

# Rapid and Minimally Invasive Deployment of Grid Middleware by means of Native Virtualization

Markus Baumgartner\*, Marco Descher†

**Abstract.** *While harvesting existing resources for the grid is desirable, in practice it is rarely achieved due to heterogeneous environments, shortcomings of the middleware, and restrictive software specifications. Native virtualization allows us to increase the number of grid resources by exploiting resources otherwise not accessible. In this paper, we present a proof-of-concept implementation of this approach based on the VMware virtualization software. It comprises ready-to-use virtual machine images for a gatekeeper node or Computing Element and job execution nodes or Worker Nodes that can be deployed onto an existing computing infrastructure without impairing its primary purpose. The prototype was successfully used to integrate an existing computing lab into a grid. Our measurements show that the performance penalty of native virtualization is small unless heavy use of the disk is made and therefore is negligible for many applications.*

## 1. Introduction

A huge number of institutions operate computing facilities potentially available for grid computing, that however are dedicated to another specific purpose. Up to now the exploitation of these resources for grid computing was hindered due to the following reasons:

- The computers are dedicated to a non-grid purpose and heterogeneous in nature, but their gridification must not make them subject to comprehensive modifications.
- A grid resource typically has to accord with a certain software specification comprising middleware, libraries and applications.
- The amount of abstraction provided by current grid middleware is not sufficient to utilize all available hardware architectures and software platforms. In practice, grid operators therefore specify certain operating systems suitable for the purpose of their grid.

A method to ease the usage of these non-dedicated resources using virtualization is shown in this paper. Our implementation is tailored to meet the requirements of and to be used within the Austrian Grid [1] but the concept can be applied to other grids where cycle harvesting is desired but restricted by heterogeneous architectures or the need to conform to software specifications.

---

\*Institute of Graphics and Parallel Processing, Johannes Kepler University, Linz, Austria, email: mb@gup.jku.at

†Research Center Process and Product Engineering, Fachhochschule Vorarlberg, Dornbirn, Austria, email: marco.descher@fhv.at

## 1.1. Server Virtualization

Virtualization is a broad term that is applied to a range of different topics. We therefore confine the definition to the specific application within our paper: *Server virtualization* is the creation of one or more virtual instances of a guest operating system machine within a host system.

There are four different subtypes of server virtualization:

- *Emulation* provides the functionality of a specific target hardware completely in software. This allows to emulate an architecture using a completely different architecture (E.g.: PearPC [2]).
- *Native or full virtualization* simulates just enough hardware for an unmodified operating system to run. The guest OS must be designed for the host system CPU type (E.g.: VMware Player [3]).
- *Para-virtualization* does not simulate hardware. A special API that requires OS modifications is provided instead. The guest OS is aware that it is running in a virtualized environment (E.g.: Xen [4]).
- *Operating system level virtualization* virtualizes a physical server at OS level. This allows for multiple isolated and secured server instances on a single physical server (E.g.: Linux-VServer [5]).

As our main objective is to break the operating system boundaries and hence enable different operating system environments to be used as resources for grid computing, native virtualization is the right method. It enables us to introduce a homogeneous gridified guest OS environment, running on heterogeneous host OS environments in a non-intrusive and rapid deployable way. Currently this allows us to exploit Linux and Windows systems running on x86 architecture. Due to the recent change of Apple Computer Inc. to the x86 architecture, we expect to be able to use resources running Apple OS X in the near future too.

## 1.2. Current Applications of Server Virtualization in Grid Computing

GILDA [6] is a virtual grid laboratory providing an infrastructure for grid training and dissemination. GILDA has installed a fully operational grid, the GILDA testbed, based on the gLite [7] middleware and its predecessor LCG. The testbed offers different user interfaces to account for the varying levels of trainees' expertise, including an intuitive web interface as well as a command line user interface for advanced trainees. Auxiliary services include a certification authority to issue grid certificates for trainers and trainees, a monitoring system to inspect the status of the testbed and its resources, and a virtual organization accommodating trainees and trainers. The services offered by GILDA also include virtual images [8] based on the VMware virtualization software. They contain pre-installed distributions of a Linux operating system and the gLite middleware. Separate images are available for different components of gLite. Those images can be leveraged to simplify the set-up of a training infrastructure or as a reference for installing the middleware.

In addition to being a means of efficiently spreading and deploying pre-installed middleware distributions, virtualization techniques have also been proposed in the area of Dynamic Virtual Environments, an example is given by Foster et al. in [9], where virtual images are used to provide application-specific environments on a per-job basis.

