Is Virtualization Killing Single System Image Research?

Jérôme Gallard, Adrien Lebre, Geoffroy Vallée, Pascal Gallard, Stephen L. Scott, Christine Morin

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Is Virtualization Killing Single System Image Research?

Jérôme Gallard — Adrien Lebre — Geoffroy Vallée — Pascal Gallard — Stephen L. Scott — Christine Morin

N° ????

Novembre 2007

Thème NUM

ISSN 0249-6399 ISRN INRIA/RR--????--FR+ENG

Is Virtualization Killing Single System Image Research?

Jérôme Gallard* , Adrien Lebre* , Geoffroy Vallée† , Pascal Gallard‡ , Stephen L. Scott† , Christine Morin*

Thème NUM — Systèmes numériques
Projet PARIS

Rapport de recherche n° ???? — Novembre 2007 — 23 pages

Abstract: Nowadays, use of clusters in research centers or industries is undeniable. Cluster usage is typically based on two different models: (i) batch schedulers and (ii) single system image (SSI). In the first case, applications are scheduled by a "supervisor", the batch scheduler, according to cluster resources availability. In the second case, an SSI operating system (OS) gives the illusion that a distributed system is a standard SMP machine, allowing users to use standard UNIX tools to manage their applications.

Even if SSI solutions are usually more complete in terms of functionality, batch schedulers are usually preferred because of their simplicity in term of both configuration and usage. Moreover, since few years, combining virtual machines and batch systems offer 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 SSI technologies are now deprecated.

In this paper, we analyze whether virtualization technologies will surpass the SSI approach, or if these two models are not contradictory but complementary. In fact, after evaluating different configurations, we show that by combining both approaches, we can improve several aspects associated to application computation such as flexibility of administration, simplicity of use, security and portability.

Key-words: Single System Image, Virtual Machine, Container, Resource Management, Cluster.

* INRIA Rennes Bretagne Atlantique, Rennes France - firstname.lastname@irisa.fr
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.
† Oak Ridge National Laboratory, Oak Ridge, USA - {valleegr,scottsl}@ornl.gov
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.
‡ KERLABS, Gevezé France - pascal.gallard@kerlabs.com

Unité de recherche INRIA Rennes
IRISA, Campus universitaire de Beaulieu, 35042 Rennes Cedex (France)
Téléphone : +33 2 99 84 71 00 — Télécopie : +33 2 99 84 71 71

La virtualisation condamne-t-elle les recherches sur les systèmes à image unique ?

Résumé : De nos jours, les grappes de calculateurs sont très utilisées aussi bien dans les laboratoires de recherche que dans l'industrie. La gestion des grappes se fait traditionnellement par les systèmes à exécution par lots (*batch scheduler*) ou par les systèmes à image unique (*single system image - SSI*). Dans le cas des systèmes à exécution par lots, les tâches sont soumises à un ordonnanceur capable de les exécuter au cours du temps selon la disponibilité des ressources de la grappe. Dans le second cas, le système à image unique fournit l'illusion d'utiliser la grappe comme une machine SMP standard avec les outils UNIX traditionnels de gestion des tâches.

Même si les SSIs sont plus complets en terme de fonctionnalités, les systèmes à exécution par lots sont généralement plus utilisés de part leur simplicité d'administration et d'utilisation. De plus, depuis quelques années, la combinaison des machines virtuelles et des systèmes à exécution par lots offre une plus grande flexibilité dans la gestion des ressources d'une grappe grâce à des fonctionnalités comme par exemple la migration de machine virtuelle (*live migration*). De part ces avancées dans ce domaine, certains pensent que les approches de type SSI ne sont plus appropriées.

Dans ce document, nous tentons de répondre à la question : la virtualisation va-t-elle surpasser les SSI ou au contraire, existe-t-il des complémentarités entre ces deux systèmes ? Après évaluation de différentes combinaisons SSIs/VMs, nous constatons que la combinaison des deux approches permet d'améliorer plusieurs aspects ayant trait à la gestion des grappes tels que la flexibilité d'administration, la simplicité d'utilisation, la sécurité et la portabilité.

Mots-clés : Système à image unique, Machine virtuelle, Conteneur, Gestion de ressources, grappe de calculateurs.

1 Introduction

Clusters are today a standard computation platform for both research and production. Batch schedulers or single system image systems are frequently used to manage clusters. In the first case, a head node is in charge of scheduling applications whereas in the second case, a Single System Image (SSI) makes an abstraction of the cluster giving the illusion of an SMP machine.

Since 5 years, several studies have focused on combining Virtual Machines (VMs) and batch schedulers in order to exploit cluster resources. Features provided by virtualization solutions (such as isolation, resource management, portability, suspend/restart) enable more advanced resources management capabilities. For instance, a virtual OS isolated in one VM can be migrated whatever the underlying architecture is. Thus, administrators are able to make any maintenance operations without impacting availability aspects. On the other side, use of isolation mechanisms makes management of security constraints easier for developers.

This trend around virtualization seems to impact directly cluster management and more precisely SSI technology which enables in some ways, similar capabilities. Several well-known SSI solutions such as openMosix [13], have planned to end their development efforts and simply end the project. In that sense, we wonder whether the virtual machine technology will surpass the SSI systems or if these two models are complementary.

This paper addresses these questions and investigates in which ways, the association of both virtualization and SSI could improve the usage and management of distributed platforms as well as for the execution of applications on such platforms (cluster administration, application debugging, security and so on).

To our best knowledge, virtualization and SSI approaches have been used in a common way only in the Peta-SSI project [17]. In this specific case, the key idea consists in 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.

The remainder of this paper is organized as follows: Section 2 clarifies the notion of virtualization and virtual machines; Section 3 gives a brief background on SSI systems; Section 4 investigates the complementarity of virtualization and SSI; Section 5 reports lessons learnt and preliminary experimental results confirming some of these lessons. Section 6 concludes and gives some perspectives.

