Complementarity between Virtualization and Single System Image Technologies*

Jérôme Gallard[1], Geoffroy Vallée[2], Adrien Lèbre[1], Christine Morin[1], Pascal Gallard[3], and Stephen L. Scott[2]

[1] INRIA Rennes - Bretagne Atlantique, PARIS project-team, Rennes, France
[2] Oak Ridge National Laboratory, Oak Ridge, USA
[3] KERLABS, Rennes, France
{jerome.gallard,adrien.lebre,christine.morin}@inria.fr
{valleegr,scottsl}@ornl.gov
pascal.gallard@kerlabs.com

Abstract. Nowadays, the use of clusters in research centers or industries is undeniable. Since few years, the usage of virtual machines (VM) offers more advanced resource management capabilities, using features such as virtual machine live migration. Because of the latest contributions in the domain, some may argue that single system image (SSI) technologies are now deprecated, without considering some complementarities between VMs and SSI technologies are possible.

After evaluating different configurations, we show that combining both approaches allows us to better address cluster challenges such as flexibility for the usage of available resources and simplicity of use. In other terms, the study shows that VMs add a level of management flexibility between the hardware and the application, whereas, SSIs give an abstraction of the distributed resources. The simultaneous usage of both technologies could improve the overall platform resources utilization, the cluster productivity and the efficiency of the running applications.

Keywords: cluster, virtualization, SSI, resource management.

1 Introduction

Clusters are today a standard computation platform for both research and production. Batch schedulers or single system image systems (SSI) are frequently used to manage clusters. In the first case, a head node is in charge of scheduling applications whereas in the second case, the SSI makes an abstraction of the cluster resources creating the illusion of an SMP machine. Several studies have focused on combining virtual machines (VMs) and batch schedulers in order

* The INRIA team carries out this research work in the framework of the XtreemOS project partially funded by the European Commission under contract #FP6-033576. ORNL's research sponsored by the Laboratory Directed Research and Development Program of Oak Ridge National Laboratory (ORNL), managed by UT-Battelle, LLC for the U. S. Department of Energy under Contract No. DE-AC05-00OR22725.

E. César et al. (Eds.): Euro-Par 2008 Workshops, LNCS 5415, pp. 43–52, 2009.

to provide better resource control [1]. Features provided by virtualization technologies (such as isolation and suspend/resume) enable more advanced resources management capabilities. For instance, one VM can be suspended or migrated to another node. Thus, administrators are able to make maintenance operations without impacting the platform availability. On the other side, isolation mechanisms simply the management of security constraints.

This trend around virtualization seems to impact directly cluster management and more precisely SSI technology which enables, in some ways, similar capabilities. For instance, the openMosix SSI project [2] has recently closed. In that sense, we wonder whether virtualization technology will surpass the SSI systems or if these two models are complementary.

This paper addresses these questions and investigates in which extends the association of both virtualization and SSI technologies could improve the usage and management of distributed architectures as well as application execution (*e.g.,* administration, application debugging, and security).

To our best knowledge, virtualization and SSI approaches have been used only in the Peta-SSI project [3], using VMs in order to study the system scalability, "emulating" a large number of nodes. In other terms, only one capability (virtual machine stacking) provided by virtualization solutions has been studied. In this document, we analyze the potential benefits of all major capabilities provided by the usage of VMs in an SSI context. This study has been done in a theoretical way (no results of experiences are presented in this document).

The remainder of this paper is organized as follows: Section 2 clarifies the notion of virtualization and virtualization. Section 3 gives a brief background on SSI systems. Section 4 investigates the complementarity of virtualization and SSI. Section 5 reports lessons learnt. Section 6 concludes.