## 2. Materials and methods

### 2.1. The virtualization environment

We considered four different native virtualization environments: VMware Server and Player, Parallels Workstation [10] and QEMU [11]. The decision was made according to the requirements presented in table 1.

Requirement:	VMware Server	VMware Player	Parallels Workstation	QEMU
Free Software	X	X		X
No License restrictions		X	non-free	X
Host OS support (Windows, Linux)	X	X	X	X
SMP functionality	X	X		X

Table 1. Comparison of virtualization environments

The list of requirements was arranged due to the necessities posed upon the software:

- The virtualization software has to be free, as not to impose any costs for provisioning and must not be subject to license restrictions, prohibiting its usage for compute intensive respectively grid applications.
- The virtualization software has to be running on both Linux and Windows operating systems, to cover a considerable portion of the currently available operating systems.
- The virtualization software has to support symmetric multi processing to support systems equipped with either multiple processors or dual core systems in order to fully exploit the compute capability of the system.

Both VMware Player and QEMU satisfy our requirements. As VMware is already in use by the GILDA project we followed this solution. VMware is defined as native virtualization software. It however makes available parts of para-virtualization technology by providing optional enhanced device drivers (packaged as VMware Tools) [12].

### 2.2. Overview

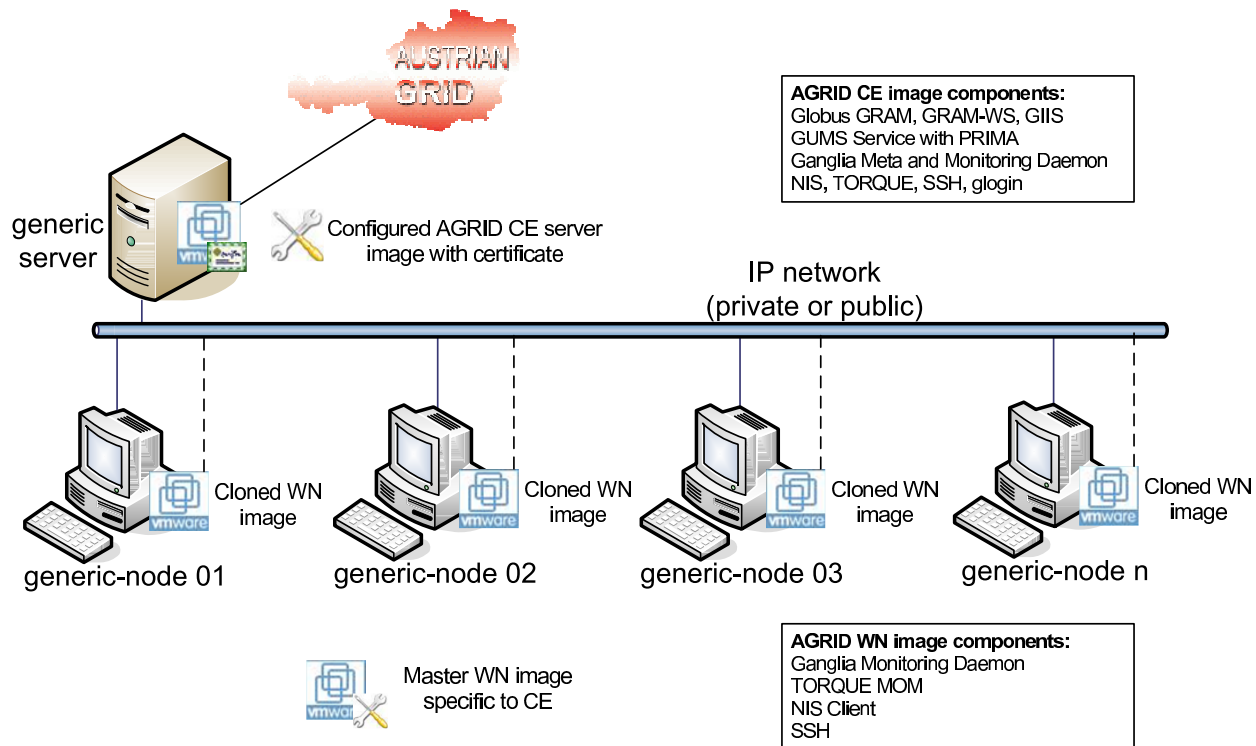
The Austrian Grid Software Specification [13] makes no distinction between the inner components of a resource in order to keep resources transparent to users and developers. For a resource operator, however, the structure of a resource is relevant. In most cases, a resource is either a single system or a cluster. In the latter case, the front-end node and compute nodes in the cluster usually perform different roles. To refer to these roles and therefore to the components of a resource, we have adopted the naming scheme already established within various existing grid projects, such as LCG, the DataGrid, and EGEE.

