to new or upgraded systems. Additionally, new scientific applications are either developed and tested to fit to the current generation HPC systems and need to be ported once they go into production, or they are developed for the next-generation HPC systems without access to respective testbeds.

Annual or semi-annual HPC system upgrades and new installations incur the highest overhead due to the significant change in hardware architecture and system software. Porting, recompiling, and retuning existing or newly developed scientific applications is still a complex task requiring HPC system and HPC center specific knowledge, such as:

- Where to deploy scientific application packages (sources and binaries)?
- Which compiler/linker and compiler/linker flags to use?
- Does the system perform cross-compilation?
- Which system libraries to link and where to find them?

*This research is sponsored by the Office of Advanced Scientific Computing Research; U.S. Department of Energy. The work was performed at the Oak Ridge National Laboratory, which is managed by UT-Battelle, LLC under Contract No. De-AC05-00OR22725.

- How to find and use dependent software packages?
- Which system-specific workarounds to use?
- What needs to be in the batch job script?

HPC centers typically house multiple different machines. In addition to the high upgrade/installation frequency of individual systems, there is a significant diversity within and between HPC centers in deployed system architectures, hardware, interconnects, operating systems, compilers, library versions, runtime environments, and debugging and monitoring tools. The deployment of scientific applications involves customization not only to various HPC platforms, but also to HPC center specific environments.

The Harness HPC Workbench [15] project at Oak Ridge National Laboratory, the University of Tennessee, and Emory University focuses on tools and mechanisms to simplify scientific application development and deployment tasks, such that only minimal adaptation is needed when moving from one HPC system to another or after HPC system upgrades. The ultimate goal is to improve the productivity of application scientists in order to allow them to focus more on actual science instead of software development and deployment activities.

This paper describes recent accomplishments at Oak Ridge National Laboratory in collaboration with the University of Reading in providing a virtualized environment concept and prototype for scientific application development and deployment as part of the Harness HPC Workbench research effort. We begin with an illustration of the overall technical approach. We continue with a presentation of the detailed software design of the developed prototype and a discussion of obtained experimental results. We go on with a description of related past and ongoing work in this area. This paper concludes with a short summary of the presented research and a brief overview of future work.

2. Technical Approach

In order to provide seamless scientific application development and deployment across various HPC systems and HPC centers, our overall technical approach focuses on the concept of adapting the HPC system environment to the actual needs of individual scientific applications instead of the traditional scheme of adapting scientific applications to individual HPC system environment properties. By using virtualization technology, scientific applications can be developed and deployed inside a virtualized environment specifically adapted to their requirements. Virtualized environments can be deployed on a large number of supported HPC systems at various HPC centers, while providing the same view to scientific applications. The overall goal of this approach is to provide portable virtualized environments for scientific applications using efficient adaptation to physical HPC system environments.

The Harness HPC Workbench effort initially focuses on mature, lightweight virtualization technologies for Unix-type systems with an emphasis on virtualized file system structure and virtualized shell environment variables. This limitation ensures obtaining production-type experience from scientific application developers and system administrators without dealing with the software stability issues resulting from an immaturity of the virtualization layer, the performance impact caused by a more than insignificant runtime overhead of the virtualization layer, and the workarounds needed for lightweight operating systems that deal with unsupported functionality.

The virtualization approach for the file system structure is based on the well-known `chroot` command, which is available on Unix-type systems and provides support for a virtualized file system structure by changing the root directory of a given process into a different given directory. Since a significant part of the configuration of a Unix-type system depends on its root file system structure, individual adaptation to scientific application needs can be achieved by providing customized root file system structures for scientific application processes. The `chroot` virtualization technology is known for its negligible to minimal runtime performance overhead for applications running inside the virtualized environment.

The virtualization approach for the shell environment variables is based on creation and modification of shell environment variables after executing the `chroot` command, before executing the actual scientific application.

In order to provide for portability across various HPC systems and HPC centers, virtualized environment configurations are described in Extensible Markup Language (XML) format. These XML virtualized environment configuration descriptions contain all necessary information about virtualized file system structure and virtualized shell environment variables. Each virtualized environment configuration description includes adaptation information for a targeted HPC system and a specific scientific application, while a hierarchical inheritance scheme allows system administrators to provide standardized XML virtualized environment configurations for common use cases that can be extended with application specific requirements.