2 Introduction to Virtualization

Virtualization is an active research subject in operating systems since the 70's but regained popularity with the latest technologies (such as multi-core processors) which provide extra computational capabilities (new machines can com-

pete with multiple individual servers that are few years old). A way to use this extra capabilities is to execute virtual machines on top of physical machines. The concept of virtual machine is interesting because it enables the following properties:

- *Isolation*: isolation degree of virtual machines from the bare hardware and other applications running in different VMs. A fully isolated VM can be compromized without compromizing any other VMs, the hypervisor or the host OS.
- *Server consolidation*: capability of changing on demand resources allocated to a specific VM.
- *Virtual Machine Portability*: capability of migrating virtual environments to different hardware architectures. It means that the virtualization solution has to *emulate* other hardware architectures on which VMs can be based.
- *Application Portability*: capability of executing application inside a VM without modifications. In other terms, the capability to adapt the virtual machine to application needs instead of adapting the application to the hardware characteristics.
- *Suspend/Restart*: possibility to take a snapshot of and resume VMs.

However, these latest contributions led to different types of virtualization, some focusing on the virtualization of a full system (*i.e.*, creating a virtual hardware on which an OS can be executed), others focusing on the virtualization of processes inside a given operating system (abstracting this process from the local resources).

Section 2.1 gives a classification of these different solutions, highlighting their similarities and differences. Section 2.2 summarizes capabilities enabled by the different virtualization solutions that are considered in this document.

2.1 Virtualization Classification

Two typical virtualization approaches are possible: one which implements the virtualization at the hardware level and the other at the operating system level.

For simplification, we limit our study to three "virtualization solutions": the *container* approach and the two types of hardware virtualization based on the well-known Goldberg classification [7]. These solutions are widely used and representative of the commercial and research based solutions.

2.1.1 Hardware Level Virtualization: Goldberg Classification

The system running in a virtual machine is named a *guest OS* since it is a full operating system running on a virtual hardware. Because of this isolation from the bare hardware, the VM cannot execute any privileged instruction at

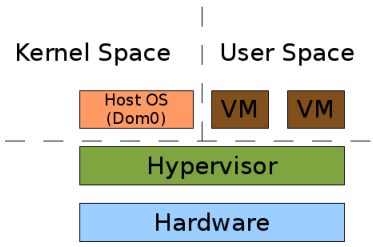

Figure 1: Type-I Virtualization Architecture
The host OS can execute privileged instructions

the processor level or even access the physical devices. To access the physical devices, drivers are hosted in a privileged operating system, called *host OS*. Moreover, virtual machines run concurrently and their execution is scheduled by the *hypervisor*. The hypervisor is also in charge of hijacking the execution of protected processor instructions (such as access to the memory page table) and resolving or forwarding to the host OS these instructions (typically the hypervisor manages only the memory for the VMs, other privileged instructions have to be forwarded to the host OS).

Goldberg identified two different types of hardware virtualization: *type-I* and *type-II*. The classification is based on a model composed of two functions: ϕ and f. ϕ makes the correspondance between process running on the guest OS and the VM resources. f makes the correspondance between the VM resources and the bare hardware.

Type-I Virtualization According to the Goldberg model, type-I virtualization is based on the equation $f : R_{n+1} \rightarrow R_n$. In other terms, the hypervisor makes the translation between the machine resource of a machine $n+1$ to a machine n, *i.e.*, the hypervisor is running on the bare hardware and both the host OS (used mostly as a driver domain) and virtual machines are running on top of the hypervisor. Figures 1 shows the architecture of a type-I virtualization solution.

It provides a good isolation for applications: each VM runs its own OS and a kernel attack on the guest OS does not impact the host OS nor the other VMs. A type-I architecture provides server consolidation functionality. In fact, thanks to VM migration mechanisms, it is possible to change the cluster size on demand: when a node is added, VM previously running on other nodes can be migrated to these new nodes to balance the load. Before a node is stopped its VM can be migrated to another node The portability is limited by hardware constraints and migration can occur only if the two host OS are based on same hardware resources. Finally, each guest OS can be suspended and restarted later.

Xen [3] is an example of type-I virtualization.

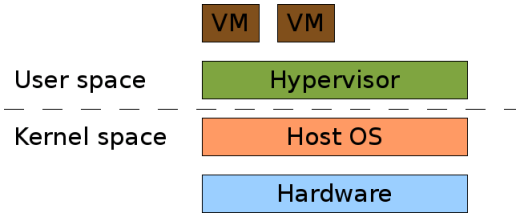

Figure 2: Type-II Virtualization Architecture

Type-II Virtualization Type-II virtualization solutions are based on the following model: $f : R_{n+1} \rightarrow P_n$, with R_{n+1} the resource of the machine $n+1$ and P_n the process running on the machine n. In other terms, the hypervisor makes the translation between the resource of a machine $n+1$ and a process running on the machine n, *i.e.*, the hypervisor runs on top of an existing operating system (which is used for resource management), and VMs runs on top of the hypervisor. Figure 2 shows the architecture of a type-II virtualization solution.

The hypervisor provides good isolation for applications: each VM has its own OS and a kernel attack on the guest OS cannot impact the host OS nor other VMs. Type-II virtualization also enables server consolidation: thanks to the migration capability, it is possible to manage the cluster size on demand. Concerning the portability, if the type-II hypervisor integrates emulation capabilities, it is possible to migrate a VM from an architecture to another.

QEMU [4] and VMware [18] are examples of type-II virtualization.

2.1.2 Operating System Level Virtualization: Containers

When Goldberg did his classification, only hardware level virtualization solutions were available and therefore this classification does not integrate virtualization solutions inside an OS.

