Virtualization Management for Grids and SOA

S. Freitag, R. Yahyapour
{stefan.freitag, ramin.yahyapour}@udo.edu
Dortmund University of Technology
44221 Dortmund, Germany

G. Jankowski, R. Januszewski
{gracjan, radekj}@man.poznan.pl
Poznan Supercomputing and Networking Center
61-704 Poznan, Poland

CoreGRID White Paper
Number WHP-0005
August 31st, 2008

Institute on Resource Management and Scheduling

CoreGRID - Network of Excellence
URL: http://www.coregrid.net

CoreGRID is a Network of Excellence funded by the European Commission under the Sixth Framework Programme

Project no. FP6-004265

Virtualization Management for Grids and SOA

S. Freitag, R. Yahyapour
{stefan.freitag, ramin.yahyapour}@udo.edu
Dortmund University of Technology
44221 Dortmund, Germany

G. Jankowski, R. Januszewski
{gracjan, radekj}@man.poznan.pl
Poznan Supercomputing and Networking Center
61-704 Poznan, Poland

CoreGRID WHP-0005

August 31st, 2008

Abstract

Virtualization introduces an abstraction layer on top of resources, so that physical characteristics are hidden from the user. At present CPU, network, and storage virtualization are the most widespread forms. Academic and/ or commercial data processing centres already use or are at the verge of adopting those virtualization technologies - mostly to increase their overall system utilization and to offer highly available services.

In the context of Grid Computing and SOA a broader field of application is possible. This paper discusses how Grid Computing and SOA can take advantage of resource virtualization by integrating it at different levels of the software stack. The study focuses on the impact in the area of resource management and scheduling by supporting virtualization. To this end, the meaning of jobs is extended towards a broader meaning. It is proposed to move in Grid Computing from simple job submission towards a (more challenging and complex) submission of virtual machines. Necessary changes, requirements and problems at the Grid Middleware and the LRMS level are further discussed.

In the long run both, user and resource provider, benefit from the adoption and combination of these technologies. Users no longer need to modify their application software depending on the offered execution environment at a resource. They just once create a virtual machine and directly include the necessary execution environment and the application level software as well. Subsequently the virtual machine can be submitted to every capable resource without requiring adjustments. On the other hand the virtualization layer reliefs the resource providers from selecting a fixed and limited execution environment and therefore increases the flexibility and system utilization as well. Due to strong interlinks between Grids and SOA results in the field of Grid Computing can be naturally extended to SOA environments. Similarly, the deployment and hosting of a service-based application can be managed within a virtual environment. Typically applications in SOA follow more a transaction-oriented, long running scenario in comparison to the typical space-shared job execution in most Grids. Here, the advantages of managing virtual environments with dynamic adaptation of the resource allocation is a major advantage too.

This research work is carried out under the FP6 Network of Excellence CoreGRID funded by the European Commission (Contract IST-2002-004265).

1 Introduction

Virtualization became a common technique for IT systems in many application environments. The idea dates back to the days of mainframe computing [Stra59] but gained increased popularity in recent years. The availability of powerful virtualization solutions gained special interest in managing commercial and academic data centres. Many centres already moved to increased virtualization of resources, or are in the process of adopting these techniques. The main goal of these activities is a server and/or storage consolidation for better system utilization, and lower cost of ownership. However, virtualization also provides new and powerful features in light of service-based infrastructures or Grids. This paper discusses the potential impact of virtualization technologies, possible usage scenarios and open research challenges or gaps.

Virtualization is often associated with virtual machines and corresponding CPU abstraction. However, the idea and current trends show a broader meaning that also includes different kinds of resources. The majority of current applications are in the area of CPU virtualization, storage virtualization and network virtualization. Generally, virtualization hides the physical characteristics of resources from the resource consumers, which can be end users or applications. It comprises resource decomposition to multiple smaller entities (e.g. in running multiple virtual machines on a single physical resource) or aggregation of smaller resources to appear as single entity (e.g. in storage virtualization by merging many SANs into a single virtual storage environment). Virtualization is used in many different contexts, which can be grouped into two main types:

- platform virtualization [Ram04], involving the simulation of whole computers, and
- resource virtualization, involving the simulation of combined, fragmented, or simplified resources.

In the area of Grid Computing virtualization gained more and more interest, but less in terms of service consolidation and/or increasing server utilization. Instead, virtualization allows addressing multiple problems in Grid systems, like coping with the heterogeneity of Grid resources, the difference in software stacks, enhanced features in resource management like a more general checkpointing or migration models. Adopting virtualization in smart ways gets us closer to real Grid computing with more flexibility in the type of applications and the resources to use.

Grid computing as a concept and Grid Middlewares as its implementations share a deficiency which virtualization can help to overcome. Both, concept and implementation, assume job execution on remote resources which fit the job requirements. Requirements here comprise on the one hand hardware and on the other hand software. In this paper we do not look further into hardware requirements. Due to numerous resources in a Grid the probability to find available resources is relatively high. However, the required software stack or a particular application environment is not always available on resources and as such limits the number of suitable resources. Virtualization allows easing this situation. Physical Grid resources do no longer need to fulfil the software requirements of a job. With virtualization e.g. the on-demand deployment of pre-configured virtual machines containing all the software required by a job is possible.

Similarly, there is an added flexibility to resource management and application execution. Running virtual machines can be manipulated by freezing them similar to checkpointing or by using live-migration features to move them from one set of physical resources to another while keeping the whole virtual execution environment. This does not only apply to virtual machines and compute resources but also the dynamic customization of network and storage environments.

SOA is besides Grid Computing another key technology in distributed systems. A considerable overlap between the goals of Grid Computing and the benefits of SOA based on Web services was pointed out in the recent year. Thus prospects of virtualization in context of SOA are analyzed as well to elaborate similarities and differences compared to the use with Grid Computing. This also links to the current trend of considering Cloud Computing which has many links to Grid computing but limits the scope to a specific service

provisioning. Most cloud solutions are already based on virtual machine and virtual storage management. Therefore, it is natural to provide a general view on the challenges and possibilities of resource management for virtualized environments.

In the following we will look further into the available features by virtualization and its benefit for Grids and SOA. To this end, we will first discuss the different types and features of virtualization in Section 2. Next, in Section 3 we briefly analyze the situation in Grid environments followed by a presentation of selected existing work in using virtualization in Grids. Section 5 discusses potential relevant use cases for applying virtualization techniques in Grids including models which are not yet available. In Section 6 we extract open research issues and technology gaps that need to be overcome for a successful exploitation of virtualization. Section 7 addresses the commonalities as well as the differences of virtualization in service-oriented architectures vs. Grids. We conclude with a brief summary and outlook.