In the Harness HPC Workbench virtualization approach, a scientific application is implemented on a development platform and deployed to a different HPC

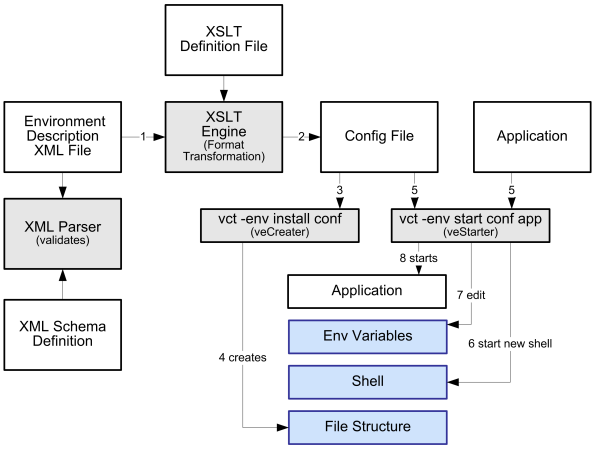

Figure 1. Architecture and Workflow of the Virtualized Environment Approach for the Harness High Performance Computing Workbench

system by providing a virtualized environment configuration description and by executing respective commands for virtualized environment installation and scientific application execution inside an installed virtualized environment (see Figure 1). The virtualized environment commands are part of the Harness HPC Workbench and operate using intermediate implementation-specific configuration files in order to separate XML parsing from virtualized environment operations. This permits operation without requiring an XML parser for each particular HPC system and without a repetitive XML parsing performance overhead. The translation from XML to intermediate configuration files is only performed once for each specific virtualized environment configuration description and may be performed on a different system, such as the scientific application development system, using implementation-specific Extensible Stylesheet Language Family Transformation (XSLT) definitions.

When moving a scientific application from one HPC system to another or after an HPC system upgrade, a different or modified XML virtualized environment configuration description is needed with HPC system specific adaptation information. This adaptation method assures that particular HPC system properties are considered for the adaptation to scientific application needs, thus avoiding coarse-grain HPC system abstraction and profiting from HPC system specific advantages, such as tuned numeric libraries.

In order to minimize the overhead of providing different XML virtualized environment configuration descriptions for each scientific application and for each HPC system, the hierarchical inheritance scheme may be used to modify existing XML virtualized environment configuration descriptions for common use cases provided by system administrators. These XML virtualized environment configuration templates may contain HPC system and HPC center specific knowledge and may be automatically updated by system administrators after HPC system upgrades. They may also enforce HPC system and HPC center specific policies within virtualized environments.

3. Detailed Design

The implemented prototype consists of an XML schema, an XSLT definition, and two separate Unix-shell commands, `veCreator` and `veStarter`. The provided XML schema allows the validation of XML virtualized environment configuration descriptions, while the XSLT definition permits the translation of XML virtualized environment configuration descriptions into a Unix-shell format. The virtualized environment management commands, `veCreator` and `veStarter`, use the Unix-shell virtualized environment configuration description to, respectively, create a virtualized environment, *i.e.*, to prepare its file structure, and to start an application inside the virtualized environment, *i.e.*, to `chroot` into the prepared file structure and to create/modify shell variables.

3.1. Virtualized Environment Creation

Based on the the virtualized environment configuration description, the `veCreator` command prepares the file structure of a virtualized environment into a "`chroot`-able" file structure, *i.e.*, into a virtualized root directory structure containing parts of the original HPC system root directory structure and additional modifications that perform the actual adaptation to specific scientific application needs, such as additional/renamed libraries or modified configuration files. The virtualized root directory structure modifications are at the file and directory level. Modifications at the software package level and support for operating system distribution images may be added at a later stage.

The current virtualized environment prototype supports three methods for incorporating files and directories into the virtualized root directory structure of a virtualized environment: copy, link, and UnionFS.

The copy method represents the easiest way to incorporate files and directories into the virtualized root directory structure of a virtualized environment. A file or an entire directory is copied from its original location to the target location inside the virtualized root directory structure. The copy method protects the original file or directory from operations inside the virtualized environ-

ment. This sandbox characteristic applies to most use cases for virtualized environments. However, copying large parts of the original HPC system root directory structure into every virtualized root directory structure incurs a high performance overhead for virtualized environment creation and results in a significant usage of file system storage resources.

