Dynamic Provisioning of Virtual Clusters for Grid Computing*

Manuel Rodríguez[1], Daniel Tapiador[2], Javier Fontán[1], Eduardo Huedo[1], Rubén S. Montero[1], and Ignacio M. Llorente[1]

[1] Departamento de Arquitectura de Computadores y Automática
Facultad de Informática
Universidad Complutense de Madrid, Spain
{manuelro,jfontan}@fdi.ucm.es, {ehuedo,rubensm,llorente}@dacya.ucm.es
[2] European Space agency ESA
Villanueva de la Cañada, Spain
Daniel.Tapiador@sciops.esa.int

Abstract. Virtual machines can greatly simplify grid computing by providing an isolated, well-known environment, while increasing security. Also, they can be used as the base technology to dynamically modify the computing elements of a grid, so providing an adaptive environment. In this paper we present a Grid architecture that allows to dynamically adapt the underlying hardware infrastructure to changing Virtual Organization (VO) demands. The backend of the system is able to provide on-demand virtual worker nodes to existing clusters and integrate them in any Globus-based Grid. In this way, we establish a basis to deploy self-adaptive Grids, which can support different VOs in shared physical infrastructures and dynamically adapt its software configuration. Experimental results on a prototyped testbed show less than a 10% overall performance loss including the hypervisor overhead.

1 Introduction

Recently, interest in virtual machines is quickly growing, as hardware support provided by new generation microprocessors significantly reduces the overhead of virtualization [1]. Typically, hypervisors like Xen [2] or VMWare [3] take advantage of these new hardware features to improve their performace. Computational Grids can greatly benefit from virtualization. A Grid is a highly heterogeneous system both in terms of the hardware and software configuration of its components. This fact reduces the number of potential resources to run a given

* 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, by Ministerio de Educación y Ciencia, and through the research grant TIN2006-02806, and by the European Union through the research project RESERVOIR Contract Number 215605.

E. César et al. (Eds.): Euro-Par 2008 Workshops, LNCS 5415, pp. 23–32, 2009.

application, which usually requires specific versions of different software components (e.g. operating system, libraries or post-processing utilities). Moreover, the installation, configuration and maintanace of these different components significantly increases the operational costs of the Grid infrastructure. Finally, those organizations that contribute resources to a Grid usually want to limit the interaction of Grid applications with their own *internal* workload.

Among other solutions, Virtual Machines (VM) can solve the aforementioned problems. From the user perspective, VMs ensure the correct execution of the application by encapsulating software configurations in a "well-known" environment. On the system administrator side, VMs are an efficient technology to isolate and partition the system. Thus, allowing them to set the amount of resources devoted to Grid jobs. Also, the operational cost of the infrastructure is reduced as specific appliances to run an application class can be prepared, configured and deployed.

The integration of virtual machines in Grid environments has been previously explored by several works. For example, the In-VIGO project [4] establishes a basic layer of virtual Grid resources upon which any grid middleware can be deployed. The Virtual Workspace Service [5], exposes the functionality needed to manage workspaces –abstraction of execution environments implemented through VMs. Also, a straightforward deployment of virtual machines to execute scientific codes in a Grid has been analyzed in [6] (see Section 2 for an additional description of other related works).

In this work we propose a novel architecture for the dynamic provisioning of computational services on a Grid infrastructure. The system leverages virtualization technologies to provide flexible support for different Virtual Organizations (VO). Usually, the resources of a Grid site support different VOs (e.g. Bioinformatics or High Energy Physics). The proposed system is able to balance the amount of resources allocated to each VO in terms of their dynamic requests.

On the other hand, different VOs need different software, or even different versions of the same software. Traditionally, the cost of the installation, configuration and maintainace of VO-specific worker nodes have limited the flexibility of the infrastructure. In our case, the system will deploy on-the-fly VO-specific worker nodes to execute their applications.