2 Virtualization

Virtualization introduces an abstraction layer in the hard- and software stack. Within this layer resource decomposition to multiple smaller entities or aggregation of smaller resources to a single entity may happen. For the user who is situated above the virtualization layer the decomposition or aggregation is fully transparent. Examples for resource decomposition are VLANs [Vla06] which allow running several subnets over one network port. A SAN on the other hand is considered as an aggregation of several hard disks and looks like a single entity to the user. As seen in these examples virtualization is already used for storage and network, but also complete compute platforms can be virtualized.

2.1 Network virtualization

At network layer level virtualization is categorized in internal and external virtualization. External virtualization joins network partitions to a single virtual unit. Examples for this type of virtualization are VPNs and VLANs. A VPN needs for each participating network partition a gateway installed on its head node. Communication between nodes in different network partitions is tunnelled: as it reaches the outgoing gateway, it is encapsulated, routed by a virtual router to the destination gateway, there extracted and then forwarded to its destination node. For users inside a VPN the various network partitions seem like one large internal network.

Internal virtualization offers network-like functionality to software containers on a resource. This type of virtualization is often used in combination with virtual machines. E.g. in dom0 of a Xen host system besides physical network interface cards also virtual ones are present. These virtual Ethernet cards are connected to virtual interfaces (VIF) located inside the virtual machines. Configuration of the virtual devices allows the virtual machines to run in a bridged, routed or NATed network [Xen08a].

2.2 Storage virtualization

In storage, virtualization is a means by which multiple physical storage devices are viewed as a single logical unit. Virtualization can be accomplished at server level, fabric level, storage subsystem level, and file system level and in two ways: in-band and out-of-band [Tat03]. In this context in-band virtualization implies data and control flow over the same channel, whereas these flows are separated in out-of-band storage networks. This usually is achieved by separating data and meta-data into different places. An advantage of the latter approach is the availability of the full SAN bandwidth to I/O requests.

2.3 Platform virtualization

Resource providers benefit from platform virtualization in many ways. First and foremost it reduces hardware costs including costs for maintenance and helps to save energy and space. Further on a higher quality of service in terms of redundancy and security can be achieved. In general, user data can be hidden inside the virtual machine and it is therefore inaccessible for other users sharing the same physical machine at the same time. Also the dangers of a security compromise are limited; gaining root access to a virtual machine does not imply access to any other virtual machine running on the same host. Moreover, on a single resource support for various operating systems, even older or outdated ones can be achieved simply by providing the virtual machines.

Due to node failure or hardware maintenance large cluster installations are prone to frequent changes in resource availability; however users and applications expect high availability and reliability. In this case virtualization allows e.g. the eviction of applications from physical nodes to prepare for maintenance. This is done transparently to the individual applications running in the virtual machines [Bha08]. Also dynamic load balancing of virtual machines is possible in modern VM infrastructures.

For end-users a virtue amongst others lies in the flexible and fast deployment of virtual machines. Furthermore the environment provided by a virtual machine is always predictable and clean, therefore it is the first choice for software testing and deployment.

Nevertheless, virtualization implies also disadvantages. Communication from and to a virtualized resource needs to pass the virtualization layer; compared to a non-virtualized system a performance drop of up to 20% [Tat06] may occur. This drop was measured executing inter-cluster MPI jobs over WAN.

As stated earlier, if a virtual machine is compromised, other virtual machines running on the same host are not affected; they are isolated from each other. Yet a host machine is compromised by an attacker, access to all virtual machines on this host is possible. Therefore special security mechanisms must be installed on the physical machine.

2.4 Virtualization standards

The process of software (re-) distribution is often still done by setting up and then cloning physical or virtual machines. When using virtual machines this approach can be simplified by e.g. obtaining a pre-configured disk image containing all necessary software packages; only the process of cloning is left over to the site or system administrator. Hence the term virtual appliance was established and comprises a virtual machine with one or more disk images including an operating system, a bundle of application level software and additional meta-data for installation and configuration of the virtual machine [Vmw07a].

Virtual appliances allow a vendor-neutral packaging of virtual machines. The meta-data as part of the virtual appliance specifies how to install, configure and run the appliance on each virtual platform. Since no external dependencies have to be resolved on-demand deployment of virtual appliances is possible. Profiting from the charm of virtual machines, virtual appliances can be migrated between different resources and it is also possible to run multiple appliances on the same physical host.

To support the concept of virtual appliances in large scale e.g. for deploying distributed databases, the Open Virtual Machine Format (OVF) was introduced. It is a hypervisor neutral, extensible and open specification for packaging and distribution of collections of virtual machines[OVF08a, OVF08b]. Those virtual machines are interpreted as one functional unit/ virtual appliance. A XML wrapper contains information on required installation and configuration parameters for the virtual machines, enabling the appliance to be virtualization platform-independent.

3 Grid Computing

The primary focus in Grid Computing lies in secure resource sharing in terms of access to computers, software, and data in a dynamic environment. Sharing of those resources has to be fine grained and highly controlled[Fos01]. Moreover, Foster et al. [Fos02] proposed a three point checklist which characterizes a Grid more in detail:

- delivery of nontrivial qualities of service;
- usage of standard, open, general-purpose protocols and interfaces e.g. for inter-communication;
- coordination of resources that are not subject to centralized control.

The endeavours to implement the basic ideas behind Grid Computing ended up in the development of several Grid Middlewares as they exist at the present time. Development was in the early days mostly driven by specific communities requesting amounts of computational power and storage capacities. Here e.g. the community of high energy physicists at CERN were key players (developers and users) concerning the gLite Middleware. Currently this Middleware is deployed in about 250 sites in all EU countries and allows access to over 50,000 compute nodes.

Another Grid Middleware development started in 1997 with the German UNICORE project. It was extended in 2000 and renamed to UNICORE Plus and reached a mature production state. The goal of UNICORE Plus was to develop a grid infrastructure together with a computing portal for engineers and scientists to access supercomputer centres from anywhere on the Internet [Uni08]. In 2005, standards became important contributions to Grid middleware. UNICORE [Rie05] and also other Grid Middlewares moved and still move away from their once designed protocols and interfaces towards standards like OGSA-BES [Bes07] or OGSA-RUS defined by the Open Grid Forum. However, the great unification or common standards did not yet materialize. For impelling the use of common standards major transitions in code bases of the Grid Middleware are necessary. Due to the lack of additional investment in Grid Middlewares development and existing production status of Grids for active user communities introducing changes is an arduous task. But this change in the development process helps to increase the interoperability; job exchange across different Grid Middlewares is as a long term perspective. On this note, virtualization can contribute by introducing new aspects.

4 Existing work

At present virtualization is widely used on the one hand for service consolidation and on the other hand for achieving high availability for central Middleware services [Car07].