As a lightweight alternative, the link method uses symbolic links instead of copying files and directories. While this method significantly reduces the performance overhead for virtualized environment creation and substantially saves file system storage resources, its disadvantage lies in the direct connection between source and target files and directories. The properties of an incorporated file or directory, such as its ownership and permissions, cannot be changed independently from the source. Furthermore, modifications of the target are reflected in the source, thus eliminating the sandbox characteristic. However, partially eliminating the sandbox characteristic may be useful to reflect existing HPC system and HPC center specific policies, such as existing user accounts and access rights, within virtualized environments.

Due to the fact that the file system virtualization approach is based on `chroot`, symbolically linking target files and directories to sources outside a "`chroot`-ed" file structure requires to mount the original root directory structure within the virtualized root directory structure, such that a directory `/home/peter /env1/.osroot` is the original root directory structure within the virtualized environment `env1` of `peter` using the `mount --bind / home/peter/env1/.osroot` command line within the `veCreator` command. This necessary workaround for the link method further eliminates the virtualized environment sandbox characteristic as the entire original root file system is made visible within a virtualized environment as the `/.osroot` directory.

The third supported method for incorporating files and directories into the virtualized root directory structure of a virtualized environment utilizes recent advances in stackable file systems. UnionFS [24, 22] is a file system driver for Unix-type systems, which allows the unification of several directories into one virtual directory by combining their content. The content of a union are the merged files and subdirectories of its branches. For a user, a union looks like a normal directory. Even branches from different file systems can be merged into one union. UnionFS also supports copy-on-write, hide-on-delete, and configurable access limitations. However, UnionFS has also certain limitations, such as the missing `mmap` support and runtime performance impact. Furthermore, UnionFS is still in development and considered an immature software product by many HPC centers and therefore not widely deployed in HPC systems.

Source	Connection	Target	Method
rw	static	rw	Copy or UnionFS with Copy-on-Write
rw	static	ro	Copy
ro	static	rw	Copy or UnionFS with Copy-on-Write
ro	static	ro	Copy
rw	dynamic	rw	Link
rw	dynamic	ro	UnionFS with Read-Only
ro	dynamic	rw	Not Supported
ro	dynamic	ro	Not Supported

Table 1. Comparison of Methods for Incorporating Files and Directories into the Virtualized Root Directory Structure with Respect to Connection and Permissions

The comparison of methods for incorporating files and directories into the virtualized root directory structure with respect to connection and permissions in Table 1 shows the various possible combinations and use cases. All described methods refer to sources located in the local original root directory structure. Support for remote sources exists for the copy method by using Uniform Resource Locators (URLs) and respective access methods, such as `scp` or `wget`.

Since software package managers typically do not allow multiple versions of the same software to be installed simultaneously, a common technique that has been adapted by many HPC centers [13] uses a unified naming scheme for software installations on a shared networked file system, such as `/apps/<package>/<version>/<machine>`. Integrating these software packages into the virtual file system structure can be easily performed using XML virtualized environment configuration templates supplied by the HPC center. Multiple templates can be combined through the `include` feature of the virtualized environment configuration schema, such that the XML virtualized environment configuration for an application just includes the templates of its dependent software packages. This approach is quite similar to the Modules approach used to date (see Sec. 5), while it adds the virtualized file system capability.

3.2. Virtualized Environment Invocation

Based on the the Unix-shell virtualized environment configuration description, the `veStarter` command "`chroot`-s" into the prepared virtualized root directory structure, creates and modifies shell variables accordingly, and starts the given scientific application. Upon termination of the scientific application process, the `veStarter` command terminates as well.

HPC systems typically use a parallel start mechanism to spawn the scientific application processes on compute nodes in a single program multiple data (SPMD) fashion. Instead of directly executing the scientific application, the `veStarter` command is executed on the compute nodes by the parallel start mechanism, such that the `veStarter` wraps the scientific application including its command line arguments. This wrapping can be easily integrated with batch job management solutions for HPC systems, such that the invocation of MPI [17] via `mpirun -np program [options...]` is automatically wrapped into `mpirun -np veStarter config program [options...]`, where `config` identifies the virtualized environment configuration description file that may be retrieved from a shell environment variable during job submission.