To achieve these two goals, we propose a multi-layer architecture to provide virtual worker nodes to clusters inside a Grid. The infrastructure, given a set of virtual machine images, is capable of deciding the number and the kind of worker nodes to be created on each cluster. This way, the computing elements of a grid can be adapted to fit the changing software requirements and computational demands. Section 3, provides a detailed description of the system.

The paper also analyzes, in Section 4, a prototype implementation of the architecture. In particular, we discuss the overhead and interaction of all the components of a classical middleware stack, namely: local resource management systems (Sun Grid Engine in our case), Grid resource services (Globus GRAM), information services (Globus MDS4) and meta-schedulers (GridWay). Finally, in Section 5, we discuss our experience and explain our future work.

2 Related Work

The idea of a virtual cluster which dynamically adapts its size to the workload is not new. Jeffrey Chase et al., from Duke University, describe in [7] a cluster management software called COD (Cluster On Demand), which dynamically allocates servers from a common pool to multiple virtual clusters. Although the goal is similar, the approach is completely different. COD worker nodes employ NFS to mount different software configurations. In our case, the solution is based on VM, and the system is studied from a Grid perspective.

Ananth I. Sundararaj et al., from Northwestern University, have also worked on dynamic cluster configuration [8] with a different approach. They have developed a network tool that connects virtual machines, making the connectivity problem identical to that faced by the user when connecting any new machine to his own network. This allows to migrate VMs across different domains while preserving the same IP. Our aim is not to develop a totally virtualized architecture, but to include the advantages of virtualization in existing infrastructures in a non-intrusive way.

OSG (Open Science Group) defines Edge Services, that mediate access between a site and the external world. They can handle common grid operations like job submissions or data movement. I.Foster et al. [9], employ VMs to bring them up dynamically only as they are needed. Although this work is also based on virtualization technology and the Virtual Workspace Service (VWS) the approach is different. In this paper, our goal is not to create a new service (Edge Service) but to improve and adapt existing ones, supporting different VOs in a shared physical infrastructures.

Finally, Amazon Elastic Computing Cloud [10] provides a remote VM execution environment. It allows to execute one or more "Amazon Machine Images" on their systems, providing a simple web service interface to manage them. Users are billed for the computing time, memory and bandwidth consumed. This service greatly complements our development, offering the possibility of potentially unlimited computing resources. It would be possible to employ a grid-enabled Amazon Machine Image, and create as many instances as needed, getting on-demand resources in case the physical hardware cannot satisfy a peak demand.

3 Description of the Architecture

In this section we detail the architecture of the system and its basic behavior. In its design, we have focused in avoiding dedicated systems. The virtual worker nodes provided by the underlying physical infrastructure can register in existing clusters queues (computing services). So the flexibility of the Grid infrastructure can be boosted up without requiring dedicated hardware, or modifying neither existing applications nor software configurations. The final goals of the proposed architecture are:

– Dynamically adapt a shared infrastructure to support different VOs, by balancing the physical resources allocated to each VO.

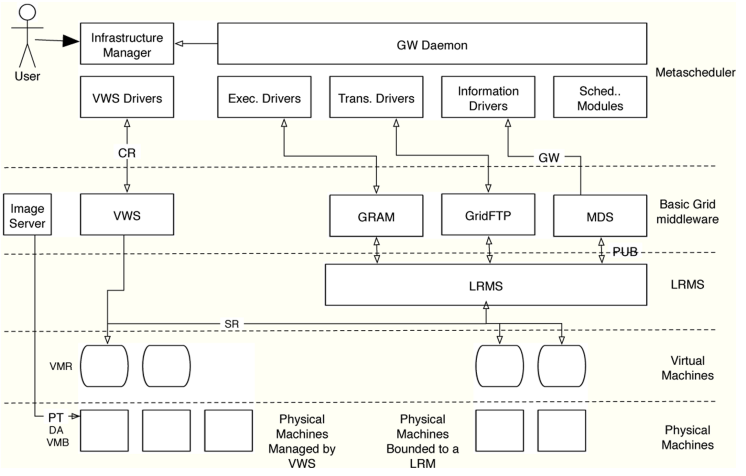

Fig. 1. Architecture Overview. Arrows represent an intertaction between components, the time of this interaction is labeled in the figure.