The Globus Alliance develops currently Virtual Workspaces [Kea05, Fos06]. These services provide interfaces to VM management functions based on WSRF protocol, whereas a Workspace is a combination of meta-data and a resource allocation request. At the moment only XEN [Xen08] as virtualization technology is supported. Here one good point is the dual-use of the underlying LRMS. If wanted it still accepts usual jobs and also virtual machines. It was shown in [Aga07] that a HEP application, namely the ATLAS application could be deployed by using the Virtual Workspace service.

The idea of dynamic reassembling of existing clusters for different applications is elaborated in [Eme06]. The LRMS, in this case MOAB in combination with Torque [Clr08, Clr08a], were modified, so that the batch system server is able to add or remove Xen based virtual machines to/ from Torque resources on demand. Xen Grid Engine (XGE) [Fal06] is also based on Xen hypervisor monitor and extends the Sun Grid Engine cluster management system.

In [Sot07] leasing resources for job execution was examined. It was shown that a VM-based approach performs in terms total execution time and average delay incurred by best-effort requests only slight worse than a non-VM-based scheduler supporting task pre-emption. [Sot08] proposes an architecture that makes on-demand short-term leasing of physical resources cost effective. An improved performance for resource providers and users was achieved when using virtualization for this purpose, even in the presence of overhead incurred by using virtual machines.

5 Using Virtualization in Grid Computing

Integration of virtualization and Grid Computing can be impelled at different levels. Evaluations at Grid Middleware and LRMS are carried out next.

5.1 Grid Computing Level

Currently, the gLite Middleware, which is used by the LCG HEP (high energy physics) VOs, has strong dependencies to Scientific Linux 3/ 4 as operating system. Getting gLite to work with other operating systems is a complex task. By encapsulating the Grid Middleware into a virtual appliance, resource providers can support the HEP VOs with minimal effort; the only requirement is support of a virtualization platform like XEN or VMware. Like the gLite Middleware, also other Grid Middlewares can be packaged as virtual appliances. Then resource providers no longer need to commit to a designated Grid Middleware, but can setup middlewares on demand. An approach more preferable is the user-based dynamic deployment of the Middleware services. This makes the resource providers' intervention in the deployment process superfluous and he no longer has to deal with software management and application requirements.

Before the semi or fully automated deployment process starts, the virtual appliances have to be made available to the local resource. One way to do this is utilizing the storage facilities which most Grid Middlewares offer. Those facilities can be used as repositories for the virtual appliances, whereas for file access and file transfer standardized protocols exist. Globus Toolkit [Glo08] and gLite Middlewares for instance provide access via OGSA-DAI respectively SRM. If the virtual appliance is not located on a storage facility, the user is requested to transfer it first there. Later on it will be transferred from the storage facility to the resource. From the users point of view directly sending the virtual appliance to a resource is feasible, but may result in serious performance degradation. E.g. in gLite Middleware data transferred from a user interface (UI) to a compute element (entry point at resource provider level) passes through the resource broker. As

many users may invoke the resource broker at the same time, the data transfers will slow down the broker unnecessarily.

In any case, before a user is able to submit jobs to a Grid, information about appropriate resources is needed - depending on who selects resources either at user or at resource broker level. In both cases a match-making process usually queries an information provider service which in turn delivers a list of resources satisfying the job requirements. The information representation often follows the GLUE scheme [Glu07] which describes in a precise and systematic manner available resources in Grids. For now the GLUE scheme offers no option to reflect the availability of a virtualization platform and its capabilities on compute clusters. But this is an indispensable requirement for the match-making process regarding virtual appliances.

The understanding of the term job has also to be revised. At present a job is written in well-defined job description languages like JDL and JSDL [Jdl06, Jsd07] or as a shell script. After combining virtualization and Grid Computing not only those jobs can be sent to a Grid, but also virtual machines or virtual appliances. From there the difference between a job and a virtual machine disappears: virtual machines can be considered as jobs and vice versa.

Besides compute capabilities Grids offer also possibilities to store data, e.g. the dCache Storage Element [Dca08] as part of gLite Middleware. Virtualization of Grid storage facilities brings several benefits for resource providers. Because of the declining costs of storage per megabyte, resource providers heavily increased their storage capacities. As a result of this complexity for managing storage increased disproportionate. At this point virtualization can help to reduce the complexity by hiding the storage backend from the operators and manage it on behalf of them. From the users point of view virtual storage can also ease data management. Obvious current development is not very far in this respect. But it is still expectable that we see major advancements in this area in the future. With the advent of virtual storage, faster networks etc. the location of data becomes less an issue. Still smart data management features will be needed. However, it will be less in the mind of the user.

5.2 LRMS Level

Integrating virtualization technology and all its features into the local resource management system (LRMS) layer is one of the key research areas in Grid Computing at the moment. Typically an LRMS supports job suspension and checkpointing out of the box, whereas virtualization offers the same features not for jobs, but for virtual machines. As a matter of fact, virtualization exhibits with live and deferred migration of virtual machines even one more feature.

Combining the aforementioned LRMS features (suspension, checkpointing) with migration helps the LRMS in dynamically changing the current resource allocation. Since LRMS usually cannot look into the future and act on the currently available situation, resource or workflow situation can change and a previously done RMS decision is not efficient anymore. In this case the earlier chosen resource allocation can be temporarily revoked e.g. by checkpointing and snapshooting jobs/virtual machines. The freed physical resources return to the pool of available resources for job execution and the jobs are put in a hold state until further processing. This procedure ensures a high efficiency for LRMS. Furthermore adjustments of resources (e.g. number of CPUs, RAM) assigned to a virtual machine are possible and allow dynamic updates in service quality. Depending on the virtualization platform this can be done e.g. on-the-fly or by suspending and resuming.

Checkpointing and migration have been subject of research for many years. Solutions [Elm04, Laa05] exist for HPC scenarios. However, the models are still quite special and do not work with arbitrary applications. Virtualization will probably improve this situation. Running applications (as virtual machines) could be stopped and resumed. The resource situation can be changed by temporarily running virtual machines in parallel with less priority. Or virtual machines could be transferred to different resources belonging to the same provider or maybe to another provider in the future.

In case not only one but many virtual machines are deployed as a functional unit, migrating one of them to another resource provider or even to another site of the same provider is a complex task. Communication between the different virtual machines in a closed system - like the LRMS is one - can be easily achieved by placing all virtual machines in the same subnet and/ or provide a dedicated VPN or VLAN for the functional unit only. The management of VPNs is simple: in case of user space daemons a VPN network can be automatically created via the job submission script [Kau06].

