Management of Virtual Machines on Globus Grids Using GridWay

A.J. Rubio-Montero[1], E. Huedo[2], R.S. Montero[2] and I.M. Llorente[2]

[1]Centro de Investigaciones Energéticas, Medioambientales y Tecnológicas. Avda. Complutense 22, 28040 Madrid, Spain. antonio.rubio@ciemat.es

[2]Facultad de Informática Universidad Complutense de Madrid 28040 Madrid, Spain ehuedo@fdi.ucm.es {rubensm,llorente}@dacya.ucm.es

Abstract

Virtual machines are a promising technology to overcome some of the problems found in current Grid infrastructures, like heterogeneity, performance partitioning or application isolation. In this work, we present an straightforward deployment of virtual machines in Globus Grids. This solution is based on standard services and does not require additional middleware to be installed. Also, we assess the suitability of this deployment in the execution of a high throughput scientific application, the XMM-Newton Scientific Analysis System.

1 Introduction

Since the late 1990s, we have witnessed an extraordinary development of Grid technologies. Nowadays, different Grids are being deployed within the context of a growing number of national and transnational research projects (e.g. EGEE[1], TeraGrid[2] or OSG[3]). These projects have achieved unseen levels of resource sharing, offering a dramatic increase in the number of processing and storage resources that can be delivered to applications.

However, a growing heterogeneity on the organizations that joins these projects hinders the development of large scale Grid infrastructures. Grid resources do not only differ in their hardware but also in their software configurations (operating systems, libraries, and applications). This heterogeneity increases the cost and length of the application development cycle, as they have to be tested in a great variety of environments where the developers have limited configuration capabilities. Therefore, some of the users are only able to use a small fraction of the Grid. Other projects, like EGEE, limit heterogeneity somewhat by imposing a fixed configuration for Grid resources.

Moreover, most of the Grid infrastructures do not allow administrators to isolate and partition the performance of the resources they devote to the Grid. In this way, the applications of a grid user can affect the execution of other grid or local users. This limits the quality of service and reliability of actual platforms, preventing a wide adoption of the Grid paradigm.

Among other solutions, the use of virtual machines in computational Grids are being explored to solve the previous problems. Virtual machines add a new abstraction layer that allows partitioning and isolating the physical hardware resources [6]. They also offer a natural way to face a highly heterogeneous environment. In this context, a step forward was made by K. Keathy et al. [13]: the virtual workspaces, which provides an abstraction of a whole execution environment. Virtual workspaces can be implemented using virtual machines, and can be grouped in complex virtual architectures like clusters [16].

In this work, we present an straightforward deployment of virtual machines in a Globus Grid, which overcomes some of the problems of current virtualization solutions, that are described in Section 2. In particular, we will consider the execution of a high throughput scientific application: the XMM-Newton Scientific Analysis System. The target application and the benefits of its execution within a virtual machine are briefly described in Section 3. The main characteristics of the proposed deployment strategy are then presented in Section 4. Following this, in Section 5 we discuss some implementations details on a experimental testbed. Then, we give some performance results in Section 6. The paper ends with some conclusions and hints of future work in Section 7.

[1]www.eu-egee.org
[2]www.teragrid.org
[3]www.opensciencegrid.org

1-4244-0910-1/07/$20.00 ©2007 IEEE.

2 Related Work

In the last years, the processors' performance has increased enough to renew the interest in the virtual machines technology. These technologies include operating system partitioning (e.g. Solaris Containers [1]), complete hardware emulation (e.g. VMWare [2]), or para-virtualization (e.g. Xen [5]). Virtual machines presents attractive benefits like server consolidation, virtual machines isolation, performance partitioning or legacy applications execution among others [6].

Recent works have also made an important effort to integrate these virtualization technologies in Grid infrastructures. However, not all of these projects embrace the Grid philosophy. This philosophy, proposed by Foster [7], defines a Grid as a system (i) not subject to a centralized control, (ii) based on standard, open and general-purpose interfaces and protocols; that (iii) provides some level of quality of service. This third requirement is largely satisfied by these projects.