- Reduce the operational cost of the Grid infrastructure, by providing a simple way to provide on-demand software configurations to VO users.
- Minimize the *Gridification* time, by executing VO applications in a well-known pre-defined environment.

The base of the architecture (depicted in Figure 1) is the Virtual Machine Layer. This layer is responsible for the creation of virtual worker nodes to be registered in a cluster. Typically, the functionality of this layer will be provided by a hypervisor. In the present work we will consider Xen 3.

The worker nodes (either physical or virtual) are managed by a Local Resource Management System (LRMS layer in Figure 1). In our case we will use a Sun Grid Engine (SGE [11]) instance. The SGE cluster is configured with a different queue for each VO, which also provides a VO appliance. So, when a new worker node is to be deployed the corresponding appliance is used. This way, several software configurations can coexist on a single cluster and a job can be executed in the correct one by just specifying the queue name. We would like to note that having a queue for each VO is a usual configration for Grid resources (for example in the EGEE [12] infrastructure).

The previous resources are exposed as Grid services by the components of the third layer, Basic Grid Middleware. The prototype version considered in this paper uses the services provided by the Globus Toolkit 4, namely: information (MDS4), execution (GRAM4) and file transfer services (GridFTP); along with a virtualization interface (Virtual Workspace Service, VWS). The ability of VWS to deploy several copies of a single image along a physical cluster, with a pre-set pool of host names and IPs, keeps integration with LRMS layer simple.

At the top layer (Management) we usually find meta-schedulers that manage, control and monitor the execution of Grid applications. We have chosen the

GridWay [13] meta-scheduler. As, it provides some features which are required by the proposed architecture:

- Scheduling capabilities: GridWay employs a dynamic scheduling system. It can detect when a new machine has been added or removed from a cluster, and redistribute the work load.
- Fault detection & recovery capabilities. Transparently to the end user, GridWay is able to detect and recover from any of the Grid elements failure, outage or saturation conditions [14].

3.1 Infrastructure Manager

Finally, the Infrastructure Manager module completes the system architecture. This component is responsible for adapting the Grid computational services to the dynamic Grid computing demands. The Infrastructure Manager decides when to add new worker nodes (and their type) to a given computing element (cluster queue). So, allowing Grid administrators to adapt their services according to a pre-defined set of policies (e.g. the cluster should be shared in a 2 to 5 ratio between the fusion and bioinformatics VOs.)

The following sequence of actions (see Figure 1) describes the actions that takes place when the Information Manager decides to add a new worker node to the computing element:

1. The Infrastructure Manager request a new VM to the VWS (using a predefined appliance for the VO). The VWS determines the best node to run the virtual machine, based on the resources requested (e.g memory). The arrow labeled "CR" in Figure 1 represents the time since the Infrastructure Manager sends a deploying petition until it receives a confirmation.
2. If the VM image (appliance) is not local to the host system, it accesses the image server via a suitable protocol (e.g. GridFTP) and obtains a copy. We will refer to the time employed on the image transmission as "PT".
3. Once the image has been transferred, the physical node's DHCP server configuration file is altered in order to establish VM's IP and hostname. The time to assign a hostname and IP will be referred as "DA".
4. When these operations conclude, the VM is booted. "VMB" denotes the time since the hypervisor (Xen) receives the execution instruction until it starts booting the VM. Also, "VMR" is the time need to actually boot the system. Note that while VMB is constant, VMR highly depends on the virtual machine configuration and services.
5. When the VM has been deployed and is running, it registers on LRMS frontend (arrow "SR" in Figure 1) as an execution host.
6. After a given time, the Grid information system (MDS) detects the new node and publishes it. This step is labeled "PUB" in Figure 1.
7. Finally, the meta-scheduler (GridWay) will refresh the information of the available Grid resources, and detect the new worker node (label "GW"). Then, according to the scheduling policies, it will allocate jobs on this new resource by interfacing with the Grid execution service (GRAM).