In this paper we introduce a native virtualization-based solution for deploying a Computing Element (CE) and a number of Worker Nodes (WN). As User Interface (UI) we recommend the Austrian-Grid LiveCD solution introduced by the University of Innsbruck [14], but other Globus-compatible installations can be used as well.

Rapid deployment of grid resources is encumbered by the complexity of current grid middleware that requires considerable configuration efforts to adapt to different parameters found at each resource location. Our approach aims at reducing the overall deployment effort by eliminating manual configuration of the most frequently deployed grid element, i.e. the worker node.

Our solution allows cloning and deploying large numbers of worker nodes from a master worker node image after assigning this image to its computing element one time. Worker nodes account for the largest part of resources in a production grid. Enabling self-configuration for worker nodes therefore is most efficient.

In contrast to various other components in a grid, such as computing or storage elements, worker nodes can be considered *clients* in nature. Computing elements which act as interface to the grid for worker nodes are more *server* in nature. We therefore restricted the configuration to these elements and relocated worker node configuration there. Figure 1 gives an example of how a small site could



**Figure 1. Integration of a site: The CE image is configured and the master WN image is assigned to the CE. Afterwards the WN is cloned  $n$  times and deployed.**

be integrated into the grid using native virtualization. The example installation provides a single CE and a couple of WNs that make use of an existing infrastructure while at the same time its original purpose continues to be served. Hence, it represents a suitable method for harvesting the resources of a computing lab at a school or university.

### 2.3. Infrastructure Requirements

Even when using virtualization, some requirements regarding the underlying network infrastructure remain. Due to the large amount of possible network and routing configurations on a site, we had to define the general requirements for the system to work.

Requirement	Compute Element	Worker Node
official hostname and grid certificate	required	not required
official IP address	required	not required
unrestricted network access to	internet, worker nodes	computing element

**Table 2. Infrastructure requirements for CE and WN**

These requirements are summarized in table 2. A CE acts as an interface between a computing infrastructure and the grid and must therefore have access both to the grid network, which in most cases is the public internet, as well as to the WNs which make up the computing infrastructure represented by the CE. For WNs, no such requirement exists as they are not directly accessed from the outside but are accessed internally by the CE in order to execute and manage jobs. In addition, grid security requires valid certificates to be present on any host contacted from a grid which in turn can only be issued to host names registered in the public Domain Name System.

## 2.4. Image Set-Up

A VMware *image* consists of a number of files, essentially the configuration file describing the virtual hardware to be provided and a disk image file storing the contents of the virtual hard disks. Several optional files can be provided or are created during runtime. Distributing the configuration file and the disk image however is sufficient to boot a virtual machine.

### Base Image

To create the initial VMware disk image and VMware configuration file we used a software called VMware Utilities [15] which can be obtained free of charge. The resulting configuration file was customized manually in a later phase of the image setup process.

Creating the base system involved the following steps:

- Installation of the operating system
- Installation of components shared by both CE and WN and removal of superfluous packages
- Introducing additional security measures

Later communication between CE and WN required Unix users accounts and groups to be available. An initial keypair for each user was created allowing for password-less logins between the CE and WN in both directions, which is a requirement for the batch system. This image served as the base for the creation of the CE and WN images. An optional software called VMware Tools was installed in the Base Image. It optimizes the virtual machine performance and allows to trigger a shutdown of the virtual machine by shutting down the host machine.

### Computing Element Image

The main components of the CE image are the Globus grid middleware which is used to provide GRAM, GRAM-WS and GIIS interfaces to the grid, the TORQUE batch scheduling system to manage and Ganglia to monitor the worker nodes. The Globus Toolkit installation and configuration was

greatly facilitated by VDT, a compilation and online-repository of widely-used grid middleware from different providers. In addition, the Grid User Management System [16] and the PRIMA callout module [17] were installed on top of Globus to provide a user and virtual organization management system which is compatible with the VOMS system used in the Austrian Grid and other grid projects.