3.3. System Security Aspects

Since the execution of the `mount` and `chroot` commands requires super-user rights, the well-known and secure `sudo` mechanism is used to give the `veCreator` and `veStarter` commands permission to execute, respectively, `mount` and `chroot` with super-user rights, while continuing to deny direct access to these commands to normal users. In order to securely execute a scientific application, the `veStarter` command changes back to the original user context after executing `chroot` by executing `su $ORIGINAL_USER`.

Certain system security vulnerabilities exist related to the properties of a virtualized environment, such that original system configuration files, like passwords, can be compromised by extending read or even write access within the virtualized environment. Furthermore, original system configuration files may be replaced entirely in the virtualized environment. In the first case, relaxed sandbox characteristics allow leaking of original system configuration files to the virtualized environment without appropriate protection, while strong sandbox characteristics fail to enforce the existing system configuration in the second case. A careful balance between relaxed and strong sandbox characteristics is required for optimal system security and virtualized environment usability. Templates for virtualized environment descriptions are able to assist system administrators and help enforcing HPC system and center policies by providing preconfigured virtualized environment descriptions.

Additional system security aspects arise from the fact that virtualized environments typically need to follow the same cyber security rules like real environments. System administrators may be reluctant to support virtualized environments if they cannot assure that existing policies are not circumvented. A verification (or even certification) of the presented prototype tools and of specific virtualized environment configuration descriptions is needed for production-type deployment with cyber security policies.

4. Experimental Results

Based on the developed prototype implementation, numerous functional and several performance tests were conducted in order to check for errors and efficiency. Both, the `veCreator` and `veStarter` commands provided correct functionality based on various XML virtualized environment descriptions that were validated against the XML schema and with the help of the XSLT definition translated into Unix-shell virtualized environment descriptions.

Two separate performance test series were conducted, one for virtualized environment creation using `veCreator` and one for virtualized environment invocation using `veStarter`. Both test series focused on the different methods for incorporating files and directories into the virtualized root directory structure with respect to creation and runtime overhead.

The virtualized environment creation tests in Table 2 involved 32935 files out of the directories `/bin` `/lib`, `/sbin` and `/etc` of a Fedora Core 6 Linux installation. The virtualized environment creation was performed in 65 seconds using the copy method and 5-6 seconds for the link and UnionFS methods.

Method	Creation	Access	Read/Write
Copy	65s	95%	100%
Link	5-6s	94%	100%
UnionFS	5-6s	94%	60-99%

Dual Pentium D 3.4 GHz, 4GB RAM, Western Digital WD2500JS, Linux 2.6.15, ext3, UnionFS 1.3

Table 2. Comparison of Methods for Incorporating 32935 Files and Directories into the Virtualized Root Directory Structure with Respect to Virtualized Environment Creation Time and File Access/Read/Write Runtime Performance

The virtualized environment invocation tests in Table 2 involved various file system access benchmarks, such as a small benchmark script that opens a number of files, Iozone, Postmark, and kernel source compilation. The tests revealed a 6-7% performance hit just for being in a "`chroot`-ed" environment for all methods for rapid file access, such as opening a large number of files. However, this performance hit is negligeble as the time spend for opening files is typically significantly smaller in comparison to the time spend reading and writing files. There was virtually no runtime performance difference between the copy and link methods with the exception of a caching effect of 1%. There was also no noticable read/write performance difference for the copy and link methods. As expected, there was a read/write performance difference for the UnionFS method. Using the I/O-intensive Postmark benchmark, the introduced overhead can be as big as 70%. However, using the CPU-intensive kernel source compilation benchmark, the overhead was as small as 1%.

These tests were performed on a single compute node using the local file system in order to measure the maximum performance impact of the virtualized environment. More details and an analysis about the performed functional and performance tests are available in the Master thesis documenting this research and development effort [10].

5. Related Work

The Modules software package [18] allows to dynamically modify a user environment by using module configuration files. Each module configuration file contains the information needed to configure the shell for an application. After the Modules software package is initialized, the environment can be modified on a per-module basis using the module command, which interprets module configuration files. Typically, these configuration files instruct the module command to alter or set shell environment variables such as `PATH`, `MANPATH`, and others.