2 Introduction to Virtualization

Virtualization is an active research topic in operating systems (OS) since the 70's but regained popularity with the latest technologies which provide extra computational capabilities (new machines such as multi-core processors can compete with multiple individual servers that are few years old). A way to use this extra capabilities is to execute VMs on top of physical machines. From our point of view, the concept of VM includes five major features: (i) *isolation* (*i.e.,* degree of isolation between VMs, the bare hardware and applications running in different VMs), (ii) *server consolidation* (*i.e.,* capability of changing on demand resources allocated to a specific VM), (iii) *application portability* (*i.e.,* capability of executing an unmodified application), (iv) *virtual machine portability* (*i.e.,* capability of migrating virtual environments to different hardware architectures), (v) *suspend/restart* (*i.e.,* possibility to take a snapshot/resume of VMs).

Nowadays, two typical virtualization approaches are possible: one which implements the virtualization at the system-level (well-known Goldberg classification [4]) and the other at the process-level (*containers*). Part (a) of Table 1 summarizes functionalities supported by virtualization solutions.

Table 1. Selected Capabilities Enabled by Virtualization (a) or SSI (b)

	Virtualization (a)			SSI (b)	
	Container	Type-I Virt.	Type-II Virt.	Partial-SSI	SSI (full-SSI)
Isolation	-	+	+	-	-
Server conso.	+	+	+	-	+
App. Portability	-	+	+	-	-
VM Portability	-	-	+	-	-
Suspend/Restart	+	+	+	+	+

System-level Virtualization (Type-I, Type-II). This approach aims at virtualizing a full OS. For that, a virtual hardware is exposed to a full OS within a VM. The system running in a VM is named a *guest OS*. According to isolation properties associated with virtualization, the VM cannot execute privileged instructions at the processor level. To access the physical devices, drivers are hosted in a privileged OS, called *host OS*. Moreover, VMs run concurrently and their execution is scheduled by the *hypervisor*. The hypervisor is also in charge of forwarding all privileged instructions from VMs to the host OS.

Goldberg created a model for system-level virtualization, model based on two functions, ϕ and f. The function ϕ makes the correspondence between process running on the guest OS and the resources (exposed within the VM) whereas f makes the correspondence between resources allocated to a VM and the bare hardware. Based on those functions, Goldberg identified two different types of system-level virtualization: *type-I* (*e.g.,* Xen [5]) and *type-II* (*e.g.,* QEMU [6] and VMware [7]).

Process-level Virtualization (Container). It consists of running several processes concurrently on top of the same OS, each having its own view of available resources (*e.g.,* OpenVZ [8], *chroot* [9] and *containers* capabilities provided by recent kernels). The Goldberg classification is only focusing on the former level virtualization solutions and does not integrate such process-level virtualization solutions. In this paper, we only consider OpenVZ-like solutions. Recent kernel containers approach are not taken into account.

3 Introduction to Single System Image

An SSI is an OS that aims to abstract the distributed nature of the cluster in order to ease users, administrators and programmers tasks. For that, a SSI globally manages distributed resources. Because a transparent management of resources is difficult to implement at user-space (it is typically the responsibility of the OS), most of the SSIs are implemented at OS-level. Two kinds of SSI exist: (i) partial-SSI and (ii) SSI (or full-SSI).

Partial Single System Image. A *partial-SSI* only allows a global management of a subset of cluster resources (typically processes) and only from a central

location (*i.e.,* a "head node"). All processes are manipulable from this head node like if they were local processes; on other nodes, resources are still viewed as distributed. The abstraction of the distribution of resource is therefore "partial". Glunix [10], Bproc [11] or Cplan [12] are examples of partial-SSIs.

Full Single System Image. A *SSI* (or full-SSI) provides not only a global management of processes but also a global management of all other resources, and therefore gives to users the illusion to use an SMP machine. In such systems, there is no head node; each node is equal and has a global view of the distributed resources. Users are able to run SMP applications on the cluster without application modification or recompilation. For instance, SSIs implement a Distributed Shared Memory (DSM). This functionality enables the execution of OpenMP parallel applications based on the shared memory programming paradigm. Therefore, a SSI abstracts the complexity created by the resource distribution. Kerrighed [13] and OpenMosix [14] are examples of such SSIs.