The integration of virtual machines in Grid environments was initially explored by the In-VIGO project [4]. The In-VIGO infrastructure adds some virtualization layers to the classical Grid model. The first layer builds sets of virtual Grid resources upon which *standard* grid middleware (e.g. Globus) can be deployed.

The XenoServers [9] aims to develop a network of globally distributed servers to deploy distributed services in exchange for money.

Other approach, more in line with the Grid philosophy, is the Virtual Workspace Project [13], which provides an abstraction of an execution environment that can be dynamically deployed. However, it requires a thorough knowledge of the remote cluster architecture and the access to the deployed workspace is not straightforward, as is not integrated with other Globus services like GRAM. We would like to note that these problems are currently being tackled, see for example [8].

3 XMM-Newton Science Analysis Software (SAS)

XMM-Newton is the most sensitive X-ray satellite ever built and the largest satellite ever launched by ESA. It has been operating as an open observatory since the beginning of 2000, providing X-ray scientific data through three imaging cameras and two spectrometers, as well as visible and UV images through an optical telescope. The large amount of data collected by XMM-Newton is due to its unprecedented effective area in the X-ray domain in combination with the simultaneous operation of all its instruments. All the data taken by this satellite are kept in the XMM-Newton Science Archive (XSA).

The large amount of data available makes necessary to optimize the management of hardware resources to prevent a data processing slow down. In this context, Grid technology offers the capability of managing not only the user queries retrieving data from the archive, but also the online processing of that data.

The Scientific Analysis System (SAS) [3] is a software suite for the interactive analysis of all the XMM-Newton data, making possible the tailoring of the data reduction to the scientific goal. In addition it makes possible a re-calibration of the data whenever new calibration files have been released. Although SAS is considered a fully mature package, a large number of people are still working to improve it, and periodically new versions of SAS are released.

The execution of the SAS software in a Grid has been successfully studied in [11]. However, the deployment of the new versions of the SAS software in the ESA sites is not trivial, and requires an important effort from the system administrators. This problem can be easily solved by distributing virtual images with the latest release of the software. Note also that this will improve the robustness of the software as is always executed and developed in the same environment.

4 Execution Management of Virtual Workspaces

In this section we describe the deployment and execution management of single virtual machines in a Grid infrastructure. The approach presented in this work consists in encapsulating a virtual workspace in a grid job. It also incorporates the functionality offered by a general purpose meta-scheduling system. So, the genuine characteristics of a Grid infrastructure (i.e. dynamism, high fault rate, heterogeneity) are naturally considered in the proposed solution.

This strategy does not require additional middleware to be deployed, as it is based on well-tested procedures and standard services. Moreover, it is not tied to a given virtualization technology. However, it presents some drawbacks:

- The underlying local resource management system is not aware of the nature of the job itself. Therefore, some of the potential benefits offered by the virtualization technology (e.g. server consolidation) are not fully exploited.

- This approach is only suitable for single process batch jobs, which naturally arise in High Throughput Computing problems, or workflow computations. However