However, when a virtual machine as part of a functional unit migrates to another resource provider or site, the network connectivity within the functional unit has to be ensured continuously. To do so a VPN connection between the new location of the virtual machine and the original network must be established. The incoming and outgoing network traffic of the migrated virtual machine is then re-directed by the VPN server. Noteworthy is the need for on-demand creation of the VPN server on both sides of the network tunnel.

An essential requirement for using VPN are distinct subnet addresses of the interconnected networks; otherwise routing between the physical subnets is not feasible. Even more, a certain level of trust between resource providers is needed before they will couple their networks VPN - usually this level of trust can not be taken for granted. To conclude, in general smaller sites are not affected by the above mentioned disadvantages.

To bring the idea of creating virtual networks dynamically to an end, physical switches in compute centres could be enhanced so that they support dynamic creation and re-configuration of VLANs. For each set of virtual machines (a functional unit) a VLAN could be spanned in the compute cluster. This enforces on the one hand security by separation of the virtual machines on network level even if they are running on the same physical host. On the other hand in combination with the IEEE802.1p standard it enables providers to offer different QoS level concerning network characteristics.

Momentarily configuration of VLANs is still a manual task. A mechanism for automation would be of great help, but command line interfaces of physical switches from different vendors usually share only a minimal set of functions. An additional abstraction layer which provides virtualization platforms the means to change the network layout on demand is needed.

Not to speak of the dynamic changes in firewall rule sets. As a virtual machine migrates to another site all necessary ports for the services running in the virtual machine have to be configured. If the resource provider is aware of the services inside the virtual machine necessary ports can be opened in the firewall, but in general this is not the case. Even if providers are service-aware problems arise when default service ports are not used. Migration of virtual appliances as they contain meta-data which can be extended by adding the required open ports seems the better approach.

Next, after creation of an overlay network via VPN, access to file systems and local/ remote storage facilities has to be taken into account. We can not emanate from a global file system like AFS which is available across resource provider borders. Access to local storage at the original site has to be forwarded to new location of the virtual machine. E.g. many LRMS support the use of scratch directories for jobs. Those directories are used to store temporary data and are created on a per-job basis. Virtual machines can make use of such directories to download application data. Therefore it is clear that either the contents of the directory have to be copied to the new location of the virtual machine or by other means access to the once local directory has to be established before it starts execution. Also pending and on-going data transfers (e.g. to remote storage facilities) initiated from inside the virtual machine have to be redirected. Here solutions for migration of not only the VM, but also the data/ storage situation are needed.

5.3 Scenarios

5.3.1 Spanning virtual clusters over multiple physical resources

Physical resources of a provider are limited in number of compute nodes, storage capacity, and available software. Users demanding more compute nodes or other software than the available resources offer can not use the services of the provider and need to find another one which fits the users' needs. For the resource provider, especially smaller ones, this may result in under-utilization and therefore financial penalties.

One possibility to cope with this problem lies in increasing on demand capacities of the provider, whereas a dynamic increase is preferable over a static one. A static increase can be achieved by e.g. adding more compute nodes or hard disks. This investment in new hardware is only reasonable if the resource provider expects more of those larger jobs in the future. Otherwise under-utilization will occur due to the fact of the grown site size and the only temporary higher utilization the large jobs caused. A dynamic increase can be achieved by applying the concept of resource leasing. We can see such scenarios in commercial environments in which resource demand follow certain cycles during a month or a year. For better cost efficiency, it might make sense to temporarily increase the available resource set by adding third party resources on demand, for instance, to cope with year-end business or singular high workload.

To this end, the resource provider has to query an information service about available physical resources and their attributes (interconnection, software...) of other providers. If no compatible resources are reported by the information service, the provider is not able to increase the capacities and therefore has to postpone or deny execution of a job exceeding its (static) capacities. Virtualization can help to overcome the lack of finding compatible resource. The idea of leasing is extended by deploying virtual machines at remote sites - one of the main requirements is provision of a virtualization layer (at the remote site).

5.3.2 Dynamic relocation of grid jobs

After the jobs - which are wrapped in virtual machines - are deployed on assigned physical resources, on some conditions, it can happen that the re-location of these virtual machines is required. Furthermore, to increase the reliability level of the GRID environment the possibility of taking checkpoints of the computing process would be desirable. The leading VM implementations provide snapshots techniques which allow storing the intermediate state of the VM into non-volatile memory and later to restore the VM and its state even in different [1] hosting environment. This technique can be employed to take into practice both the migration and checkpoint-restart functionality.

The VMs re-location can be done either within the confines of the individual cluster (intra-cluster migration) or between distant clusters (inter-cluster migration). The reasons for the migration can vary a lot. However, the most likely scenarios are as follows. In case of intra-cluster migration one of the nodes can be required to be shut down for maintenance reasons or in case of inter-cluster migration even the whole cluster can be temporally cut off. In such a case, earlier VMs migration would allow us for saving a lot of CPU cycles, while on the contrary, without the migration functionality the computing would have to be started from the beginning.

The checkpointing technique can also be utilized by load-balancing algorithms in order to temporary pre-empt VMs of lower priority and execute VM of higher priority. Similarly, when a given job turns out compute longer than the previously assigned "wall-time" allows, the involved VM can be pre-empted and the job would be recovered when the user is granted another slot of time. One more possibility of utilizing checkpointing is in installations made up of commodity PC nodes which during day time are used in office as desktop computers and during night time are switched to work as computing clusters. In such a situation the cluster has to be frozen during office working hours.

[1] Different in terms of location and not internal architecture. Currently snapshots created by different producers' software are not compatible.

The migration functionality can be useful also because of different accounting and billing policies. In Grid environment the end users can be charged for utilizing the resources and the price associated with equivalent resources can be different (e.g. due to competition reasons). In such a scenario the matching/matchmaking algorithm can assume that an eminent factor in resource selection is the associated price. At first time, the selected resource provider can be the cheapest one but over time this may change and in such a case a demand for migration arises.

Depending on individual scenarios, the different actors can be responsible for triggering and managing the snapshots. In the simplest case, the user should be provided with the interface allowing to initiate snapshot, migrate and resume commands. But this would require the user to actively monitor and analyse the load and the state of the related resources. Therefore, even though the user driven migrations and checkpoints should be possible, the production grade environments should automatically monitor, snapshot, migrate and resume the VMs transparently to the user. Therefore, according to the scenario the automation of checkpointing and migration activities can be done on different levels. The snapshots can be triggered and managed on intra-cluster level by the LRMS or on inter-cluster level by Grid Resource Broker or by other Grid level management entity.

