System-Level Virtualization Research at Oak Ridge National Laboratory[1]

Stephen L. Scott, Geoffroy Vallée, Thomas Naughton,
Anand Tikotekar, Christian Engelmann, Hong Ong

Oak Ridge National Laboratory, Oak Ridge, TN 37830, USA

Abstract

System-level virtualization is today enjoying a rebirth, after first gaining popularity in the 1970s as a technique to effectively share what were then considered large computing resources to subsequently fade from the spotlight as individual workstations gained in popularity with a "one machine – one user" approach. One reason for this resurgence is that the simple workstation has grown in capability to rival that of anything available in the past. Thus, computing centers are again looking at the price/performance benefit of sharing that single computing box via server consolidation.

Hardware and software technology vendors have noticed this renewed interest as well and have responded with a variety of technologies designed to enable advanced virtualization capabilities. However, industry is only concentrating on the benefits of using virtualization for server consolidation (enterprise computing) whereas our interest is in leveraging virtualization to advance high-performance computing (HPC). While these two interests may appear to be orthogonal, one consolidating multiple applications and users on a single machine while the other requires all the power from many machines to be dedicated solely to its purpose, we propose that virtualization does provide attractive capabilities that may be exploited to the benefit of HPC interests. This does raise the two fundamental questions of: is the concept of virtualization (a machine "sharing" technology) really suitable for HPC and if so, how does one go about leveraging these virtualization capabilities for the benefit of HPC.

To address these questions, this document presents ongoing studies on the usage of system-level virtualization in a HPC context. These studies include an analysis of the benefits of system-level virtualization for HPC, a presentation of research efforts based on virtualization for system availability, and a presentation of research efforts for the management of virtual systems. The basis for this document was material presented by Stephen L. Scott at the Collaborative and Grid Computing Technologies meeting held in Cancun, Mexico on April 12-14, 2007.

Key words: virtualization, high-performance computing, fault tolerance, system management

1 Introduction to System-Level Virtualization

System-level virtualization is used for a number of reasons, but the three major justifications are [1–3]: (i) *isolation*, (ii) *consolidation*, and (iii) *migration* [2]. We describe these points in the following sections after a brief description of the terminology used in this article.

Terminology The execution of a virtual machine (VM) implies that one or more virtual systems are running concurrently on top of the same hardware, each having its own view of available resources. The operating system (OS) of the VMs is referred to as the *guest OS*. This system is a traditional OS, but does not necessarily see all the available physical resources. The guest OS only views resources that have been allocated to the VM. VMs hosted on the same machine are run concurrently with the *hypervisor* managing the concurrent execution. The hypervisor is responsible for mapping the physical resources to those used by the virtual machines. Due to its role as a VM manager, the hypervisor is also called the *Virtual Machine Monitor* (VMM). The hypervisor is typically a small system running beside the VMs. The hypervisor typically does not include drivers or other device specific mechanisms to access the physical hardware, e.g., network cards or hard drives. Therefore, the hypervisor is coupled with a traditional operating system, called the *host OS*. This host OS provides users log-in access for the administration of physical resources and an interface to the hypervisor for VM management.

1.1 Virtualization Capabilities

Isolation Isolation is aiming at improving the security and reliability of the system by isolating the execution environment for application in a VM which cannot corrupt the bare hardware[1,3]. For that the virtualization solution

Email addresses: scottsl@ornl.gov (Stephen L. Scott), valleegr@ornl.gov (Geoffroy Vallée), naughtont@ornl.gov (Thomas Naughton), tikotekaraa@ornl.gov (Anand Tikotekar), engelmannc@ornl.gov (Christian Engelmann), hongong@ornl.gov (Hong Ong).

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

exposes to VMs a virtual hardware, *i.e.*, a VM is limited to execution of unprivileged instructions with the hypervisor overseeing all other operations.

Consolidation Server consolidation is the primary market for virtualization solutions, because it enables the sharing of expensive servers between different customers with the guarantee that each customer will have its own view of the system and be isolated from other users. This allows service providers to consolidate work to fewer servers (cost effective server usage) and also to support incompatible or legacy operating environments without the need for separate hardware.