Containers mechanisms, provided by recent kernels, enables virtualization "inside" a given OS, *i.e.*, several processes are running concurrently on top of the same operating system, each having its own view of available resources. Note that with containers the hypervisor and the host OS are "merged"; we refer to this merge simply as host OS. Figure 3 shows the architecture of such a system.

The host OS provides container support which is a certain form of isolation. All privileges actions are executed by the host OS, *i.e.*, the host OS hijacks all privileged actions from VMs. The main issue in this approach is that a kernel attack on a container is the same thing that a kernel attack on the host OS. This architecture also provides a "server consolidation" capability. In fact, thanks to the container migration mechanism, it is possible to manage the size of all allocated resources on demand. However, container migration implies strong

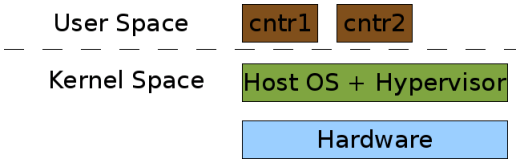

Figure 3: Container Based Virtualization

architecture constraints which limits the portability aspects. Finally, containers host OS enables the suspend/restart capability.

OpenVZ [14] and *chroot* [6] are examples of containers.

2.2 Summary

Table 1 summarizes the functionalities provided by each kind of virtualization solutions.

	Container	Type-I Virt.	Type-II Virt.
Isolation	-	+	+
Server consolidation	+	+	+
Application Portability	-	+	+
VM Portability	-	-	+
Suspend/Restart	+	+	+

Table 1: Selected Capabilities Enabled by Virtualization

3 Introduction to Single System Image

An SSI is an operating system that aims to abstract the distributed nature of the cluster in order to ease users, administrators and programers tasks. There are two kinds of SSI: (i) partial-SSI and (ii) SSI (or full-SSI).

Partial-SSI allows a global management of processes in the cluster but provide this global view only from "head nodes". For instance, the *ps* command on an partial-SSI displays all processes running within the cluster. All processes are manipulable from the head node like if they were local processes. Glunix [5], Bproc [10] or Cplan [15] are examples of partial-SSI with one head node.

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 the user the illusion to use an SMP machine. For instance, SSIs implement Distributed Shared Memory (DSM). Thus, users are able to run SMP applications on the cluster without application modification or recompilation (*e.g.*, on such a system, the *cat /proc/meminfo* command displays the sum of the memory of all cluster nodes). These functionnalities enables the execution of OpenMP parallel applications based on the shared memory programming model. Moreover, the

SSI enables a dynamicaly customization of the process scheduling policy, based on application needs and cluster nodes load. Finally, all nodes are equal, *i.e.*, there is no head node. Kerrighed [12], OpenMosix [2] are examples of such SSIs (to our knowledge, Kerrighed is the most advanced SSI).

The SSI technology has several interesting functionalities for cluster management, high performance, high availability, and ease of use & programming. However, a SSI provides the same level of isolation for applications as a standard OS, *e.g.*, a process executing on top of a SSI sees all other processes running concurrently.

SSI also provides server consolidation capabilities: with the process migration capacity and with the "on demand" node removal/addition capacity, it is possible to expand or shrink the cluster size on demand. *SSI-node-add* and *SSI-node-remove* are used by the cluster administrator to change dynamically the cluster size. These commands trigger the automatic reconfiguration of the set of distributed services implementing the SSI.

Concerning the portability issue, it is not possible to run an SSI in a heterogeneous cluster. The main reason is because on each cluster node needs to execute the same kernel binary (which is impossible in a heterogeneous environment).

Concerning the application suspend/restart capacity, the SSI provides a cluster wide process-level suspend (*SSI-appli-ckpt*) and process-level restart (*SSI-appli-restart*). However suspending the whole SSI cannot be done by the SSI itself and requires an external mechanism which is not yet available.

4 Combining Virtualization and Single System Image

In this section we expose 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 [1], (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 applications. To achieve our main objective, we analyze the benefits of the five capabilities enabled by virtualization (cf. Section 2.2).

Our analysis is composed of three different parts: (i) SSI & containers; (ii) SSI & type-I virtualization; and (iii) SSI & type-II virtualization.

4.1 Single System Image & Containers

Containers allow applications to be isolated from each other on the same node. Generally it also provides a suspend/restart and live migration mechanism. In addition, containers enable the dynamic management of allocated resources. For instance, it is generally possible to assign an IP address, to allocate memory and CPU time to each container.

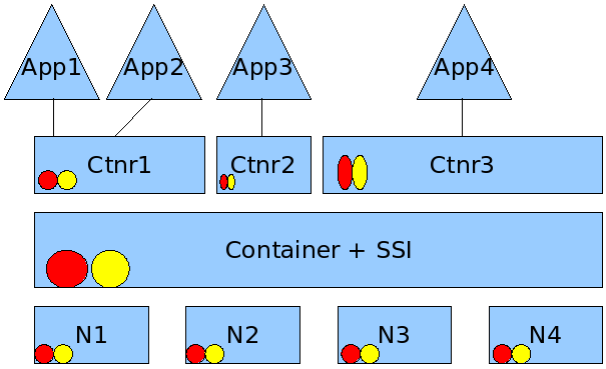

Figure 4: Containers Upon SSI
Little circles in the nodes represent resources (like memory etc.) and big circles upon the SSI box represent the virtualized resources providing by the SSI.

4.1.1 Container Upon Single System Image

Figure 4 depicts the architecture of a typical system running containers upon an SSI. With this architecture, the SSI abstracts resources of the distributed platform. 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 host OS is extended and becomes a distributed OS.