Assuming that the mechanisms for deploying and snapshooting VMs are already provided, the next issue is the integrated snapshot images management. In a fault-tolerance approach, the images should be stored in any external repositories in a way assuring that the failure of VM hosting environment does not affect the durability and availability of the image. Thanks to that, there is a possibility of resuming the VM in another place. The repository should also assure the integrity, consistency and confidentiality of the stored images. On one hand, the access to images should be limited only to the entitled users and on the other hand the user should be sure that the images are not altered in any way and that they are unavailable and completely obfuscated for other users. In load-balancing approach, the images can be moved either directly between involved sites or with help of any intermediate repository. In any case, the earlier mentioned security properties (durability, availability, confidentiality, integrity, consistency) have to be fulfilled as well.

Since the images can be quite large, the operation of transferring them across the network imposes some overhead. Therefore, smart repositories and replicas location management are necessary. The fast and short (in term of routing hops) connections between sites involved in migration or to the repositories in case of checkpoint like snapshots can significantly increase the overall performance of the considered environment. This is especially relevant in case of periodically performed checkpoints where each subsequent checkpoint accumulates to the final overhead. Another cost related issue is the garbage-collection of outdated images. The image becomes outdated when the correlated VM finishes its work successfully and there will not be need for resuming the VM for failure reason. As storage space is not infinite and for sure expansive, the mechanism removing the old images is required. The mechanism being activated on successfully finalization of given VM or alternatively any periodically launched garbage-collector can be provided. In the long run it can be foreseen that network capacity will increase to a point where the size of the images and thus the transfer time will be less of a problem anymore.

In case of distributed or parallel jobs, more than one VM can be involved in the computation. It means that the individual VMs have to communicate through the network and, additionally, it is likely that, according to SLA contract, a level of QoS must be established. This is typically necessary for co-allocating resources to assure that all resources (here virtual machines) are available at the same time. This leads to several problems in term of managing parallel snapshots and re-establishing the network state after a recovering.

6 Problems

Employing virtualization related technologies into Grid environment introduces a series of correlated issues which have to be considered in order to get a reliable and trustworthy system. The issues that we consider the most important ones are presented below.

A wide range of issues is related to the general topic of defending the system against malicious users. In case the user can provide his own VM images, there is a threat that the VM would contain any software which could try to compromise the hosting system or other of its VMs. Similarly, if the VM images are fetched from the pre-existing images repository, the dishonest user can install on the given VM any malicious software before the image is returned to the repository. Therefore, the hosting systems should treat the individual VMs as not trusted ones and the firewalls and other security mechanisms should protect the hosts from the individual VMs. Additionally, the images that are lent to users from the repositories should not return to the repository. We can say that images should be cloned before they are given out to the user and the image once cloned and assigned to any user should not return to the repository. The only exception from this rule is an image treated as checkpoint of intermediate results of working VMs. Such an image can be uploaded to a special image repository but the only user who has an access to the image is its owner. This policy protects both, the owner from the leak of sensitive information and the other user from the malicious VM images. The repositories themselves should provide mechanism for assuring that the once stored image is not modified. A reliable way of images consistency validation should be available. Additionally, when the individual virtual machines finish their work successfully the related checkpoint images are not useful any more so a kind of garbage collector of past images should be implemented.

Unfortunately, not only the end user can be malicious, the same concerns the resource provider. Therefore, if we do not trust either the resource provider or his clients, the virtual machines should contain software which would protect them from unauthorized access of any kind. Additionally, there is a really complex issue of assuring that the virtual resources of virtual machine are mapped to the physical resources of hosting system in a declared proportion. The user who acts within the VM has not a simple way to check what physical resources are assigned to his VM. Such a situation is not only inconvenient for end users but is also challenging from accounting subsystems point of view. It is also possible that in case of migration scenario the images are copied between hosting systems. Certainly, in this case the mechanism assuring the consistency and confidentiality of transferred images are also required.

Since the VM images are transferred between repositories and hosting systems, the smart policy of repository placement and replicas management is highly desired (especially as the images can be quite large). Additionally, in load-balancing or fault-tolerance related scenarios, when the virtual machine is restored on the remote site, the reconfiguration of network connections utilized by the VM can be necessary. It means that mechanism allowing for automatic reconfiguration of VPNs, firewalls or even storage related arrangements would be required. Similar issues regard the QoS and SLA related settings and contracts. Therefore, when the VM is to be resumed the previously negotiated parameters related to the involved services have to be restored as well.

Since the paper considers the virtualization in context of Grid environment, another important issue is cooperation/integration of VM related technologies with Grid Resource Brokers (in short brokers). Unfortunately, contemporary brokers are not able to perform resources matching/matchmaking basing on virtual machines related properties. Therefore, they have to be extended with the possibility of finding and assigning the “placeholders” to which the virtual machines can be submitted. Additionally, the brokers have to be able to perform basic management operations like suspending, resuming and migrating virtual machines.

To utilize the variety of VMs in Grid environment on large scale, the common interface to manage them is required. The VMs of any type should be available and manageable through one, high-level, well-abstracted SOA-based interface. Apart from deploying, suspending, resuming and migrating VMs such an interface should allow to setup the related VPNs and virtual storage elements. The initiative that

strives to provide framework with such high-level interface is Globus Alliance leaded Virtual Workspaces project [Kea07].

The long term stable and widely acceptable high-level interface to VMs should depend on some abstract interface to individual low-level VMs. Thanks to that, the high-level services can be written once and do not change anymore (excluding the security and bug tracking related changes) while the individual VMs can be incorporated by providing specialized "drivers" which translate the abstract interface to the interfaces imposed by the given VMs. Up to now, the most advanced project which works on such an interface is libvirt [Libv08].

Another highly wanted functionality is inter-operability of VMs of different vendors. In perfect situation the images and snapshots of VMs of different brands should be compatible and mutually interchangeable. Unfortunately, contemporary implementations of VMs vary a lot from each other and further standardisation on this field is required. This incompatibility between different VMs has to be taken into account during work on integration of Grid Resource Brokers with VMs. The brokers and matching algorithms have to be extended with the capacity of describing and selecting the VMs and the related images and snapshots.

One of inherent issues of distributed IT environments is global time synchronization. The heuristic algorithms to synchronize the clocks between individual nodes are known for many years and in most cases they are sufficient [Tan02]. However, in environments constituted from VMs the time synchronization becomes even more complicated. If the VM is being suspended, its local clock becomes frozen. Next, when after some time the VM is resumed, its clock becomes unsynchronized comparing to the global or real clock. It can affect the correctness of services and mechanisms which depend on globally synchronized clocks or just on time continuity. Examples of such time-sensitive instruments can be X.509 certificates, session's management or tools related to accounting and billing mechanisms.