### Worker Node Image

By modifying the boot scripts of the Base Image a WN image was created, which is separately available. The user must supply a single parameter, i.e. the IP address of the CE, to allow the WN to start up. Using this information, all required information can then be retrieved over the network. As VMware does not offer a way to pass parameters from the host to the guest system, the IP address has been encoded into the only parameter which is configurable in the configuration file: the MAC address of virtual network interfaces. Two such MAC addresses have to be used to encode a four-byte IP address because only the least-significant two bytes of a MAC address can be modified. This method has also been discovered by [9]. A more elegant solution of this problem could be achieved by using DHCP or SDP, though requiring the presence of a DHCP or SDP infrastructure on-site and thus reducing the portability of the image.

The base image already includes almost all components needed to operate the WN, additional components merely comprise the client components of Ganglia and TORQUE.

### 2.5. Image Distribution and Startup

The image creation process described in the previous section is a one-time process which has already been accomplished as a result of this work. These prototype images in fact need some refinement to make production use more convenient, however, they could already now be transferred to other sites to integrate more resources into the grid. Such a refinement could include the provision of elaborate configuration scripts for the CE image that eliminate the need for manual configuration, which is currently still necessary.

As soon as the images have been distributed, the start-up merely requires launching the VMware Player software on the host machines and instructing it to load the specified image. This starts the boot process of the image which then takes care of properly launching the resource and integrating it into the grid.

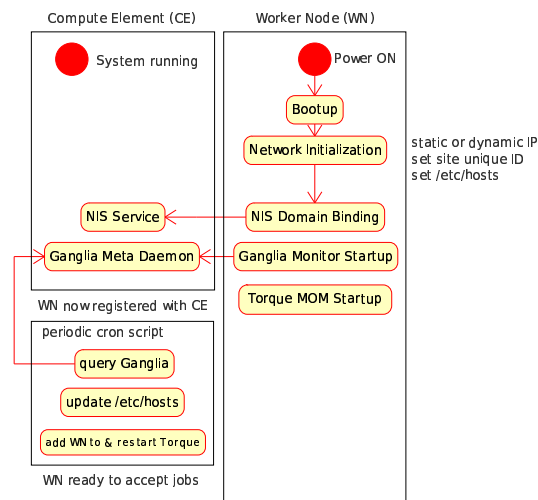
### 2.6. Interaction between CE and WN

The next section describes the process of registering a WN with a CE and executing a job on it. It assumes, that a CE is already running and properly configured. The CE then interacts with the WNs in the following scenarios:

- A WN starts up and is added to the batch system
- The status of a job is modified on one of the WNs (e.g. a job is started or canceled)
- The output of a job is retrieved from the WN by the CE as soon as the job has finished
- The WN exchanges status information with the CE

As the CE is the *master node* of the Grid resource to be represented it must be up and running as first node. The WNs gain the information to which CE they have to register to by using the method described in section 2.4..

## Registration procedure



**Figure 2. Integration of an additional worker node**

The startup sequence of a WN and the procedure for adding it to the pool of available WNs is as follows (see also figure 2):

1. After the boot of the virtual machine, the **network configuration** starts. The WN retrieves its IP address information using either dynamic or static allocation. It sets its *site unique identity* and generates a basic `/etc/hosts` file to allow name resolution of the CE.
2. The **Network Information Service (NIS)** binds itself to the CE such that names of the neighbour WNs can be resolved centrally from now on.
3. The **Ganglia Monitor** starts up and submits status information such as CPU usage and available RAM to the CE. The WN is now registered with the CE.
4. A Cron script on the CE takes care of managing the available WNs by periodically querying the Ganglia status, and in case of any changes to the available WN performing the following:
  - (a) updating the `/etc/hosts` file
  - (b) adding the WN to the batch system configuration
  - (c) restarting the batch system
5. The WN is now ready to accept jobs from the CE.

## Un-registration procedure

A WN can go down deliberately or unexpectedly. In both cases, the Ganglia cluster monitoring system as well as the batch system will detect the absence of that node and stop scheduling jobs to this WN:

1. As soon as a node is unreachable, this is detected by TORQUE and the node is marked as down
2. A periodic clean-up script removes nodes that are down from the batch system configuration

Jobs that are executed while a node goes down are lost and must be resubmitted by the user.

## 2.7. Performance Penalty of Virtualization

The performance of native virtualization in the context of grid computing has been previously researched, however often focusing on the scalability of multiple virtual machines running on one host. As this use case is of little interest to us, we have focused on measuring the performance penalty introduced by virtualization in an out-of-the-box scenario. In essence, our own performance measurements confirm previous research, where a performance penalty of about 5 to 10% on application performance was observed [9][19].