The SSI technology has several interesting capabilities for cluster management, high performance, high availability, and, ease of use and programming. However, in this document, we focus only on the functionalities described in Section 2. Part (b) of Table 1 summarizes them.

4 Combining Virtualization and Single System Image

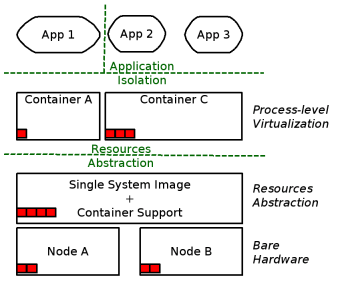
In this section, we present a systematic analysis of the combination of SSI and virtualization technologies. To realize this study, we selected three different target applications: (i) a web server such as Apache [15], (ii) an MPI-like application (based on message passing), and (iii) an OpenMP-like application (based on a shared memory). We think that these kinds of application are representative of a large part of business and scientific software. To achieve our main objective, we analyze the benefits of the five capabilities enabled by virtualization (cf. Section 2) with configurations exploiting SSIs.

4.1 Single System Image and Containers

Containers (*e.g.,* OpenVZ-like solutions) allow applications to be isolated from each other on the same node. Moreover, it is generally possible to assign an IP address, to allocate memory and CPU time to each container. Hence, a container could be migrated in most cases from one node to another.

Container Upon Single System Image. Figure 1 depicts the architecture of a typical system running containers upon an SSI. With this architecture, the SSI abstracts the distributed resources. Based on this "simplified" and "unified" view of the distributed system, global resources can be dynamically and transparently assigned to containers in order to fit at best applications needs. In other terms, containers could dispose of more resources that is available on one node.

Isolation: Applications are isolated from the bare hardware by containers that are running on top of the SSI. However, an application could hijack a container,



Little squares represent the amount of resources provided by the system. For instance, each node provides 2 resources and the SSI provides 4 resources (aggregation of resources provided by node A and node B).

Fig. 1. Containers Upon Single System Image

and in consequence, compromise the security of the whole system (there is one kernel for all containers). This property is not validated.

Server Consolidation: The SSI globally manages all resources, it is possible to change on demand the resources allocated to each container. These capabilities are very interesting for an Apache server administrator: according to the frequentation of a web site, it is possible to allocate more or less physical resources to the cluster (resizing the containers accordingly). This property is validated.

Application Portability (AP1): Thanks to the SSI, containers can span multiple nodes. Thus, an OpenMP application or an Apache server could take advantage of the SSI DSM (spanning nodes) whereas, an MPI application could take advantage of several containers each of them having their own IP address. Application portability is therefore guaranteed by such an architecture.

Virtual Machine Portability: Containers have not been designed to create a virtual hardware different from the hardware it is running on. Thus, the virtual machine portability is not validated.

Suspend/Restart: Containers can be suspended/restarted at any time by any other entity running with the correct privileges inside the system. Moreover, the SSI can suspend/restart any containers since a container is a set of standard resource from the SSI point of view. This property is validated.

Single System Image Upon Containers. Figure 2 presents the use of an SSI upon containers. The architecture is not realistic because no individual kernel can run in a container, only user-level applications can be hosted.

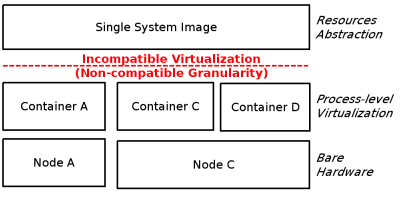


Fig. 2. Single System Image Upon Containers

4.2 Single System Image and Type-I Virtualization

Type-I virtualization solutions have an hypervisor running directly on top of the bare hardware and "hosting" the host OS and the VMs.

Type-I Virtualization Upon Single System Image. Figure 3 shows the architecture of a type-I virtualization solution running upon a SSI. This approach enables the implementation of a "global type-I hypervisor", including SSI features into the hypervisor. Such a global hypervisor can transparently and globally manage resources (creation of an SMP illusion) and typically the resource allocated to VMs is not restricted to the local resources.