Another problem is the determination of a so-called global consistent state while the distributed application is being checkpointed. Due to the lack of the global clock it is likely that the checkpoints of individual processes do not allow for resuming the whole job in a way assuring that no message is lost or that all re-sent messages are received in a proper way [Moo02]. The algorithms for finding the global consistent state within the set of independent, non synchronized checkpoints as well as the algorithms for forcing each checkpoint to be globally consistent have been proposed [Kal00]. Therefore, to checkpoint distributed program, which spreads out over more than one VM, the additional mechanisms to ensure the coherency of distributed checkpoint is required. However, if the processes which constitute the parallel program reside within the same VM the internal communication channels are suspended together with the whole VM so there is no risk to lose or duplicate the messages.

7 Using Virtualization in Service-based infrastructures

The use of Service-oriented Architectures (SOA) became a common and important programming paradigm and architectural choice for many distributed application scenarios. Technically, SOA describes a programming style for creating reliable and loosely coupled distributed systems which deliver functionality as a service. SOA components are usually implemented as modular services that can be discovered and used by clients. In general a service

- may be individually useful, or can be integrated to provide higher-level services
- communicates with its clients by message exchange
- advertises details such as their capabilities, interfaces, and policies

Prior to discussing the benefit of virtualization for Service-oriented Infrastructures (SOI) which facilitates the execution and management of SOA-based systems, we briefly look into the potential relation of SOAs and Grids. This subject is manifold; however two main perspectives can be seen: First, SOA as a mean to build Grids and Grid infrastructures; second, SOA as a paradigm for application use cases.

Considering the first dimension, it is clear that service-oriented architectures already became a main stream technology to build distributed systems. In its field it clearly displaced many other communication and programming paradigms. Similarly, Grids have been affected by SOA since several years ago. Many grid middleware services are already service-based. The transition from Globus GT2 to GT3 or GT4 showed this transition already many years ago. Most new protocol or grid service extensions are per se service based. This is also reflected by the progress in standardization bodies like the Open Grid Forum. New definitions and protocols use a service-based approach. Thus using SOA for Grids is common practise and not especially note-worthy. There is no special consideration necessary in this area for applying virtualization techniques.

However, considering the second dimension by using SOA as an application scenario, the situation is much different. Due to the historical evolution of Grids, there is a strong community from high-performance computing. This community is very much job-centric in terms that the computational infrastructure often refers to HPC resources like parallel computers or large-scale clusters. The computational or data-intensive jobs are typically batch driven as this yields high job throughput as well as high and efficient utilization of the resources. That means, jobs are overwhelmingly executed in a space-sharing fashion in which resources like CPUs or cores are devoted exclusively to a given job by a local resource management and scheduling system. While there are no idle resources available for a submitted job, such a job is queued. This model has been extended to computational Grid by enlarging the available resource landscape to more sites. However, the basic job profile stayed the same. However, we also see grids in other usage scenarios, especially in commercial enterprise data centres. Here, we have more diversity in application requirements. On one hand, we also see computational related jobs that are batch driven (e.g. crash test simulations, portfolio risk analysis). On the other hand, we have applications which are transaction oriented and typically longer running. Such applications can be executed in a more time-sharing fashion. For instance, these applications can be business processes that need to be executed continuously in a hosting environment. Enterprises also apply grid technologies in this field and are typically interested in adaptive IT solutions. The main goal of adaptive IT is to better exploit existing resources and add flexibility to scale and adapt the infrastructure to the dynamic demand. For instance, business processes in an enterprise resource planning system (ERP) may have high resource demand which may vary in certain monthly cycles (e.g. creating bills at month end), a large web shop may encounter varying customer demand during daily, weekly or annual cycles (e.g. year end business). To adapt the IT infrastructure dynamically and also offer resources to new application in an agile and dynamic fashion became common industry goal for large-scale data centres. The use of virtualization in the area of resource management is a key component for such scenarios too.

With the increased need of on-demand hosting such service based applications, the LRMS and the Grid scheduling services need new functionalities. Basically, from a theoretical viewpoint there is no real difference from allocating a computational batch job or a service-based application despite that the runtime requirements may differ. In real life, we see that service-based applications are often not independent entities. Often application processes require a multitude of service in a certain workflow. Some of these services are internal, manageable services which need to be provisioned and controlled; while others are external services outside of the scope of our application management; maybe provided by a third party (e.g. as a data source or information provider). That means, we may have two types of job management tasks: First, the scheduling, planning and execution of workflows which require services as part of the workflow. Second is the allocation and hosting of services which run for a longer time and may be repeatedly used in a transactional fashion.

The execution of services, by e.g. Enterprise Java Beans, follows a container concept in which a hosting environment has predefined ways to deploy new services. However, again real life shows that the independence of service implementation from hosting environments is still limited. Thus, the use of virtual machines which contain the whole hosting environment for web services provides a higher degree of flexibility. Similar to using VMs for computational batch jobs in Grids, we can use VMs for hosting service-based applications. The use of VMS as a cornerstone for service-oriented infrastructure is a clear industry trend. The virtual infrastructure allows scaling and dynamic resource allocation. Another point to mention is that typically the provisioning of web services requires a multi-tier architecture which, independent on whether it is 3- or 4-tier, requires an application server and a database. Thus we need the co-allocation of resources during resource management of hosting these servers. Again, virtualization allows us to encapsulate the complexity of the specific configuration of these servers and let the Grid RMS concentrate on managing the VMs.

Cloud computing found major attention recently as several commercial players started to offer corresponding service offerings. The scenario of cloud computing is very much similar to grid computing. However, the scope has been more limited and restricted in comparison to the grand vision of Grids. This made the problems at hand more tractable and ultimately allowed easier and faster solutions while research in Grids had and still has to cope with more challenges. For instance, Grids consider multi-provider scenarios in a large-scale, global environment. The applications at hand can be almost arbitrary and the resources and services are not limited to specific types. Clouds on the other hand, are very specific offerings like Amazon EC2 or S3 which can be easily used and accounted for computation and storage. It is assumed that in the long run, there will be a convergence of these approaches as they tackle the same problem space. Most existing cloud offerings heavily leverage virtualization to provide its services. Therefore, our aforementioned considerations also apply to Cloud computing in all parts. Providers of clouds also need to manage their resources in an effective and automated way. Similarly, consumers/users of clouds need management functionality to easily deploy applications to cloud services and monitor the execution.

Summarizing, there are strong interlinks between SOA, or more precisely SOI, Clouds and Grids. There are several differences. However, these differences are relatively minor as we see a common trend of convergence in these areas. Clearly, the logic of resource management and scheduling strategies will have to deal with all these requirements. Looking on the use and benefit of virtualization techniques, there is not much difference at all. Virtualization allows us to abstract from the internals of applications or services and provides a powerful manageability of the infrastructure. Our previously made assessment of scenarios and problems also apply for these SOA use-cases.