Virtual Machine Migration The capability to migrate entire VMs between servers, makes it possible to improve the quality of service by balancing the global load between several servers without interruption of application execution and by moving VMs (and therefore applications) when a failure is predicted for a specific server. This technology also enables a transparent (for users) programmable downtime of servers by migrating VMs to other servers before server shutdown for maintenance.

1.2 Classification

In the 1970s, Goldberg [4] classified the different system-level virtualization solutions into two categories (see Fig. 1): (i) *type-I virtualization* where the VMM runs directly on the bare hardware, and (ii) *type-II virtualization* where the VMM runs on top of the host OS. Since the type-I virtualization has direct access to resources, performance is comparable to that of native execution. In contrast, type-II virtualization incurs additional overhead due to the layering of the VMM on top of the host OS when servicing resource requests from VMs. The type-II is well suited for development, where some performance may be reduced in exchange for greater diagnostic and development capabilities.

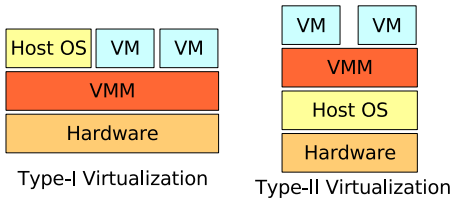

Fig. 1. Classification of Virtualization Techniques

A third hybrid form called *para-virtualization* [1,3] has emerged that fits the criteria of *type-I* virtualization. In para-virtualization, modifications are made to the host OS and guest OS that are essentially a new software-only architecture. This additional work provides the improved performance of *type-I*.

2 Why System-Level Virtualization for High-Performance Computing?

Today, high-performance computing (HPC) centers need to support multiple execution platforms. For example, at Oak Ridge National Laboratory (ORNL), massively parallel processing (MPP) platforms, such as the Cray XT4, and Beowulf-type clusters, like the ORNL Institutional Cluster (OIC), are available to users. Each of these systems targets a specific OS, requiring users to *port* their application before execution. On the other hand, a user's requirements may also differ. For instance, some users develop their applications on desktops, others on clusters; including different needs in terms of hardware and software requirements. Finally, each application has its own hardware and software constraints.

It is therefore very difficult to provide a single execution environment for all applications that is supported by quite a variety of development environments. To address these issues, the notion of *plug-and-play computing* and *execution environment customization* have been introduced. The idea is too let users specify their needs in terms of system environment and then deploy on demand this environment in virtual machines. The deployment of an application on a new execution platform is therefore direct (plug-and-play).

However, current virtualization solutions are not suitable for HPC for a number of reasons including: (i) their system footprint is significant and will interfere with application execution and performance; (ii) current solutions support base Intel and AMD architectures but do not fully support their use in some of the more exotic HPC specific architectures; (iii) because today's primary target for virtualization is enterprise server consolidation, many capabilities required specifically for HPC are slow to emerge (*e.g.*, VMM-bypass [5], RDMA [6]); and (iv) the architecture of virtualization solutions is monolithic and does not allow dynamic configuration of the VMM.

In order to have a virtualization solution suitable for HPC, many approaches are possible: (i) development of a new solution from scratch, (ii) development of a new solution based on an existing solution, such as Xen, or (iii) development of a new solution based on a HPC specific operating system, such as Catamount.

Each of these approaches have its own advantages and drawbacks. For instance, the development of a new solution from scratch allows the design and implementation of a solution to perfectly meet HPC constraints. However, such a solution is a long term effort because it implies kernel-level development which is challenging. The development of a new solution based on an existing solution (which may be an already existing VMM and a kernel for

HPC) gives more short term results but may be difficult to maintain since it was not originally designed for the HPC environment.

3 System-Level Virtualization and System Availability

Modern high-performance computing platforms are composed of thousands or even hundreds of thousands of nodes. Because each node can be subject to a failure, the global availability of the system decreases in proportion to the system scale. Therefore, applications for this environment must be fault tolerant or resilient, able to operate successfully in the face of failure, and the systems should exhibit high-availability traits.