Since the application is running inside containers, the system complexity remains the same: a distributed system is still exposed to applications that needs to be based on explicit parallelism (*e.g.*, MPI-like applications) in order to take a full benefit of available resources.

Isolation Applications are isolated from the bare hardware by containers that are running on top of the SSI. Containers can be corrupted and therefore the global isolation of the system is limited.

Server Consolidation Container-level migration and SSI-level migration can be used to move applications between nodes of the distributed system. Moreover, since the SSI globally manages all resources, it is possible to change on demand the resources allocated to each container. This capabilities are very interesting for an Apache server administrator: according to the frequentation of a web site during the day, it is possible to allocate more or less physical resources to the cluster.

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 standard resource from the SSI point of view.

Virtual Machine Portability The SSI does not support execution on different hardware architectures. Moreover containers cannot create a virtual

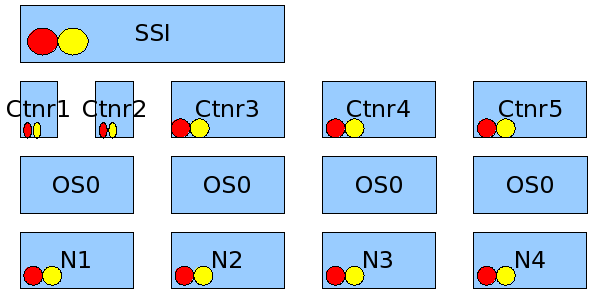

Figure 5: SSI Upon Containers
This architecture does not work because a container cannot host a fully-featured kernel.

hardware different from the hardware it is running on. Thus, the virtual machine portability is not validated.

Application Portability Thanks to the SSI, containers can be restricted to individual nodes or span multiple nodes. Therefore the nature of the application is not limited by the distributed architecture of the underlying platform. For instance, spanning to multiple nodes using the SSI shared memory, a unique memory space is exposed to the application; it is therefore possible to execute OpenMP-type application or an Apache server. At the opposite, a single container can be assigned to each process of a MPI application; each container having a unique IP; which enables the execution of standard MPI applications. Application portability is therefore guarantee by such an architecture.

4.1.2 Single System Image Upon Containers

Figure 5 presents the use of an SSI upon containers implemented in the host OS (OS0 in the figure).

The architecture is not realistic because no individual kernel can run in a container, only user-level applications can be hosted.

4.2 Single System Image & 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.

4.2.1 Type-I Virtualization Upon Single System Image

Figure 6 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 trans-

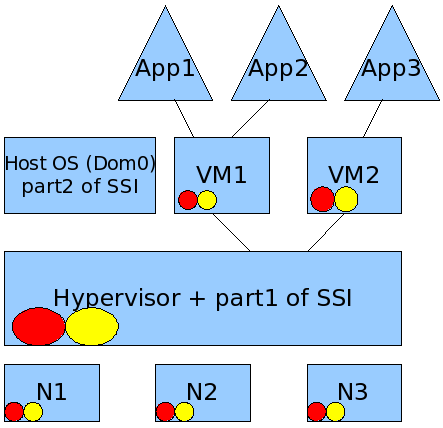

Figure 6: Type-I Virtualization Upon SSI
Little circles in the nodes represent resources (like memory etc.) and big circles upon the SSI box represent the virtualized resources provided by the SSI.

parently and globally manage resources (creation of an SMP illusion) and typically the resource allocated to VMs is not restricted to the local resources.

However, since type-I hypervisors are a minimalistic OS that do not include device drivers, the SSI that creates the SMP illusion for the hypervisor has to be split up into two parts: (i) the part running on the hypervisor for resource allocation to VMs and (ii) the part running on the privileged domain (*Host OS*) in order to extend the traditional device drivers for the global management of hardware resources. SSI typically manages resources extending OS capabilities; type-I virtualizaton deals with virtual hardware. The granularity is therefore different and no solution currently enables this architecture.

With such an architecture the platform complexity remains the same from the application point of view: the application still see a distributed platform and the application has to implement explicit parallel mechanism (MPI-like application) in order to take benefit of available resources.

Isolation The type-I hypervisor isolates applications from the bare hardware since applications are running in VMs. For instance if an Apache server is running in a VM, Apache is isolated from other applications running in other VMs and from the host OS. If a hacker is able to become root on one VM, only the local VM is compromised: isolation is validated.

Server Consolidation Type-I virtualization enables VM migration assuming applications are based on TCP for communications: if packets are lost during the migration, TCP deals with their retransmission; and the physical location of the VM can be updated changing the routing tables of the operating system (*e.g.* ARP tables). Therefore, 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.

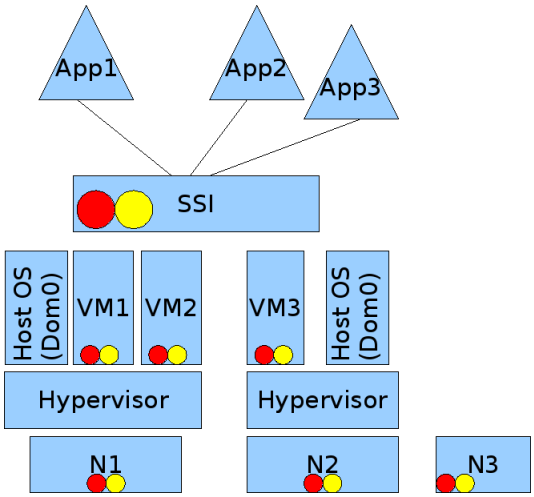

Figure 7: SSI Upon Type-I Virtualization
Little circles in the nodes represent resources (like memory etc.) and big circles upon the SSI box represent the virtualized resources providing by the SSI.

Suspend/Restart 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 partially validated.

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 completely achieved.