4 Experimental Results

4.1 Testbed Description

The behavior of the previous deployment strategy will be analyzed on a testbed based on Globus Toolkit 4.0.3 and GridWay 5.2.1. The testbed consists of three resources: two SGE clusters, and a dedicated system hosting the management services (meta-scheduler and infrastructure). The main characteristics of these machines are described in Table 1.

Table 1. Characteristics of the testbeds resources

Host	OS	CPU	Memory	Services
UCM frontend	Debian Etch	P4 HT 3.2GHz	1 GB	GT4.0.5, SGE NIS, NFS, VWS
UCM WN (2)	Debian Etch	2 x P4 HT 3.2GHz	256MB	DHCP, Xen 3
ESA frontend	Fedora Core 6	Xeon 1.70GHz	768MB	GT4.0.4, SGE NIS, NFS, VWS
ESA WN 1	Fedora Core 6	2 x Xeon 2.20GHz	2GB	DHCP, Xen 3
ESA WN 2	Fedora Core 6	Xeon 1.70GHz	768MB	DHCP, Xen 3
ESA WN 3	Fedora Core 6	Xeon 2.20GHz	2GB	DHCP, Xen 3
Manager Server	Debian Etch	Pentium M 1.4GHz	768MB	GT4.0.3, GridWay 5.2.1

4.2 Functional Analysis

The aim of the experiments presented in this section is to obtain a clear understanding of the interaction of all the components that form the architecture. Also, we will study the overhead induced by each component, so we can evaluate the the cost of the benefits that virtualization adds to the Grid in terms of flexibility, lower operational costs and enhanced security.

Deployment Overhead. Let us start by measuring the deployment time of a VM under several conditions. As different hardware configurations lead to different results, we have only employed one cluster (UCM, see Table 1) to carry out these tests. Table 2 presents the average results (over 25 runs) obtained while deploying one, two and three VMs on the same physical machine, respectively.

In Table 2, time precision is one second. Hence, the values obtained in the measurements of CR, DA, VMB, VMR and SR can be considered the same for the three experiments. We would like to note that the overhead induced by the SGE layer is negligible (the time to register a worker node in the cluster is less than a 1% of the total deployment time).

Although most of the overheads are constant regardless of the number of VMs deployed, the growth of propagation time (PT) makes this approach unfeasible. These results suggest to use a sequential deployment approach for worker nodes.

On a sequential deployment, VMs are deployed one at a time. Only one image is being transferred from the image server to the execution node, so I/O overhead is greatly reduced, obtaining nearly a constant time. Both approaches can be compared in figure 2. We would like to remark that when deploying more than 3 VMs simultaneously most of the times error situations occurred.

Table 2. Deployment time (in seconds) when starting one, two and three VMs simultaneously. CR: Command Received in Physical Node. PT: Image propagation time. DA: DHCP server alter. VMB: Xen start time. VMR: VM booting time. SR: SGE registering.

VMs Created	CR	PT	DA	VMB	VMR	SR	Total
1	2	96.8	4.58	3	10.67	1.5	118.58
2	2.54	279.58	4.82	6.44	11.73	1.88	308.02
3	1	472	3.75	5.8	11.83	1.5	495.88

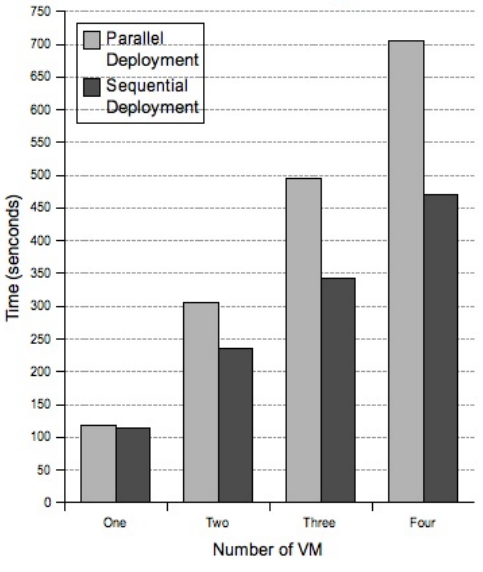

Fig. 2. Deployment overhead for simultaneous and sequential approaches

Shut Down Overhead. When shutting down a VM (Table 3), we have measured three relevant values. In this case the time is constant regardless of the number of VMs being shut down.