We have chosen freely available microbenchmarks that test single sub-systems of a computer and modified them where necessary. A benchmark run consists of a group of three tests measuring the performance of CPU, disk, and memory. The CPU benchmark stresses the processor by performing a series of operations, mostly floating-point, with a total run time of about 20 seconds. The disk benchmark comprises tests of disk throughput by performing POSIX read and write operations on a file using a block size of 16K. The memory benchmark measures the throughput of memory block read and write operations.

All benchmarks were run on identical hardware but the underlying software stack was varied in each run. The most likely hosting environment of a virtual grid infrastructure as described in this paper will be mostly unmodified computers that must continue to serve their original purpose. To simulate this scenario in our benchmarks, they were run on out-of-the box installations of VMWare Player (version 1.0.2-29634). Hence, the benchmarks in this paper must not be considered as a reference for the maximum performance achievable by a virtualization product. Instead, they are intended to show what can be achieved in a specific scenario.

The compiled benchmark binaries were identical in all runs. First, the reference performance was determined by executing the benchmarks on the system running a native Linux installation. The virtual machine performance was then measured by executing the benchmarks within a virtual machine, once hosted by VMware Player for Windows and once hosted by VMware Player for Linux. Each benchmark was run a total of 14 times and a 20% trimmed mean calculated from the numbers. The results are reproduced in figure 3. The virtual machine performance is given relative to the native performance.

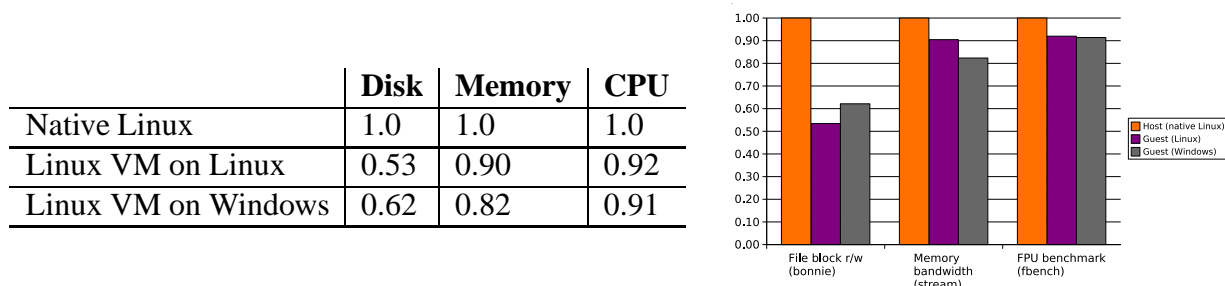


Figure 3. Relative performance of virtual machines compared to native speed

We observe that the performance penalty stays below 10% for applications stressing processor and memory and can therefore be neglected for those applications. A different statement must be made for applications making heavy use of the disk, which can be expected to suffer from a considerable performance degradation.

### **3. Discussion and Future Work**

The following points are suggested for further discussion and possible future application:

- At the moment the assignment of WNs to a specific CE is implemented by encoding the IP address of the CE into the MAC addresses of two unused ethernet interfaces. Further scenarios could enhance the WNs to retrieve the CE IP and the site unique identifier via the DHCP server. This alternative requires access to the local DHCP server, which cannot be taken for granted at all sites. As this alternative would also require modification of the local network infrastructure the deployment would not be minimally invasive any more, hence it could be offered as an option only.
- In a native virtualization environment, the guest system is not aware of the real workload of the host system. Hence, the host system might work to capacity, whereas the guest system reports inactivity. This might lead to wrong scheduling decisions. This must be investigated further.
- The image could be ported to other virtualization solutions. This is mostly attractive for the CE part of the installation, as the application of e.g. Xen on server systems already rises. This would further ease the gridification.
- WNs could be configured to start in “independent persistent mode”. In this mode during normal operation the WNs harddisk behaves like a normal writable drive, but after a reboot of the WN it is booted from the original state. Hence the WN image is not alterable easing administration and offering additional security.
- We currently use TORQUE as batch management system. TORQUE however is not specifically designed for cycle harvesting environments. An evaluation and possible implementation of Condor [20] is currently under investigation, as Condor offers better handling of fluctuating host pools (e. g. by offering possibilities to checkpoint running jobs and start them on another node if a node gets unavailable).
- Within most institutions, deploying such a scenario would require the permission of the local IT department and possibly network administration which typically tend to be rather conservative regarding such installations.
- The benchmark evaluation could be extended by adding benchmarks of the virtual network sub-system as well as a study of real-world application performance..