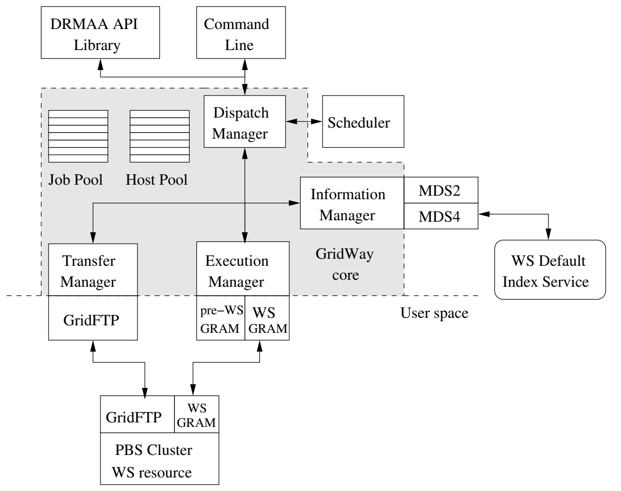

Figure 1. GridWay architecture, and interaction with Grid Services

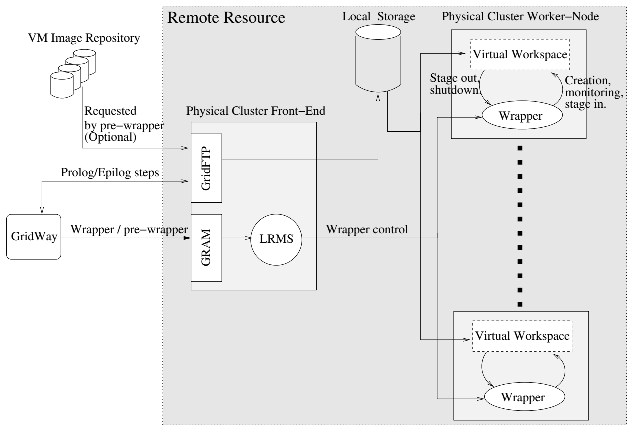

Figure 2. Schematic representation of SAS execution within a virtual machine, in a Grid environment

this use case represents a small fraction of the applications that can potentially benefit from the virtualization technology in a Grid.

4.1 Virtual Workspace Scheduling

The deployment of the virtual workspaces is managed by the GridWay[4] meta-scheduling system [10]. GridWay allows unattended, reliable, and efficient execution of jobs on heterogeneous and dynamic Grids. It performs, transparently to the end user, all the job scheduling and submission steps [15], namely: resource discovery and selection, and job preparation, submission, monitoring, migration and termination.

Job execution is performed in three steps: *prolog*, for creating the remote experiment directory and transferring the executable and input files; *wrapper* for executing the actual job and obtaining its exit code; and *epilog* for transferring back output files and cleaning tasks. The *prolog* and *epilog* phases are done by interacting with the Grid file transfer services (GridFTP in this work), while the *wrapper* step interfaces the Grid execution services (WS-GRAM in our case). The architecture of GridWay and the interaction with the Grid services of the target infrastructure are shown in Figure 1.

The description of the grid job must include all the requirements of the virtual machine within. So, the job template used by GridWay must specify the Xen Hypervisor version (that must be compatible with the Linux kernel needed by the the virtual machines), and other hardware requirements (e.g. free memory or CPU load).

[4]www.gridway.org

GridWay schedules a grid job as follows: the *Information Manager* (see Figure 1) holds a list with the available resources in the Grid, and their characteristics. This list is periodically updated by querying the information services to monitor and discover grid hosts. The *Dispatch Manager* filters out those resources that do not have free slots, or do not meet the job requirements. Then it sorts the remaining hosts according to an user-supplied rank expression. The highest ranked resource is used to dispatch the job.

4.2 Deployment of the Virtual Workspace

In the following experiments, the virtual machine images are already available at the remote resource. In general, the images could be downloaded, as any other input file, in the *prolog* state, from the client or from an image repository, using GridFTP or the Reliable File Transfer (RFT) service.

Additionally, GridWay allows the definition of an optional *pre-wrapper* phase to perform advanced job configuration routines. This phase consists in an user defined program that is executed on the cluster front-end. In our case, this program may check the availability of a given image, and transfers it from a GridFTP repository if needed. Note also that higher level data services, like the Replica Location Service (RLS), could be used.

Then, the *Execution Manager* interfaces the WS-GRAM service to submit the *wrapper* program, which performs the following actions:

- It checks the availability of the requested virtual machine image in the cluster node.
- The virtual machine is started or restored with an unique identification and MAC address. Then, the

Table 1. Summary of characteristics of the testbed resources.

Host	CPU	Memory	OS	Service Configuration
ursa	PIV HT 3.2GHz	512MB	Fedora Core 4	GT4, GridWay
draco	PIV HT 3.2GHz	512MB	Debian Etch	GT4, PBS, NIS, NFS, DHCP
draco WN	PIV HT 3.2GHz	2GB	Debian Etch	Xen3.0 testing

wrapper waits for the virtual machine activation by periodically probing its services.

- The *wrapper* copies all the input files needed by the experiment to the virtual machine, and it executes the XMM-Newton analysis program.
- The output files are transferred back to the physical cluster file system to be copied to the client host in the *epilog* phase.
- Finally it shuts down (or suspends to disk) the virtual machine.

The previous process is depicted in Figure 2.