Application Portability Because the SSI is running on the side of the type-I hypervisor, a distributed system is still exposed to applications. Therefore, only applications designed to executed on clusters (*e.g.*, MPI-like applications) can be executed on such systems (an Apache server cannot take advantage of this architecture). However, the SSI can in theory aggregate distributed resources and expose them via a single VM. In this case, distributed resources are exposed via an SMP virtual machine, applications such as DSM-based applications can therefore take a transparent benefit of distributed resources. Application portability is therefore validated.

4.2.2 Single System Image Upon Type-I Virtualization

Figure 7 shows the architecture of an SSI upon the VMs of a type-I virtualization solution. In this case, an hypervisor is deployed on each cluster node and the SSI is executed in different VMs; each VM being potentially hosted by different hypervisors.

This architecture simplifies the complexity of the platform exposed to applications via the usage of the SSI, providing the illusion of an SMP system; and the type-I virtualization isolates everything from the bare-hardware. Such a solution therefore enables any kind of application (*i.e.*, not only MPI-like applications; an Apache server could take advantage of this kind of architecture) thanks to the global and transparent management of all resources.

Isolation The type-I virtualization isolated 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 too. Isolation is therefore partially achieved.

Server Consolidation The type-I hypervisor enables VMs migration and the SSI process migration between VMs. Moreover, the virtualization solution with a balloon mechanism [16] and the SSI with the global resource management capability enable a transparent and on demand modification of resources allocated to a specific application/virtual machine.

Suspend/Restart Both the virtualization and the SSI solution provides mechanisms for suspend/restart: in the first case it is possible to suspend/restart VMs, in the second case it is possible to suspend/restart processes running on top of the SSI. However, if a VM is suspended, the SSI considers the event as a node eviction, similar to a failure. Therefore, a "synchronization" between the virtualization solution and the SSI is necessary in order to suspend VMs without compromizing the execution of parallel applications in different VMs.

Virtual Machine Portability Today no type-I virtualization solution allows the emulation of an architecture at the VM level that is different to the bare hardware. Portability is therefore not achieved. However, it seems that this problem is "just" a problem of implementation because in theory nothing prevents the implementation of such a solution. If such a solution is implemented, it is possible to migrate VMs between nodes having different architecture. However, from the application point of view, because of the SSI and its charateristics, only one architecture can be used: the one supported by the SSI.

Application Portability Applications are actually running on top of the SSI, providing a SMP illusion. This enables the execution of MPI-like, OpenMP-like and Apache-like applications.

4.3 Single System Image & Type-II Virtualization

Type-II virtualization solution run VMs upon a host OS and generally provides live migration and suspend & resume capabilities.

4.3.1 Type-II Virtualization Upon Single System Image

Figure 8 shows the execution of VMs upon an SSI. The SSI globally manages all the distributed resources; the type-II virtualization can therefore allocate distributed resources to VMs on demand in a transparent manner for both the virtualization solution and applications. The SSI becoming the host OS for the

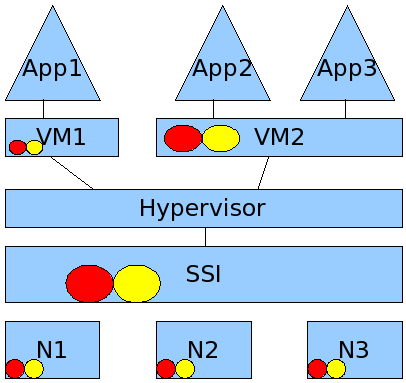

Figure 8: Type-II Virtualization Upon SSI
Little circles in the nodes represent resources (like memory etc.) and big circles upon the SSI box represent the virtualized resources providing by the SSI.

virtualization solution, SSI capabilities can also be used to manage VMs (*e.g.*, a VM is typically a process on the host OS that can be migrated by the SSI). The virtualization solution isolates applications from the SSI and the bare hardware.

A distributed system is still exposed to applications (and therefore need to be MPI-like applications).

Isolation The type-II virtualization isolates application running in the VMs from the SSI and the bare hardware. Therefore even if a VM is compromized, the SSI or the bare hardware cannot be compromized.

Server Consolidation In this architecture, the SSI provides an abstraction of the cluster resources for the VMs which can therefore be migrated. In case of node addition/removal, the SSI can change the resource assignment to the VMs for instance migrating VMs from one node to another.

Suspend/Restart Both the SSI and the type-II hypervisor provides suspend/restart capabilities: the virtualization solution can suspend/restart VMs and the SSI can suspend/restart processes that implement a VM. The suspend/restart mechanism is therefore duplicated in some way.

Virtual Machine Portability Since the SSI hosts the hypervisors, the hypervisor can only run of the hardware architecture supported by the SSI. It is therefore not possible to migrate virtual machine from one architecture to another.

Application Portability The type-II virtualization can emulate different hardware architectures at the VM level. All architectures can therefore be target by applications. Moreover, since the SSI federates all available resources, resources exposed to application in a given VM can aggregate resources from different physical nodes. For instance, local memory of different nodes can be aggregated and exposed into a VM. OpenMP-like application can therefore be executed using transparently this memory which is physically distributed (the

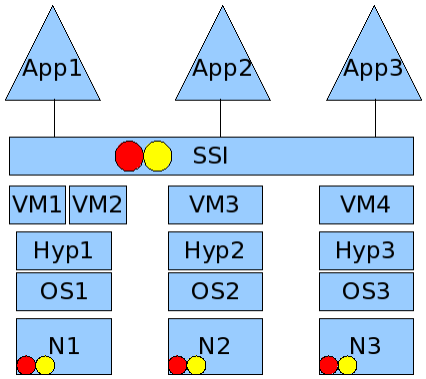

Figure 9: SSI Upon Type-II Virtualization

same for an Apache server). It is also possible to execute multiple VMs on which processes of MPI application can run. In that case, the virtual platform exposed through VMs is similar to a distributed platform, MPI-like applications can be executed without modifications.