8 Conclusion

In this paper we tried to provide an overview on the current and future use (and benefit) of virtualization techniques for Grids and SOA. The selection of usage scenarios may not be extensive but probably gives a quite thorough list of envisioned changes on how Grids and SOAs are used or deployed. In our view this trend is not actively steered or influenced by the Grid and SOA communities. Instead, we see that the common trend of moving towards virtualization in data centres is happening right now and will stay with us for a long time. This affects how we deal with resources in general, including Grids and SOAs. The paper is intended to provide a summary on the effect of this trend and provide food of thought for current developments.

We presented several scenarios which we see as relevant for adopting virtualization in the light of Grids and SOA environments. Some of these scenarios are currently under development by different groups or stakeholders; corresponding results are becoming available in the near future. Some other scenarios need more technological advancements in different areas. Thus, it is unlikely that single research groups will be able to overcome those challenges in a short time frame. Here, we expect an evolutionary approach towards these scenarios when partial features become commonly available.

The presented scenarios provide inherent benefit in managing Grids and SOA environment. Thus, we are confident that this technological evolution will eventually lead to the broad adoption of these techniques and these usage scenarios. The current interest and success in cloud computing was made possible through the adoption of virtualization. Cloud computing limits some of the challenges and difficulties that Grid computing is facing. The service provisioning is done with tighter restrictions and leaves out key aspects in managing heterogeneity and cross-administrative domain challenges. However, at the same time, clouds address a sweet spot in a business relevant market. Eventually, the different trends tackle the same problem space and will end in converged solutions. Customers of clouds will request more flexibility in provider offerings including dynamic migration, individual service levels, and free combination of different provider offerings. Similarly, Grids and their current users will adopt more virtualization models and become more independent from the specifics of resources and applications. Thus, we believe that the presented models will become key use cases in the near and mid-term future.

Nomenclature

BES Basic Execution Service

CERN Conseil Europen pour la Recherche Nuclaire

DAI Data Access and Integration

ERP Enterprise Resource Planning System

GLUE Grid Laboratory Uniform Environment

HEP High Energy Physics

HPC High Performance Computing

JDL Job Description Language

JSDL Job Submission Description Language

LCG LHC Computing Grid

LRMS Local Resource Management System

MPI Message Passing Interface

NAT Network Address Translation

OGF Open Grid Forum

OGSA Open Grid Service Architecture

OVF Open Virtual Machine Format

QoS Quality of Service

RMS Resource Management System

RUS Resource Usage Service

SAN Storage Array Network

SGE Sun Grid Engine

SLA Service Level Agreement

SOA Service Oriented Architecture

SOI Service-oriented Infrastructures

SRM Storage Resource Manager

UI User Interface

VIF Virtual Interface

VLAN Virtual Local Area Network

VM Virtual Machine

VMM Virtual Machine Manager

VPN Virtual Private Network

WSRF Web Service Resource Framework

References

[Aga07] Deploying HEP Applications Using Xen and Globus Virtual Workspaces. A. Agarwal, A.Charbonneau, R. Desmarais, R. Enge, I. Gable, D. Grundy, A. Norton, D. Penfold-Brown, R. Seuster, R.J. Sobie, D.C. Vanderster. In Proceedings of Computing in High Energy and Nuclear Physics. September 2007.

[Bes07] OGSA Basic Execution Service Version 1.0. I. Foster, A. Grimshaw, P. Lane, W. Lee, M. Morgan, S. Newhouse, S. Pickles, D. Pulsipher, C. Smith, M. Theimer. http://forge.gridforum.org/projects/ogsa-bes-wg. Last access: July 15, 2008.

[Bha08] Virtual Cluster Management with Xen. N. Bhatia, J. S. Vetter. In Lecture Notes in Computer Science, Volume 4854/2008, pages 185-194. 2008.

[Bue06] Virtualizing a Batch Queuing System at a University Grid Center. V. Bge, Y. Kemp, M. Kunze, O. Oberst, G. Quast. In Lecture Notes in Computer Science, Volume 4331/2006, pages 397-406. 2006.

[Car07] Management of a Grid Infrastructure in GLITE with Virtualization. M. Crdenas, J. Prez-Griffo, M. Rubio, R. Ramos. 1st Iberian Grid Infrastructure Conference, May 2007.

[Chi05] Deployment of Grid Gateways using Virtual Machines. S. Childs, B. Coghlan, D. O'Callaghan, G. Quigley, J. Walsh. In Lecture Notes in Computer Science, Volume 3470/2005, pages 761-770. 2005.

[Chi06] A virtual TestGrid, or how to replicate a national Grid. S. Childs, B. Coghlan, J. Walsh, D. O'Callaghan. ExpGrid workshop at 15th IEEE International Symposium on High Performance Distributed Computing. February 2006.

[Clr08] Torque Overview. Cluster Resources, Inc. http://www.clusterresources.com/pages/products/torque-resource-manager.php. Last access: July 15, 2008.

[Clr08a] MOAB Cluster Suite. Cluster Resources, Inc. http://www.clusterresources.com/pages/products/moab-cluster-suite.php. Last access: July 15, 2008.

[Dca08] dCache Project homepage. http://www.dcache.org. Last access: July 15, 2008.

[Elm04] Checkpointing for Peta-Scale Systems: A Look into the Future of Practical Rollback-Recovery. E. Elnozahy, J. Plank. In IEEE Transactions on Dependable and Secure Computing, Volume 01, No. 2, pages 97-108. 2004.

[Eme06] Dynamic Virtual Clustering with Xen and Moab. W. Emeneker, D, Jackson, J. Butikofer, D. Stanzione. International Symposium on Parallel and Distributed Processing and Applications (ISPA) Workshops 2006. September 2006.

[Fal06] Xen and the art of cluster scheduling. N. Fallenbeck, H. Picht, M. Smith, B. Freisleben. In Proceedings of the 2nd International Workshop on Virtualization Technology in Distributed Computing. 2006.

[Fos01] The Anatomy of the Grid - Enabling Scalable Virtual Organizations. I. Foster, C. Kesselman, S. Tuecke. In Lecture Notes in Computer Science, Volume 2150/2001, pages 1-4. 2001.

[Fos02] What is the Grid? A Three Point Checklist. I. Foster. 2002

[Fos06] Virtual Clusters for Grid Communities. I. Foster., T. Freeman, K. Keahey, D. Scheftner, B. Sotomayor, X. Zhang. In Proceedings of the 6th International Symposium on Cluster Computing and Grid (CCGRID). 2006.

[Glo08] Globus Toolkit Project Homepage. http://www.globus.org/toolkit/. Last access: July 24, 2008

[Glu07] GLUE Schema Specification 1.3 Draft 3. S. Andreozzi et al. http://glueschema.forge.cnaf.infn.it/Spec/V13. Last access: July 15, 2008.