5 Experimental Infrastructure

5.1 Testbed Description

The behavior of the previous deployment strategy will be analyzed on a research testbed based on the Globus Toolkit 4.0.1 and GridWay 4.7. The testbed consists of two resources: a client host, and a PBS cluster for processing purposes. The main characteristics of these machines are described in Table 1. These hosts are connected by a Fast Ethernet campus network.

The client host is *ursa*, which runs an instance of the GridWay meta-scheduler and holds the input dataset ($\sim$22 MB) with the observation data and the current calibration file to be analyzed. It also receives the analysis output files ($\sim$13 MB) with the observation events to perform post-processing tasks. The SAS tasks are scheduled to the *draco* cluster, with two Xen-capable worker nodes (WN). Note that no virtual machines are started in the cluster front-end.

5.2 Implementation Details

In the following experiments, the OS images are stored in the front-end and are accessed in the cluster nodes via NFS. Although this configuration could impose significant overheads, they have not been experienced because of the small size of the cluster used in this work.

The disk images have been obtained from a typical cluster worker-node installation with the Fedora Core 4 operating system and the XMM-Newton SAS software. The installation is then tailored by deactivating all the services not needed by the SAS software, e.g. PBS-mon. In addition, the virtual machine includes a RSH server for executing the remote commands requested by the *wrapper* program. Finally, this disk image is split in three different files to save storage space and ease its deployment (see Table 2):

- The root file system, with read-write permissions to modify the virtual machine configuration files at boot time. Two copies of this disk image are available to be simultaneously used by the two cluster worker-nodes.
- The `usr` file system, with the standard Linux applications and libraries. This disk image is read-only and shared by all the cluster nodes.
- The `opt` file system, with the XMM-Newton SAS installation. This disk image is also read-only and shared by all the cluster nodes.

Additionally, a local disk image has been created in each worker-node. The virtual machine mounts this image in the `scratch` directory, where the SAS program stores temporal files and data. So, the input/output operations performed by analysis software are always made in the local hard disk.

Table 2. Disk layout of the Virtual Machines.

Mount point	Size	Contents
`/`	500MB	Fedora Core 4 base system
`/usr`	650MB	Standard Linux applications
`/opt`	600MB	SAS 6.5.0
`/scratch`	2GB	Ext3 disk image

The virtual machines are configured with 512MB of main memory and one virtual CPU. The virtual machines network is configured with a DHCP server, which dynamically assigns IP addresses from a private network, different from the physical cluster network.

6 Results

In this section we study the performance obtained with several configurations of the virtual workspace, and the overhead induced by the virtualization in each case. Also we introduce application level metrics to analyze the results from the user point of view. Each experiment consists in performing 100 times the analysis of the same observation from the XMM-Newton satellite. Also, caching of the observation data is avoided.

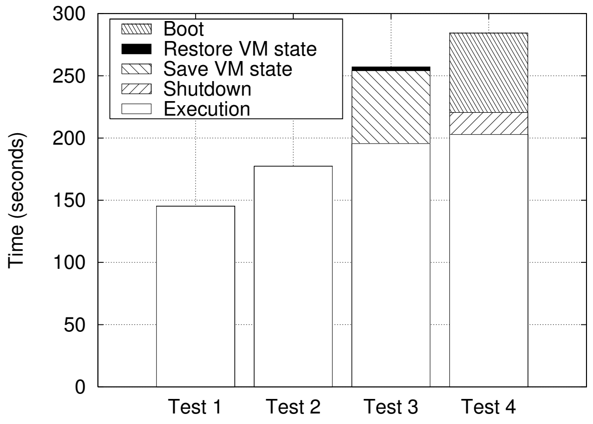

Figure 3. Average operation times obtained in the *wrapper* phase: without virtualization (Test1), with persistent virtual machines (Test2), pausing/restoring (Test3) and stopping/starting the virtual machine (Test4)

6.1 Execution without Virtualization (Test1)

The first experiment submits 100 jobs through the GridWay meta-scheduler with the standard *wrapper* program, i.e. without virtualization. To this end, the worker nodes have been also installed with a Fedora Core 4 operating system, and with the same configuration as the virtual images described in Section 5. In this way, we obtain a performance upper-bound, that will be used to study the virtualization alternatives below. The average execution time of a SAS job with this configuration is 148 seconds.