4.3.2 Single System Image Upon Type-II Virtualization

Figure 9 shows the architecture of an SSI upon VMs. In fact, each node runs a VM, and the SSI is locally deployed upon this VM. The type-II virtualization protects the bare hardware isolating the SSI and the applications. The SSI abstracts the distributed platform via a global and transparent management of distributed resources.

Thanks to the SSI, an SMP illusion can be exposed to applications, enabling the execution of any kind of applications (including those that are not MPI-like applications such as DSM-based applications).

Isolation The type-II virtualization isolates both the SSI and the VMs: if a VM is compromized, the SSI can be compromized too but the hardware cannot be isolated. The SSI only abstracts the complexity of the distributed platform for the applications, globally managing available resources.

Server Consolidation In case of node addition/removal, there are two cases: (i) an *automatic reconfiguration of the SSI*, and (ii) a *VM live migration* on another node. With the automatic reconfiguration of the SSI, the migration of an application such as an Apache server is possible. With the VM live migration, the migration of a VM from a node to another in a transparent manner for both the SSI and the applications (like an Apache server) is possible. It means that only the VM migration can enable server consolidation.

Suspend/Restart The type-II virtualization enables the suspend/restart of both the SSI and the application running in a given VM. If the VM hosts the whole application it is possible to suspend/restart the application in a transparent manner. If the VM hosts only a part of a parallel application, the whole

application needs to be synchronized and then suspended or restarted. If TCP is used by applications running inside VMs, it is possible to loosly synchronized VMs, TCP manages lost network packets.

Virtual Machine Portability Each virtual machine has its own IP address which is preserved even in case of migration. Moreover, the type-II 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).

Application Portability The type-II virtualization enables the emulation of different hardware architectures. The architecture of the bare hardware can therefore be different from the architecture supported by the SSI, but the application can only run on the architecture supported by the SSI (by definition an SSI does not support hardware heterogeneity).

5 Lessons And Experiments

The lessons learnt from this work are the following.

5.1 Containers on Top of Single System Image Clusters

Using the container based solutions in a SSI, resource exposed to applications can span multiple cluster nodes. By providing the illusion that a cluster is a virtual SMP, the SSI system retains all the advantages enabled by containers on a real SMP machine in a cluster environment. Frontiers between cluster nodes are removed. In such a configuration, cluster reconfigurations are managed by the SSI reconfiguration mechanisms. The migration and suspend features provided by the container technology remain useful to migrate applications/services between nodes.

5.2 Single Image System & Virtualization

Virtualization and SSI technologies can combined in two ways, each bringing different advantages. On one hand, virtual machines can be executed on top of a SSI cluster; on the other hand, the SSI system can be executed on top of a virtual cluster built with a set of virtual machines.

Virtual Machines on Top of Single System Image Clusters This configuration provides the same isolation advantages as containers. Furthermore virtualization solves *application* portability issues. For instance, with virtual machines, it is possible to execute an application developed for processor technology “A” and OS “B” on top of a computer running a SSI OS based on OS “C” and developed for processor technology “D”. This means that any application binary can be executed on top of a SSI OS, provided that the appropriate

virtualization technology is available. In such a configuration, the global scheduler of the SSI OS is in charge of balancing load of the VMs on the cluster nodes.

Single System Image on Top of Virtual Machines Executing a SSI OS on top of a virtual cluster composed of virtual machines is also very attractive. The use of virtual machine migration and suspend functionalities enables a flexible, simple and on demand resource allocation to applications, but also system adaptation in case of cluster configuration changes (node addition and eviction). The idea is to dimension the virtual cluster size considering the largest cluster configuration that is anticipated to be required by the application and to execute the virtual cluster on a smaller physical cluster, having multiple virtual nodes on the same physical cluster node. If an application requires more resources and more physical cluster nodes, the virtual machines are simply migrated to remote cluster nodes. When the size of the physical cluster increases or decreases, the virtual nodes are simply migrated from one node to another using the migration operation provided by the virtualization technology. This way, the virtualization technology simplifies the reconfiguration as there is no need to reconfigure the SSI OS that remains on the same number of virtual nodes.

Moreover, it becomes possible to execute multiple different virtual clusters on the same real cluster, each of them been isolated from the other ones. This is beneficial for many kinds of applications. For instance, two OpenMP applications can be executed in an isolated way in two different virtual clusters (see Figure 10). The SSI OS provides the illusion of a shared memory, the virtualiza-

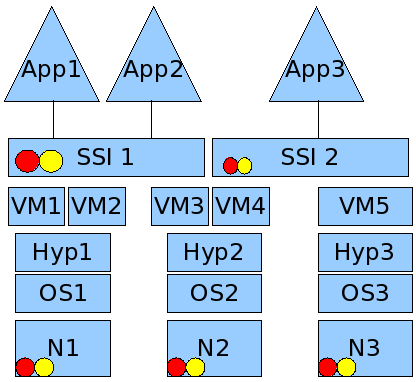

Figure 10: Isolation of Two Distinct SSIs
Virtualisation technologies offer a way to isolate several SSIs on the same cluster

tion technology provides isolation capabilities. Another example of application that can benefit from this configuration is web servers (such as Apache) which are often multithreaded. They can be deployed on several physical cluster nodes

transparently by combining virtualization and SSI technologies. Finally, this configuration also solves the portability issue of the SSI OS. Thanks to virtual machines, it is possible to execute a SSI OS based on OS "A" and available for processor technology "B" on a cluster composed of nodes providing processors of technology "C" and running OS "D".

When executing multiple virtual machines on top of a virtual cluster running a SSI OS, the advantages of the two configurations are combined: application portability, SSI OS portability, isolation, easy adaptation to application needs in terms of resources, easy adaptation to cluster reconfigurations.