Isolation (I2): The type-I hypervisor isolates applications from both the bare hardware and others VMs. For instance, if a hacker is able to become root on one VM, only the local VM is compromised: isolation is validated.

Server Consolidation (SC2): In case of a node addition, VMs can be moved to the new node; in case of node eviction, VMs can be transparently moved away. This propriety is validated.

Application Portability: Same as AP1, substituting containers by VMs.

Virtual Machine Portability: Currently no type-I virtualization solution provides emulation capabilities. Moreover, the SSI running on the side of the type-I hypervisor does not support by definition hardware architecture heterogeneity. It is therefore not possible to migrate VMs between nodes having different hardware architectures. VM portability cannot be achieved.

Suspend/Restart (SR2): Type-I hypervisor enables VM suspend/restart. However, if two applications are running in the same VM, it is not possible to suspend only one of them. Property of suspend/restart is not totally validated.

Single System Image Upon Type-I Virtualization. Figure 4 shows the architecture of an SSI upon the VMs of a type-I virtualization solution. In this

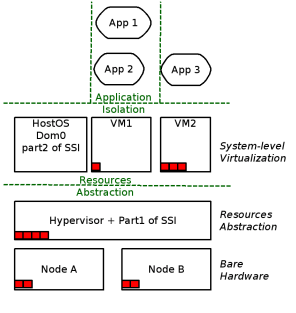


Fig. 3. Type-I Virtualization Upon SSI

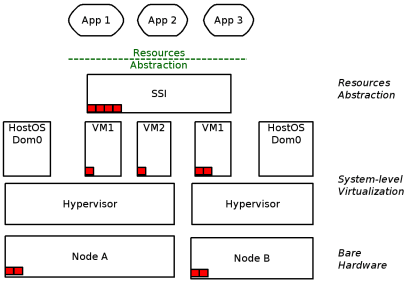


Fig. 4. SSI Upon Type-I Virtualization

case, an hypervisor is deployed on all cluster nodes and the SSI is executed in different VMs; each VM being potentially hosted by different hypervisors.

Isolation (I3): The type-I virtualization isolates both the SSI and applications from the bare hardware. However, if the SSI is compromized the management of all resources and thus all running applications may be compromized. Isolation is therefore partially achieved.

Server Consolidation: The type-I hypervisor enables VMs migration and the SSI provides process migration between VMs. This property is validated.

Application Portability: Applications are actually running on top of the SSI, providing an SMP illusion. This enables the execution of MPI-like, OpenMP-like and Apache-like applications compiled for the native OS of the SSI. This property is validated.

Virtual Machine Portability: Today no type-I virtualization solution allows the emulation of an architecture at the VM level that is different from the bare hardware. Portability is, for the moment, not achieved.

Suspend/Restart (SR3): Both virtualization and SSI solutions provide suspend/restart mechanisms, respectively suspending/restarting VMs and processes. This property is validated.

4.3 Single System Image and Type-II Virtualization

Type-II virtualization solutions run VMs upon a host OS and generally provide live migration and suspend/resume capabilities.

Type-II Virtualization Upon Single System Image. Figure 5 shows the execution of VMs upon an SSI. The SSI globally manages all the distributed resources; the type-II virtualization hypervisor can therefore allocate distributed resources to VMs on demand in a transparent manner.

Isolation: Same as I2, substituting type-I hypervisor by type-II hypervisor.

Server Consolidation: Same as SC2, substituting type-I by type-II.

Application Portability: Same as AP1, substituting containers by VMs.

Virtual Machine Portability: It is possible to migrate a VM between nodes (according to VM resource needs) or SSIs compiled for other architectures. This property is validated.

Suspend/Restart: Same as SR2, substituting type-I by type-II.

Single System Image Upon Type-II Virtualization. Figure 6 shows the architecture of an SSI upon VMs. As for the type-I, each node runs VMs, and the SSI is deployed upon them.