System-level virtualization provides three interesting capabilities that may be exploited for this purpose: (i) VM migration, (ii) VM pause/unpause, and (iii) VM checkpoint/restart. These three mechanisms enable the implementation of three fault tolerance policies: (i) reactive fault tolerance (do something after a failure occurs), (ii) proactive fault tolerance (do something before the failure occurs), and (iii) hybrid policies mixing both reactive and proactive fault tolerance.

The reactive approach fundamentally relies on the concept that there is sufficient contextual information available (checkpoint data), such that it is possible to restart an application after a failure occurs and recover close to the point of failure so that little computation is lost. Because each VM and the applications running on it is a well defined entity, isolated from all other applications and even from the core host operating system, it is possible to checkpoint the entire VM and then subsequently restart it in that original state without any prior knowledge on the part of the applications. This is possible because the context of a VM typically consists of the memory dump and a checkpoint of the file system. While all current virtualization solutions can already dump the memory of the VM (the hypervisor is in charge of the memory management of VMs), the file system is quite a different matter. This is due to the need for successive snapshots of the complete file system. One solution to address this issue is to use stackable file systems [7] for which the snapshot capability is used to create periodic checkpoints of the file system (see Fig. 2). This solution is expensive, since periodic checkpointing generates a significant amount of I/O.

The proactive approach is based on the idea that it is possible to predict some failures, for instance using hardware probes that are available on many modern motherboards. With virtualization, if a failure is predicted, two scenarios are possible: (i) VMs are migrated away from the node that is about to fail, (ii) the affected VMs are paused on a stable storage in order to be resumed later

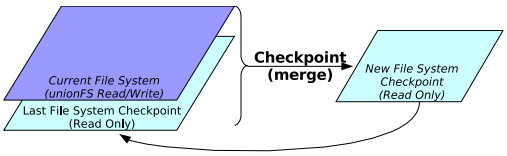

Fig. 2. Virtual Machine File System Checkpoint

when the failure is fixed.

However, proactive fault tolerance is not always sufficient because all failures cannot be predicted. Thus, it is important to be able to mix reactive and proactive fault tolerance, creating a hybrid policy. The idea is to decrease the checkpoint frequency (and therefore the checkpoint penalty) based on the ratio of predictable failures. Therefore, the proactive approach is used for predictable failures, and the reactive approach for unpredictable failures.

Because it is not possible to find a one-fit-all solution (*i.e.*, a solution that fits the application requirements, the computing center policy requirements, the users expectations, and the execution platforms characteristics), a fault tolerance framework, which allows users, system administrators, and centers to specify their own fault tolerance policies, is of particular interest.

4 System-Level Virtualization and System Management

The usage of VMs creates several challenges including: (i) how can we support multiple virtualization solutions? (ii) how can we easily manage both, the host OS and the VMs? and, (iii) is it possible to abstract the complexity of virtualization?

Several studies has been recently made to address these issues, which led to the implementation of OSCAR-V [8]. As an extension for the management of VMs using the OSCAR system installation/management suite, it integrates several prototypes developed by ORNL and by our collaborators. OSCAR-V is based on two concepts: (i) the abstraction of the notion of VMs via virtual machine management (V2M) and (ii) the implementation of tools for the definition of golden images that can be used to deploy both, host OSs and VMs.

The V2M tool aims to provide a very simple interface to users for the definition and the management of VMs; users need only specify their VMs in terms of virtual hardware (amount of memory, size of the virtual disk, number of network cards, and so on), and the virtualization solution they want to use (for instance Xen). This specification (called a *profile*) is then parsed by V2M, which in-turn automatically sets the system up, initializes the VM and provides a simple interface to interact with the VM, *e.g.*, to boot the VM

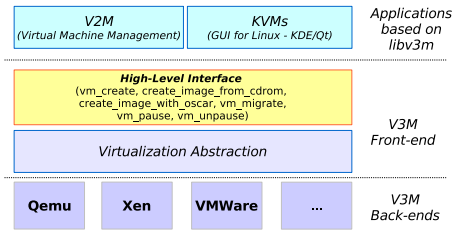

Fig. 3. V2M Architecture

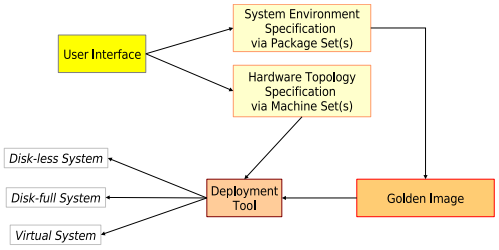

Fig. 4. Deployment of Virtual Environments

and to install the VM (see Fig. 3). Because V2M abstracts the virtualization solution, it is possible to switch from one to another simply by modifying the virtualization solution identifier in the VM profile.