Theoretical Summary Table 2 summarizes the different cases studied in this document. Cases 4 and 7 seem to show that SSI and VM are mixable to offer better flexibility.

	1	2	3	4	5	6	7
Isolation	-	N/A	+	+	+	-	+
Server consolidation	+	N/A	+	+	+	-	+
Suspend/restart	+	N/A	-	+	+	-	-
VM Portability	-	N/A	-	-	-	-	+
Application Portability	+	N/A	+	+	+	+	+

Studied Cases: (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: Automatic reconfiguration of Kerrighed; (7) SSI upon type-II: VM live migration.

Table 2: Summary of the different cases studied in this document.

Kerrighed [12] on the top of VMware [18]. The combination of a SSI OS with virtualization can be implemented with virtual machines of type-II and no porting effort is required in the current state of the technology. We made some experiments with Kerrighed [12] on top of VMs using VMware [18] Server 1.0.4. We ran our experiments on a cluster of Grid5000 [8]. The MrBayes [11] application, an MPI phylogenic solver, has been used. This application is mainly CPU intensive. The experiments consists in starting MrBayes and then simulating an hardware maintenance task by stopping all running VMs. Then, some VMs, hosting the SSI, have been migrated and then restarted in a transparent way for both Kerrighed and the application (MrBayes). Doing so, the system resumes correctly and the application completes successfully.

In order to test the portability of the VMs, we successfully start a Kerrighed cluster on the top of VMware VMs running on *Intel 32 bits* nodes and migrate all virtual cluster nodes to *AMD 64 bits* nodes.

5.3 Extensions

Multiple Combining This document mainly focuses on combining of VMs and SSIs based on a two level architecture. In such a context, we easily led some experiments based on VMware [18] and Kerrighed [12]. From our point of view, an extension would consist in investigating more complex architectures based

on several levels: VM/SSI/VM, SSI/VM/SSI, ... Figures 11 and 12 depict such ideas.

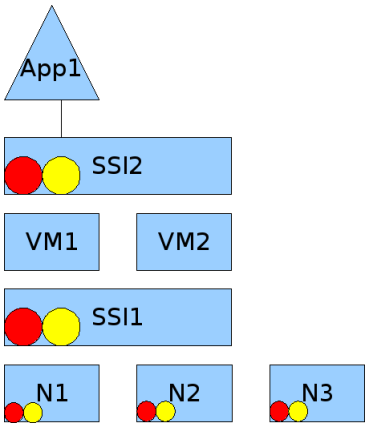

Figure 11: Architecture of an SSI on top of VMs on top of an SSI

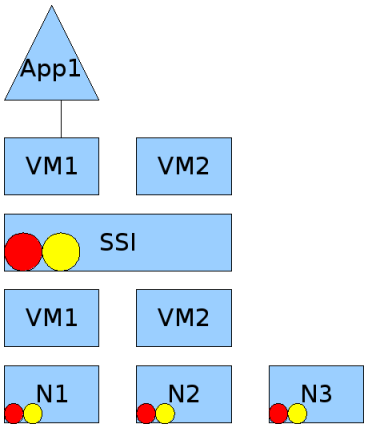

Figure 12: Architecture of VMs on top of an SII on top of VMs

In such architectures, portability is delivered by VMs whereas resource aggregation is provided by the SSI. As an example, a windows out-of-core application can benefit from a Linux SSI upon a windows cluster.

For instance, the configuration presented in Figure 12 shows that it is possible to execute a VM with a guest OS “A” available for processor technology “B” on an SSI based on OS “C” available for processor “D”. The SSI could also be run on top of VMs available for host OS “E” and for processor “F”. It seems that this kind of combining would allow an application written for a

given processor to take advantage of all the resources in a cluster even if this cluster is based on another processor architecture.

Debugging Capabilities Independently of the usage of both virtualization and SSI techniques for application execution, our experiments highlight an interest for debugging purposes: the execution of an SSI on top of VMs enables debugging of the SSI itself; the "simulation" of a large SMP machine using an SSI on top of VM running on different nodes enables SMP application debugging. These debugging capabilities cannot be provided in such a flexible way by any other solution we know.

6 Conclusion and Future Works

Nowadays, mutlicores processors, and soon manycore processors, have become mainstream in clusters. Morever, container and 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 OS for clusters irrelevant for the future?

Based on the current state of the art on SSI OS and on the container and virtualization techniques, we analysed different configurations combining the SSI OS approach with container and virtualization techniques in clusters.

From the analysis presented in this paper, we conclude that the virtualization and SSI OS complement each other. A full SSI OS makes transparent resource distribution in cluster nodes, providing the illusion of a virtual SMP machine, whereas the container and virtualization technologies provide flexibility in resource management. On-demand cluster reconfiguration is made easy thanks to the use of the VM suspend/restart and migration features.

Moreover, the virtualization techniques enable the portability of both the applications and the full SSI OS. As an example a Windows application running in a VM on top of a SSI OS can take advantage of a Linux-based SSI to make use of resources spread in several cluster nodes. Another example is the one of a Linux-based SSI running in a virtual environment on top of a cluster running Windows on each node. Such a configuration allows Linux based applications such as OpenMP applications or multithreaded web servers (*e.g.*, Apache) to take advantage of the virtual SMP provided by a SSI OS based on Linux even if the underlying cluster nodes run natively Windows. Both container and virtualization technologies provide application isolation, a capability not natively offered by the SSI OS technology. Hence, it is attractive to execute VM on top of a SSI OS to better isolate applications running concurrently on top of SSI OS.