Isolation: Same as I3, substituting type-I hypervisor by type-II hypervisor.

Server Consolidation: In case of node addition/removal, there are two cases: (i) an *automatic reconfiguration of the SSI* (*e.g.,* the SSI "knows" that nodes are

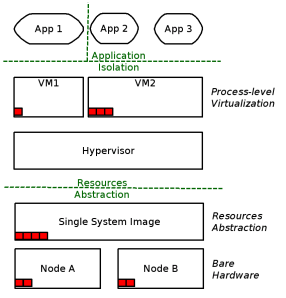


Fig. 5. Type-II Virtualization Upon SSI

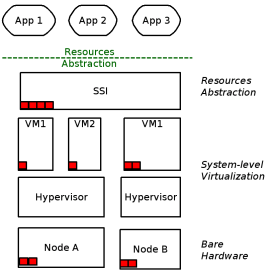


Fig. 6. SSI Upon Type-II Virtualization

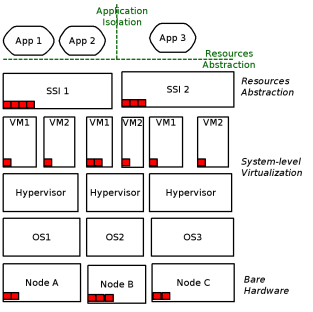


Fig. 7. Isolation of Two Distinct SSIs

added or removed), and (ii) a *VM live migration* to another node (*e.g.*, the SSI is deployed on top of several VMs, and this number is static). In each case, the property of server consolidation is validated.

Application Portability: Same as AP1, substituting containers by VMs.

Virtual Machine Portability: The type-II virtualization enables the emulation of different hardware architectures. It is therefore possible to migrate VMs to different hardware architectures, only the architecture of the virtual hardware exposed inside the VMs has to be consistent (the SSI does not support heterogeneity). This property is validated.

Suspend/Restart: Same as SR3, substituting type-I by type-II.

5 Lessons

Containers on Top of Single System Image Clusters. Using the container based solution in an SSI, resources exposed to applications can span multiple cluster nodes. By providing the illusion that a cluster is a virtual SMP, the SSI retains all the advantages enabled by containers on a real SMP machine in a cluster environment; removing frontiers between cluster nodes.

Virtual Machines on Top of Single System Image Clusters. This configuration has a major advantage: application portability. For instance, with VMs, it is possible to execute an application developed for processor technology "A" and OS "B" on top of a computer running an SSI OS based on OS "C" and developed for processor technology "D". This means that any application binary can be executed on top of an SSI OS, provided that the appropriate virtualization technology is available. For example, an IIS web server compiled for Windows OS could be deployed on the top of a VM running on an SSI compiled for Linux.

Single System Image on Top of Virtual Machines. Executing an SSI OS on top of a virtual cluster provides a flexible, simple and on demand resource

allocation to applications, but also system-level adaptation in case of cluster configuration changes (node addition and eviction). The idea is to simplify management tasks and to reduce cost of power consumption. If an application requires more (respectively less) resources and more (respectively less) physical cluster nodes, the virtual machines are simply migrated to remote physical nodes. For instance, a multi-threaded Apache server could be deployed on more or less physical nodes according to the amount of requests. Moreover, it becomes possible to execute multiple virtual clusters on the same real cluster, each of them been isolated from the others (for instance, two OpenMP applications can be executed in an isolated way on two different virtual clusters, see Figure 7).

6 Conclusion and Future Works

Nowadays, virtualization technologies are very popular for the execution of applications and services on top of computers. The motivation of this paper was to answer the following question. Do these trends make the SSI for clusters irrelevant for the future?

Based on the current state of the art on SSI and on virtualization techniques, we analysed different configurations combining SSI and virtualization techniques in clusters. From the analysis presented in this paper, we conclude that virtualization and SSI complement each other. A full SSI makes transparent resource distribution in cluster nodes, providing the illusion of a virtual SMP machine (abstraction of distributed resources, see Table 2, cases 4 and 6), whereas the virtualization technologies provide flexibility in resource management (cases 1, 3, and 5).