6.2 Persistent Virtual Machines (Test2)

In some situations, when a virtual workspace is frequently used, it could be more efficient to kept it active to reduce boot and shut-down overheads. This situation may naturally arise for high throughput computing applications like the one considered in this work. This experiment measures the inherent cost of virtualization, which is roughly 20% increment in execution time, as can be seen in Figure 3.

6.3 Saving and Restoring the Virtual Machine State (Test3)

The state of a virtual machine includes, in addition to disk images and Xen configuration files, a representation of its main memory, and the CPU registers of all its virtual processors. When the context of a virtual machine is saved, it can be later restored, keeping its configuration and resuming the execution of its processes.

This feature, which is efficiently implemented by Xen, can be used to keep in the virtual machine memory the system services, the SAS program and related shared libraries, without keeping it active and so saving system resources. In this case, when the execution of the SAS task finishes, the *wrapper* program saves the context of the virtual machine, which is restored before executing another SAS task in that worker-node.

As expected, the execution time is similar to that obtained in the previous experiment (Test2), see Figure 3. The additional overhead is mainly due to saving the state of the virtual machine, and implies an overall increment of 77% in execution time compared to Test1.

6.4 Stopping and Starting the Virtual Machines (Test4)

Stopping the virtual machine after the execution of each SAS task (and starting a new one before executing it) allows an straightforward deployment of the virtual workspaces. However, it adds an additional overhead to the boot/shutdown process of the virtual machines. In particular, the average execution time is roughly twice that obtained in Test1 (see Figure 3).

The problem size of the astronomical observations used in the above experiments has been deliberately chosen small to ease the measurement process. However, in general it will be considerably larger, increasing the total execution time to several hours. As the boot/shutdown and save/restore times are independent of the problem sizes, the additional overhead of Test4 and Test3 will be negligible. Therefore, the overall virtualization overhead will tend to that obtained in Test2.

It is interesting to analyze the system behavior from the application point of view. The performance obtained in the execution of a high throughput computing application, can be studied using the system throughput $P(n)$, defined as usual:

$$P(n) = \frac{n}{T} \qquad (1)$$

where T is the execution time of n SAS tasks. Note that the execution time includes file transfer times (29 seconds on average) and Globus overhead.

Figure 4 shows the system throughput in the execution of the previous tests. The overhead induced by the virtualization can be clearly seen in the difference in the asymptotic throughput. In this case, the overall system performance is reduced 13% (from 0.011 jobs/second in Test1 to 0.0095 jobs/second in Test2). Also, it is interesting to note that in all cases the system needs the same number of tasks to achieve half of the peak performance (see [14]).

7 Conclusions and Future Work

In this work we have presented a straightforward deployment of virtual machines in a Grid infrastructure. This strategy does not require additional middleware to be installed and it is not bounded to a virtualization technology. The overall overhead induced by the virtualization technology decreases the application performance by 13%. However, it presents attractive benefits, like increasing software robustness or saving administration efforts, so improving the *quality of life* in the Grid [12].

This solution presents several drawbacks like a limited use of the potential benefits offered by the virtualization technology (e.g. server consolidation), or a limited application range (HTC and workflow computations).

The results presented here should be seen as a first step to incorporate virtual machines in a production environment. We intend to solve the above problems by integrating the current solution with the Virtual Workspaces Service.

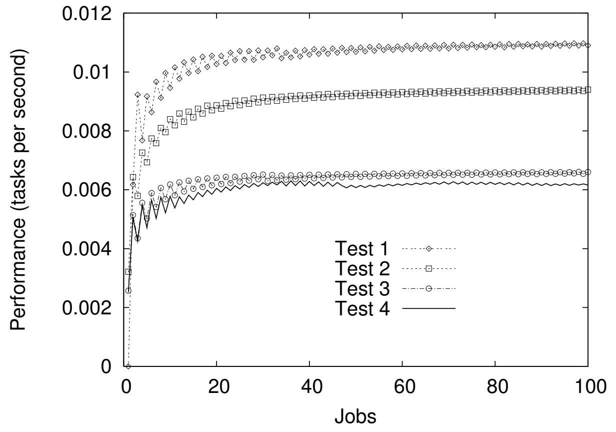

Figure 4. Testbed throughput (jobs per second) without virtualization (Test1), with persistent virtual machines (Test2), saving/restoring (Test3) and stopping/starting the virtual machine (Test4)