While the Modules software package is heavily used in HPC environments [14, 12], its approach focuses on a virtualized shell environment only. The approach presented in this paper extends the Modules concept to a virtualized file system directory structure.

The recent interest in virtualization technologies was a direct result of advances in lightweight type-I hypervisor solutions, such as Xen.

Xen [1, 23] is an open source virtual machine monitor (VMM) for IA-32, x86-64, IA-64, and PowerPC architectures. Its type-I system-level virtualization allows one to run several virtual machines (VMs) in an 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.

The presented work can be easily extended to other virtualization technologies, such as system-level virtualization solutions using hypervisors. An ongoing project at Oak Ridge National Laboratory targets the development of virtualized system environment (VSE) support for scientific computing using system-level virtualization. While the presented virtualized environment concept allows specifying the runtime environment (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. A recent paper [4], proposes the VSE concept, explains its technical approach, and evaluates related research. We refer the reader to this paper for a more extensive description of related work in the area of system-level virtualization.

The Open Source Cluster Application Resources (OSCAR) toolkit is used to build and maintain HPC clusters [11]. The toolkit has been recently extended to support system-level virtualization technology, *e.g.*, Xen and QEMU. This virtualization-enhanced version, OSCAR-V [21], 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.

Our ongoing work focuses on integrating the file-level configuration approach for virtualized environments presented in this paper with package-level VM configuration tools, like OSCAR-V.

The Virtual Workspaces project [8, 19] is an effort to exploit virtualization technology for the Grid [9]. The goal is to capture the requirements for an exe-

cution 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. The Virtual Workspaces effort focuses on a resource oriented aspect in small-scale distributed cooperative computing environments.

The presented work also relates to past and ongoing research and development efforts in simplifying software development and deployment.

The GNU Autotools [5, 6, 7] are part of an open source suite designed to assist in providing portability to source code packages for many Unix-like systems. The GNU Autotools are a collection of automated approaches to detect system properties for correctly compiling and installing applications. They support the detection of various shell environments, tools, and libraries for a wide variety of target systems. In addition, cross-compiling, *i.e.*, compilation for a target system on a different system, is supported.

NetBuild [20] is a suite of tools designed to aid in using computational software libraries that are stored on the network, without the need to have them preinstalled on a system. Instead, NetBuild will determine which libraries are not installed, identify suitable versions that are accessible from the network, and download and link them with the application. NetBuild is developed by the ICL Team at the University of Tennessee and runs on most UNIX-type platforms and also on Windows.

The Eclipse Integrated Development Environment (IDE) and its Parallel Tools Platform (PTP) [2, 3] are an open source solution for parallel application development and debugging. Eclipse offers many features expected from a commercial quality IDE, such as syntax-highlighting editor, source-level debugger, revision control, code refactoring, and support for multiple languages, including C, C++, and Fortran. PTP allows application developers to use Eclipse as a portable IDE across a wide range of parallel systems. PTP also includes a scalable parallel debugger, and tools for development of parallel programs. Eclipse already supports XML configurations for different workspaces.

Ongoing work in the Harness HPC Workbench project focuses on integrating the presented solution with Eclipse and PTP to add virtualized environment support to Eclipse workspaces in scientific application development scenarios.

Further recent and ongoing work in the Harness HPC Workbench project targets the porting process of applications in scientific HPC environments, especially in the context of legacy applications that have been developed over a number of years or even decades. In contrast to the approach presented in this paper, which allows adapting HPC system properties to scientific application needs using virtualization, this complementary effort aims at adapting scientific applications to HPC system properties using source code level porting transformations [16].

6. Conclusions and Future Work

This paper describes recent research and development efforts at Oak Ridge National Laboratory in collaboration with the University of Reading in providing a virtualized environment concept and prototype for scientific application development and deployment as part of the Harness HPC Workbench research effort. The main motivation is to offer tools and mechanisms to simplify scientific application development and deployment tasks, such that the overall productivity of application scientists increases.