### **4. Conclusion**

As a result of this work, a fully functional and homogeneous grid resource consisting of multiple virtual machines executed on heterogeneous host operating systems (Linux and Windows) was created and included into the Austrian Grid. Test jobs were successfully submitted to and executed on this

resource. Our conclusion therefore is that it is possible to rapidly harness existing heterogeneous low-load resources for grid computing using virtualization methods, provided that a suitable network infrastructure exists and administrative barriers can be overcome.

## References

- [1] The Austrian Grid Initiative, <http://www.austriangrid.at>, Last visited Dec. 21<sup>st</sup> 2006
- [2] PearPC - PowerPC Architecture Emulator, <http://pearpc.sourceforge.net>, Last visited Dec. 21<sup>st</sup> 2006
- [3] VMware Virtualization Software, <http://www.vmware.com>, Last visited Dec. 21<sup>st</sup> 2006
- [4] P. Barham, B. Dragovic et al, "Xen and the Art of Virtualization", *Proceedings of the ACM Symposium on Operating Systems Principles*, 2003
- [5] Linux VServer, <http://linux-vserver.org>, Last visited Dec. 21<sup>st</sup> 2006
- [6] G. Andronico, R. Barbera et al, "GILDA: The Grid INFN Virtual Laboratory for Dissemination Activities", *First International Conference on Testbeds and Research Infrastructures for the Development of Networks and Communities*, p304-305, 2005
- [7] Lightweight Middleware for Grid Computing (gLite), <http://glite.web.cern.ch/glite>, Last visited Dec. 21<sup>st</sup> 2006
- [8] The Grid INFN Laboratory for Dissemination Activities (GILDA), Virtual Services, <https://gilda.ct.infn.it/VirtualServices.html>, Last visited Dec. 21<sup>st</sup> 2006
- [9] K. Keahey, K. Doering and I. Foster, "From Sandbox to Playground: Dynamic Virtual Environments in the Grid", *GRID '04: Proceedings of the Fifth IEEE/ACM International Workshop on Grid Computing (GRID'04)*, p34-42, 2004
- [10] Parallels, <http://www.parallels.com>, Last visited Dec. 21<sup>st</sup> 2006
- [11] QEMU open source processor emulator, <http://www.qemu.com>, Last visited Dec. 21<sup>st</sup> 2006
- [12] Virtualization Overview, <http://www.vmware.com/pdf/virtualization.pdf>, Last visited Dec. 21<sup>st</sup> 2006
- [13] Markus Baumgartner and Peter Praxmarer. "Austrian Grid Software Specification". Downloadable at <http://www.austriangrid.at/austriangrid/internal>, May 2006. Version 0.6.
- [14] Peter Brunner, "Austrian Grid LiveCD", <http://www.dps.uibk.ac.at/~csac8265/agridlive.pdf>,
- [15] VMware Utilities Homepage, Robert D. Petruska, <http://petruska.stardock.net/Software/Vmware.html>, Last visited Dec. 21<sup>st</sup> 2006
- [16] G. Carcassi et al., "A Scalable Grid User Management System for Large Virtual Organizations", *Proceedings of Computing for High Energy Physics (CHEP2004), Interlaken, Switzerland*, 2004.

- [17] M. Lorch, D. B. Adams, D. Kafura, M. S. R. Koeneni, A. Rathi, S. Shah, "The PRIMA System for Privilege Management, Authorization and Enforcement in Grid Environments", *Fourth International Workshop on Grid Computing*, 2003.
- [18] B. Quetier, V. Neri, F. Cappello, "Selecting A Virtualization System For Grid/P2P Large Scale Emulation", *Proc of the Workshop on Experimental Grid testbeds for the assessment of large-scale distributed applications and tools (EXPGRID'06), Paris, France, 19-23 June, 2006*
- [19] R. J. Figueirido, P. A. Dinda, José A. B. Fortes, "A Case For Grid Computing On Virtual Machines", *The 23rd International Conference on Distributed Computing Systems (ICDCS), Rhode Island, USA, 2003*.
- [20] T. Tannenbaum, D. Wright, K. Miller, M. Livny, "Condor - A Distributed Job Scheduler", *Thomas Sterling (ed.): Beowulf Cluster Computing with Linux, MIT Press, 2002*.