Acknowledgment

The authors would like to thank the European Space Astronomy Center (ESAC/ESA) for providing access and support with the SAS software. We would also like to thank in particular A. Ibarra and D. Tapiador for their valuable explanations on the SAS software and its execution in the Grid; and F. Félix and A. de la Fuente for their constructive comments about the use of virtual machines with SAS.

This research was supported by Consejería de Educación de la Comunidad de Madrid, Fondo Europeo de Desarrollo Regional (FEDER) and Fondo Social Europeo (FSE), through BIOGRIDNET Research Program S-0505/TIC/000101, and by Ministerio de Educación y Ciencia, through the research grant TIN2006-02806.

References

[1] Solaris Containers. Sun Microsystems, Inc. Available for Solaris 10 OS. http://www.sun.com/software/solaris/.

[2] VMware. VMware,Inc. Available at http://www.vmware.com.

[3] XMM-Newton Science Analysis Software. European Space Astronomy Center. Available at http://xmm.vilspa.esa.es/sas/.

[4] S. Adabala, V. Chadha, P. Chawla, R. Figueiredo, J. Fortes, I. Krsul, A. Matsunaga, M. Tsugawa, J. Zhang, M. Zhao, L. Zhu, and X. Zhu. From Virtualized Resources to Virtual Computing Grids: The In-VIGO system. *Future Generation Computer Systems*, 21(6), April 2005.

[5] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield. Xen and the Art of Virtualization. *ACM Symposium on Operating Systems Principles (SOSP)*, October 2003.

[6] R. Figueiredo, P. Dinda, and J. Fortes. A Case For Grid Computing On Virtual Machines. *23rd International Conference on Distributed Computing Systems , IEEE*, 2003.

[7] I. Foster. What Is the Grid? A Three Point Checklist. *GRIDtoday*, 1(6), 2002. Available at http://www.gridtoday.com/.

[8] I. Foster, T. Freeman, K. Keahey, D. Scheftner, B. Sotomayor, and X. Zhang. Virtual Clusters for Grid Communities. *CCGRID 2006, Singapore*, May 2006.

[9] S. Hand, T. Harris, E. Kotsovinos, and I. Pratt. Controlling the XenoServer Open Platform. In *Proceedings of the Sixth IEEE Conference on Open Architectures and Network Programming (OPENARCH 2003)*, April 2003.

[10] E. Huedo, R. S. Montero, and I. M. Llorente. A Framework for Adaptive Execution on Grids. *Intl. J. Software – Practice and Experience (SPE)*, 34(7):631–651, 2004.

[11] A. Ibarra, D. Tapiador, F. Felix, C. Gabriel, C. Arviset, J. Hoar, and S. Ansari. Optimizing SAS tasks for Grid Architectures. *Astronomical Data Analysis Software and Systems XV, ASP Conference Series*, 351:520, 2006.

[12] K. Keahey, I. Foster, T. Freeman, and X. Zhang. Virtual Workspaces: Achieving Quality of Service and Quality of Life in the Grid. *Scientific Programming Journal*, 13(4):265–276, 2005.

[13] K. Keahey, I. Foster, T. Freeman, X. Zhang, and D. Galron. Virtual Workspaces in the Grid. *Europar 2005, Lisbon, Portugal,*, September 2005.

[14] R. S. Montero, E. Huedo, and I. M. Llorente. Benchmarking of High Throughput Computing Applications on Grids. *Parallel Computing*, 32(4):267–269, 2006.

[15] J. M. Schopf. Ten Actions when Superscheduling. Technical Report GFD-I.4, Scheduling Working Group – The Global Grid Forum, 2001.

[16] X. Zhang, K. Keahey, I. Foster, and T. Freeman. Virtual Cluster Workspaces for Grid Applications. Technical report, April 2005. ANL/MCS-P1246-0405.