From the experiments we conducted, it appears that the SSI OS can be very easily combined with the type-II virtualization technology with no porting effort. Combining type-I virtualization with SSI OS would require huge efforts to split the SSI OS in two parts, one integrated to the hypervisor and the other one integrated in the host OS. Thus, it is much more attractive to combine the

SSI OS with a type-II virtualization technique such as some VMware products than with a type-I virtualization technique such as Xen.

Moreover, even if individual cluster nodes become more and more powerful with their growing number of cores and an increasing amount of memory, there will always be applications requiring more cores or more memory than available in a single cluster node. This is the case of parallel applications and of multithreaded web servers such as Apache for instance. The multicore technology tends to make the shared memory model attractive for clusters (it was not the case in the past, message passing techniques being more efficiently supported in clusters). Hence, it also contributes to make the SSI technology attractive as an SSI OS such Kerrighed implements a virtual shared memory.

We have several directions for future work. We plan to extend our preliminary experimental evaluation with different kinds of applications: bags of tasks, parallel applications (MPI, OpenMP), servers (Apache with different configurations based on multiple processes or multiple threads). This would allow us to compare the behaviour of these applications on clusters running an SSI OS, running VMs or running one of combinations between SSI OS and virtualization that have been identified as attractive after the study presented in this paper. In particular, we plan to measure the performance of the applications in these different environments. It would also be interesting to further investigate the combination of the SSI OS with the type-I virtualization technology.

Kerrighed SSI OS provides a framework to easily plug global scheduling policies. Customized scheduling policies are needed in the context of an SSI OS used to execute virtual machines. Such policies to schedule VM are studied in projects such as Jaws [9]. This would be interesting to design scheduling policies for VM and to experiment them in the framework of Kerrighed.

Another work direction that we plan to investigate in the framework of the XtreemOS European project is the use of virtualization techniques in a Grid environment for commercial applications requiring strong isolation.

From a more theoritical 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 kinds of virtualization techniques, containers and the SSI OS.

References

[1] Apache. Apache software foudation welcome page, 2007. Available as http://httpd.apache.org.

[2] Amnon Barak and Oren La'adan. The MOSIX multicomputer operating system for high performance cluster computing. *Future Gener. Comput. Syst.*, Volume 13 (Number 4-5): pages 361–372, 1998.

[3] Paul Barham, Boris Dragovic, Keir Fraser, Steven Hand, Tim Harris, Alex Ho, Rolf Neugebauer, Ian Pratt, and Andrew Warfield. Xen and the art of virtualization. Bolton Landing, New York, USA, October 2003. SOSP'03.

[4] Fabrice Bellard. Qemu, a fast and portable dynamic translator. Technical report, USENIX Association, 2005.

[5] Douglas P. Ghormley, David Petrou, Steven H. Rodrigues, Amin M. Vahdat, and Thomas E. Anderson. GLUnix: A Global Layer Unix for a network of workstations. *Software Practice and Experience*, Volume 28 (Number 9): pages 929–961, 1998.

[6] GNU. Chroot, 2007. Available as http://www.gnu.org/software/coreutils/manual/coreutils.html#chroot-invocation.

[7] R. P. Goldberg. Architecture of virtual machines. AFIPS National Computer Conference, July 1973.

[8] Grid5000. Grid5000 welcome page, 2007. Available as http://www.grid5000.fr.

[9] Laura Grit, David Irwin, Varun Marupadi, Piyush Shivam, Aydan Yumerefendi, Jeff Chase, and Jeannie Albrecht. Harnessing virtual machine resource control for job management. March 2007.

[10] Erik Hendriks. BProc: the Beowulf Distributed Process Space. In *ICS '02: Proceedings of the 16th international conference on Supercomputing*, pages 129–136, New York, NY, USA, 2002. ACM Press.

[11] John P. Huelsenbeck and Fredrik Ronquist. *MrBayes : A program for the Bayesian inference of phylogeny.* http://golab.unl.edu/teaching/SBseminar/manual.pdf.

[12] Christine Morin, Renaud Lottiaux, Geoffroy Vallée, Pascal Gallard, David Margery, Jean-Yves Berthou, and Isaac Scherson. Kerrighed and data parallelism: Cluster computing on single system image operating systems. In *Proc. of Cluster 2004*. IEEE, September 2004.

[13] OpenMosix. Openmosix welcome page, 2007. Available as http://openmosix.sourceforge.net/.

[14] OpenVZ. Openvz welcome page, 2007. Available as http://wiki.openvz.org/Main_Page.

[15] Rolf Riesen, Ron Brightwell, Lee Ann Fisk, Tramm Hudson, Jim Otto, and Arthur B. Maccabe. Cplant. In *Proceedings of the Second Extreme Linux workshop at the 1999 USENIX Annual Technical Conference.*

[16] Joel H. Schopp, Keir Fraser, and Martine J. Silbermann. Resizing memory with balloons and hotplug. In *Proceedings of the Linux Symposium*, volume 2, page 313319, 2006.

[17] R. Scott Studham, Alan Cox, and Bruce Walker. Petascale single system image and other stuff, 2007. Available as http://www.cs.unm.edu/ fastos/05meeting/Fast%20OS%20PI%20meeting%20Studham.p p t.

[18] VMware. Vmware welcome page, 2007. Available as http://www.vmware.com.

Unité de recherche INRIA Rennes
IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex (France)

Unité de recherche INRIA Futurs : Parc Club Orsay Université - ZAC des Vignes
4, rue Jacques Monod - 91893 ORSAY Cedex (France)
Unité de recherche INRIA Lorraine : LORIA, Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex (France)
Unité de recherche INRIA Rhône-Alpes : 655, avenue de l'Europe - 38334 Montbonnot Saint-Ismier (France)
Unité de recherche INRIA Rocquencourt : Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex (France)
Unité de recherche INRIA Sophia Antipolis : 2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex (France)

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-6399