Based on the abstraction provided by V2M, the OSCAR management tool has been extended to support the definition of golden images for both VMs and host OSs (see Fig. 4). A golden image is completely independent to the actual hardware (physical or virtual) on which the system is deployed. Since OSCAR-V integrates virtualization solutions, such as Xen, OSCAR can transparently deploy and configure VMMs as well as associated host OSs.

5 Conclusion

System-level virtualization provides several advantages for HPC that may change the way modern HPC systems are currently used though plug-and-play computing, system environment customization, computing on demand, and transparent application resilience through system provided fault tolerance.

However, the usage of virtual machines also creates several challenges including: (i) the development of a virtualization solution suitable for HPC, (ii) the development of tools and methods for the management of virtual systems and, (iii) the use of advanced capabilities enabled by virtualization, such as VM pause/unpause, VM checkpoint/restart, and VM migration.

For that, we focused on two different projects. First is the development of a new VMM for HPC that has a small system footprint and is modular and easily extensible. Second, we developed a set of fault tolerance and system management tools able to take advantage of the capabilities provided by system-level virtualization.

Our fault tolerance effort led to the implementation of a framework for the deployment of new reactive, proactive, or hybrid fault tolerance policies, based on capabilities such as VM pause/unpause, checkpoint/restart, and migration.

Our system management effort led to the development of an integrated solution for the management of virtual systems. A core component of OSCAR-V is the V2M tool which abstract the parameters specifying the different virtualization solutions in order to provide users with a simple interface for the definition and management of VMs. Based on this abstraction, it is possible to specify and deploy systems independent to them being physical or virtual.

References

[1] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, A. Warfield, Xen and the art of virtualization, in: Proceedings of the 19^{th} ACM Symposium on Operating Systems Principles (SOSP), ACM Press, Bolton Landing, NY, USA, 2003, pp. 164–177.

[2] C. Clark, K. Fraser, S. Hand, J. G. Hansen, E. Jul, C. Limpach, I. Pratt, A. Warfield, Live migration of virtual machines, in: Proceedings of the 2^{nd} ACM/USENIX Symposium on Networked Systems Design and Implementation (NSDI), USENIX Association, Boston, MA, USA, 2005, pp. 273–286.

[3] A. Whitaker, M. Shaw, S. D. Gribble, Denali: Lightweight virtual machines for distributed and networked applications, Technical Report 02-02-01, University of Washington (Feb. 2001).

[4] R. P. Goldberg, Architecture of virtual machines, in: Proceedings of the Workshop on Virtual Computer Systems, ACM Press, Cambridge, MA, USA, 1973, pp. 74–112.

[5] J. Liu, W. Huang, B. Abali, D. K. Panda, High performance VMM-bypass I/O in virtual machines, in: Proceedings of the USENIX Annual Technical Conference (USENIX), USENIX Association, Boston, MA, USA, 2006, p. 3.

[6] W. Huang, Q. Gao, J. Liu, D. K. Panda, High performance virtual machine migration with RDMA over modern interconnects, in: Proceedings of the 9^{th} IEEE International Conference on Cluster Computing (Cluster) 2007, Austin, Texas, USA, 2007.

[7] G. Vallée, T. Naughton, H. Ong, S. L. Scott, Checkpoint/restart of virtual machines based on Xen, in: Proceedings of the High Availability and Performance Workshop (HAPCW), in conjunction with the Los Alamos Computer Science Institute (LACSI) Symposium, Santa Fe, NM, USA, 2006.

[8] G. Vallée, T. Naughton, S. L. Scott, System management software for virtual environments, in: Proceedings of the 4^{th} International Conference on Computing Frontiers (CF), ACM Press, Ischia, Italy, 2007, pp. 153–160.