Table 2. Summary of the different cases studied in this document: (1) Container upon SSI; (2) SSI upon container; (3) type-I upon SSI; (4) SSI upon type-I; (5) type-II upon SSI; (6) SSI upon type-II

	1	2	3	4	5	6
Isolation	-	N/A	+	+	+	+
Server conso.	+	N/A	+	+	+	+
Application Portability	+	N/A	+	+	+	+
VM Portability	-	N/A	-	-	+	+
Suspend/restart	+	N/A	-	+	-	+

We have started experimentations on Kerrighed [13] on top of VMware Server 1.0.4 [7] (no porting effort is required in the current state of the technology). These experiments are realized on a cluster of Grid5000 [16]. We test several kinds of applications: bags of tasks, parallel applications (MPI, OpenMP), servers (Apache with different configurations based on multiple processes or threads). This would allow us to compare the behavior of these applications on clusters running an SSI, running VMs or running one of the combinations of SSI and virtualization solutions that have been identified as attractive. In particular, we plan to measure the applications performance in such environments.

From a more theoretical point of view, we work on designing a model extending the one proposed by Goldberg to present in a uniform framework the hardware, the emulated hardware, the OS, the different virtualization techniques, containers and SSIs.

Hence, we plan to investigate the use of virtualization techniques in a Grid environment for commercial applications requiring strong isolation.

References

1. Grit, L., Irwin, D., Marupadi, V., Shivam, P., Yumerefendi, A., Chase, J., Albrecht, J.: Harnessing virtual machine resource control for job management. In: Proceedings of the First International Workshop on Virtualization Technology in Distributed Computing (VTDC) (November 2006)
2. OpenMosix, `http://openmosix.sourceforge.net/`
3. Studham, R.S., Cox, A., Walker, B.: Petascale single system image and other stuff (2007)
4. Goldberg, R.P.: Architecture of virtual machines. In: Proceedings of the Workshop on Virtual Computer Systems
5. Barham, P., Dragovic, B., Fraser, K., Hand, S., Harris, T., Ho, A., Neugebauer, R., Pratt, I., Warfield, A.: Xen and the art of virtualization. In: SOSP 2003: Proceedings of the nineteenth ACM symposium on Operating systems principles, pp. 164–177. ACM, New York (2003)
6. Bellard, F.: Qemu, a fast and portable dynamic translator. Technical report, USENIX Association (2005)
7. VMware: `http://www.vmware.com`
8. OpenVZ: `http://wiki.openvz.org/Main_Page`
9. Chroot: ion, `http://www.gnu.org/software/coreutils/manual/coreutils.html`
10. Ghormley, D.P., Petrou, D., Rodrigues, S.H., Vahdat, A.M., Anderson, T.E.: GLUnix: A Global Layer Unix for a network of workstations. Software Practice and Experience 28(9), 929–961 (1998)
11. Hendriks, E.: BProc: the Beowulf Distributed Process Space. In: ICS 2002: Proceedings of the 16th international conference on Supercomputing, pp. 129–136. ACM Press, New York (2002)
12. Riesen, R., Brightwell, R., Fisk, L.A., Hudson, T., Otto, J., Maccabe, A.B.: Cplant. In: Proceedings of the Second Extreme Linux workshop at the 1999 USENIX Annual Technical Conference (1999)
13. Morin, C., Lottiaux, R., Vallée, G., Gallard, P., Margery, D., Berthou, J.Y., Scherson, I.: Kerrighed and data parallelism: Cluster computing on single system image operating systems. In: Proc. of Cluster 2004. IEEE, Los Alamitos (2004)
14. Barak, A., La'adan, O.: The MOSIX multicomputer operating system for high performance cluster computing. Future Gener. Comput. Syst. 13(4-5), 361–372 (1998)
15. Foundation, A.S.: `http://httpd.apache.org`
16. Grid5000, `http://www.grid5000.fr`