[Jan08a] Improvement on Fault-Tolerance and Migration Facilities within the Grid Computing Environment - Integration With The Low-Level Checkpointing Packages. G. Jankowski, R. Januszewski, R. Mikolajczak , J. Kovacs. In Grid Middleware and Services – Challenges and Solutions, Volume 3, pages 305-317. 2008.

[Jan08b] The Grid Checkpointing Architecture. G. Jankowski, R. Januszewski, J.Kovacs, R. Mikolajczak. CoreGRID White Paper - WHP-0003. 2008.

[Jac08] Are VM Environments Open to Attack. J. Germain, LinuxInsider, http://www.linuxinsider.com/rsstory/63888.html. Last access: July 30, 2008.

[Jdl06] Job Description Language Attributes Specification for the gLite Middleware Version 0.8. F. Pacini. https://edms.cern.ch/file/590869/1/. Last access: July 15, 2008.

[Jia03] Violin: Virtual Internetworking on Overlay Infrastructure. X. Jiang, D. Xu. In Proceedings of the 3rd International Symposium on Parallel and Distributed Processing and Applications (ISPA). July 2003.

[Jsd07] Job Submission Description Language (JSDL) Specification, Version 1.0. A. Anjomshoaa, M. Drescher, D. Fellows, A. Ly, S. McGough, D. Pulsipher, A. Savva. http://www.gridforum.org/documents/GFD.56.pdf. Last access: July 15, 2008.

[Kal00] A survey of checkpointing algorithms for parallel and distributed computers, S. Kalaiselvi and V. Rajaraman, Sadhana, Vol. 25, Part 5, October 2000, pp. 489-510.

[Kau06] Building Mini-Grid Environments with Virtual Private Networks: A Pragmatic Approach. C. Kauhaus, D. Fey. In Proceedings of International Symposium on Parallel Computing in Electrical Engineering, pages. 111-115. September 2006.

[Kea05] Virtual Workspaces: Achieving Quality of Service and Quality of Life in the Grid. K. Keahey, I. Foster, T. Freeman, X. Zhang. Scientific Programming Journal - Special Issue: Dynamic Grids and Worldwide Computing, Volume 13, No. 4, pages. 265-276. 2005.

[Kea07] Virtual Workspaces for Scientific Applications. K. Keahey, T. Freeman, J. Lauret, D. Olson. In Journal of Physics: Conference Series, Volume 78. 2007.

[Krs04] VMPlants: Providing and Managing Virtual Machine Execution Environments for Grid Computing. I. Krsul, A. Ganguly, J. Zhang, J. Fortes, and R. Figueiredo. In Proceedings of the 17th Annual Supercomputing Conference (SC). 2004.

[Laa05] Transparent Checkpoint-Restart of Distributed Applications on Commodity Clusters. O. Laadan, D. Phung, J. Nieh. In Proceedings of 7th IEEE International Conference on Cluster Computing 2005. 2005.

[Libv08] libvirt Homepage: http://libvirt.org/. Last access: July 15, 2008.

[Men06] Optimizing Network Virtualization in Xen. A. Menon, A. Cox, W. Zwaenepoel. In Proceedings of USENIX'06 Annual Technical Conference, pages 15-28. 2006.

[Moo02] A Survey of Rollback-Recovery Protocols in Message-Passing Systems, E. N. (MOOTAZ) Elnozahy, Lorenzo Alvisi, Yi-min Wang, David B. Johnson, ACM Computing Surveys, Vol. 34, No. 3, September 2002, pp. 375-408.

[OVF08a] The Open Virtual Machine - Whitepaper for OVF Specification Version 0.9. VMware, XenSource. 2007.

[OVF08b] OVF - Open Virtual Machine Specification Version 0.9. VMware, XenSource. 2007.

[Ram04] Virtualization - Bringing Flexibility and New Capabilities to Computing Platforms. R. Ramanathan, F. Bruening. Technical Paper. Intel Corporation. 2004.

[Rie05] Standardization Processes of the UNICORE Grid System. M. Riedel, D. Mallmann. In Proceedings of 1st Austrian Grid Symposium 2005, pages 191-203. 2005.

[San05] Deploying Virtual Machines as Sandboxes for the Grid. S. Santhanam, P. Elango, A. Arpaci-Dusseau, M. Livny. In Proceedings of the Second Workshop on Real, Large Distributed Systems (WORLDS), December 2005.

[Sot07] Enabling Cost-Effective Resource Leases with Virtual Machines. B. Sotomayor, K. Keahey, I. Foster, T. Freeman. In Proceedings of the 16th International Symposium on High Performance Distributed Computing 2007 - Hot Topics session. 2007.

[Sot08] Combining Batch Execution and Leasing Using Virtual Machines. B. Sotomayor, K. Keahey, I. Foster. In Proceedings of the 17th International Symposium on High-Performance Distributed Computing 2008. June 2008.

[Stra59] Time sharing in large fast computers. C. Strachey. In Proceedings of the International Conference on Information Processing, UNESCO, pages 336-341. 1959.

[Sun04] Towards Virtual Networks for Virtual Machine Grid Computing. A. Sundararaj, P. Dinda. In Proceedings of the 3rd USENIX Virtual Machine Technology Symposium (VM). 2004.

[Tan02] Distributed Systems: Principles and Paradigms, Author: Andrew S. Tanenbaum, Maarten van Steen, Prentice Hall, ©2002, ISBN:0130888931

[Tat03] Virtualization in a SAN. J. Tate. RedBooks Paper, IBM. http://www.redbooks.ibm.com/redpapers/pdfs/redp3633.pdf. Last access: July 15, 2008.

[Tat06] Making Wide-Area, Multi-Site MPI Feasible Using Xen VM. M. Tatezono, N. Maruyama, S. Matsuoka. International Symposium on Parallel and Distributed Processing and Applications (ISPA) Workshops 2006. September 2006.

[Uni08] ZAM Project UNICORE Plus. http://www.fz-juelich.de/jsc/cooperations/unicoreplus. Last access: July 23, 2008.

[Vla06] IEEE 802.1: 802.1Q - Virtual LANs. IEEE Computer Society. http://www.ieee802.org/1/pages/802.1Q.html. Last access: July 15, 2008.

[Vmw07a] Best Practices for Building Virtual Appliances - Whitepaper. VMware. 2007

[Vmw08] VMware Homepage. http://www.vmware.com/. Last access: July 15, 2008.

[Xen08] Xen Project Homepage. http://www.cl.cam.ac.uk/Research/SRG/netos/xen/. Last access: July 15, 2008.

[Xen08a] XenNetworking. http://wiki.xensource.com/xenwiki/XenNetworking. Last access: July 21, 2008.