The technical approach focuses on the concept of adapting the HPC system environment to the actual needs of individual scientific applications instead of the traditional scheme of adapting scientific applications to individual HPC system environment properties. It utilizes `chroot` virtualization in combination with shell environment wrapping in Unix-type environments to provide for virtualized file system structure and virtualized shell environment variables. Virtualized environment configurations in XML format assure portability across various HPC systems and HPC centers by containing adaptation information for a targeted HPC system and a specific scientific application. A hierarchical inheritance scheme allows for XML virtualized environment configuration templates for common use cases and for enforcing site specific policies. The following HPC system and HPC center specific knowledge can be described in XML virtualized environment configuration templates using virtual file and directory paths:

- Where to deploy scientific application packages (sources and binaries)?
- Where to find system libraries?
- Where to find dependent software packages?

Since virtual file and directory paths are used, the impact of upgrading system libraries and dependent software packages can be configured such that (1) an upgrade is immediately visible in the virtualized environment, (2) an upgrade is not visible at all, and (3) an upgrade is only visible through a different virtual file or directory path.

The performance results show that the developed prototype is ready for testing in the field with the exception of the UnionFS method for incorporating files and directories into the virtualized root directory structure of a virtualized environment. However, we expect this issue to disappear in the future with increasing maturity of the UnionFS stackable file system driver.

Ongoing efforts focus on obtaining production-type experience from scientific application developers and system administrators with respect to the usability of the developed prototype and related system security issues and policies.

Future work will target the integration with development and runtime environments for scientific computing, such as Eclipse and PTP, and the continuation of the expansion to system-level virtualization using the VSE approach.

References

[1] 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.

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

[3] Eclipse Foundation Inc. Eclipse Parallel Tools Platform, 2007. Available at http://www.eclipse.org/ptp/.

[4] C. Engelmann, S. L. Scott, H. Ong, G. Vallée, and T. Naughton. Configurable virtualized system environments for high performance computing. In *Proceedings of the 1st Workshop on System-level Virtualization for High Performance Computing (HPCVirt) 2007, in conjunction with the 2nd ACM SIGOPS European Conference on Computer Systems (EuroSys) 2007*, Lisbon, Portugal, Mar. 20, 2007.

[5] Free Software Foundation, Inc. GNU Autoconf Project. Available at http://www.gnu.org/software/autoconf/, 2007.

[6] Free Software Foundation, Inc. GNU Automake Project. Available at http://www.gnu.org/software/automake/, 2007.

[7] Free Software Foundation, Inc. GNU Libtool Project. Available at http://www.gnu.org/software/libtool/, 2007.

[8] 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.

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

[10] B. Könning. Virtualized environments for the Harness Workbench. Master's thesis, Department of Computer Science, University of Reading, UK, Mar. 14, 2007.

[11] 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.

[12] National Center for Computational Sciences, Oak Ridge, TN, USA. Modules Software Package Description. Available at http://info.nccs.gov/resources/general/software/modules/, 2007.

[13] National Center for Computational Sciences, Oak Ridge, TN, USA. Software Package Naming Scheme. Available at http://info.nccs.gov/resources/general/software/, 2007.

[14] National Energy Research Scientific Computing Center, Berkeley, CA, USA. Modules Approach to Software Management. Available at http://www.nersc.gov/nusers/resources/software/os/modules.html, 2007.

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

[16] J. Sławiński, M. Sławińska, and V. Sunderam. Porting transformations for HPC applications. In *ISCA 20th International Conference on Parallel and Distributed Computing Systems (PDCS) 2007*, Las Vegas, NV, USA, Sept. 24-26, 2007.

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

[18] Sourceforge. Modules Software Package. Available at http://modules.sourceforge.net/, 2007.

[19] The Globus Alliance. Virtual Workspaces Project, 2007. Available at http://workspace.globus.org/.

[20] University of Tennessee, Knoxville, TN, USA. NetBuild Project, 2007. Available at http://icl.cs.utk.edu/netbuild/.

[21] 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*, pages 153–160, Ischia, Italy, May 7-9, 2007.

[22] C. P. Wright, J. Dave, P. Gupta, H. Krishnan, D. P. Quigley, E. Zadok, and M. N. Zubair. Versatility and unix semantics in namespace unification. *ACM Transactions on Storage (TOS)*, 2(1):1–32, Feb. 2006.

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

[24] E. Zadok *et al.* UnionFS: A Stackable Unification File System. Available at http://www.am-utils.org/project-unionfs.html, 2007.