Overhead introduced by SGE when shutting down a VM can be minimized by reducing the polling time. Default value is 300 seconds, so it takes an average of 150 seconds to detect the new situation. Reducing polling time limits the overhead, although increments network usage. System administrator must tune this value according to the number of nodes in the cluster and average VM uptime.

Grid Integration. Figure 3 shows a global view of the interaction of all the system components. As has been discussed previously, the time to start a virtual worker node (time since the Infrastructure Manager requests a worker node, 114 sec., till it is registered in the LRMS, 2 sec.) is roughly 2 minutes.

Table 3. Times (in seconds) when a worker node is shut down

Number	Command Received	VM Destroyed	SGE	Total
1	0.78	6.22	145	152
2	0.88	6.46	158	165.33
3	0.67	7.33	151	159

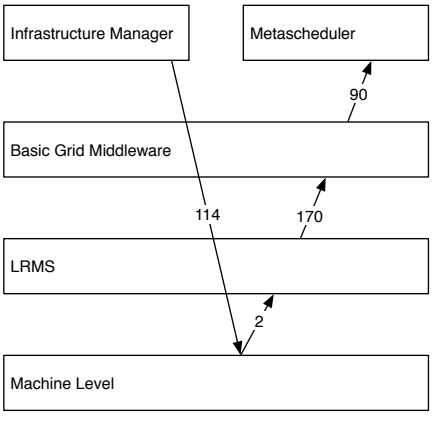

Fig. 3. Sequence of actions in a VM deployment and the associated overhead (in seconds)

The time to register the new slot in the Grid Information system (MDS4) is about 170 seconds. It is worth pointing out that MDS publishing time is greater than the time employed on deploying one VM plus SGE register time. Therefore, when sequentially deploying several VMs both times overlap, producing an additional time saving. The MDS and GridWay overhead can be limited by adjusting their refresh polling intervals.

When switching down, the same steps are accomplished. In this case, the time until the operation is accomplished at the machine layer is greatly reduced, from 114 to 7 seconds. However, time until LRMS detects the lack of the VM is incremented, from 2 to about 150 seconds. It is interesting to note that the meta-scheduler could assign jobs to the cluster during the worker node shutting down time. In this case the meta-scheduler should be able to re-schedule this job to another resource.

Virtualization Overhead. Virtualization technology imposes a performance penalty due to an additional layer between the physical hardware and the guest operating system. This penalty depends on the hardware, the virtualization technology and the kind of applications being run. Two good performance comparisons of VMware and Xen were conducted by the computer science departments at University of Cambridge [15]. and Clarkson University [16]. On these studies,

Xen performed extremely well in any kind of tasks, with a performance loss between 1 and 15%. VMware also achieves near-native performance for processor-intensive tasks, but experiences a significant slow-down (up to 88%) on I/O bound tasks.

VWS development team measured VWS performance in a real-world grid use case [17], a climate science application, achieving about a 5% performance loss. In a previous work [6], we have obtained a similar result (about 10% loss) when employing virtual machines in a Grid to execute a high throughput scientific application. This is an acceptable result, regarding the benefits in terms of modularity, portability and simplified application development.

5 Conclusions and Future Work

In this paper, we have presented a Grid architecture for the dynamic provisioning of computing elements. The benefits of this architecture is a flexible Grid able of supporting different VOs on a shared and configurable infrastructure, while reducing its operational costs.

The results obtained on the performance tests show that the proposed architecture and technologies represent a feasible solution. With a daily cluster reconfiguration, induced overhead is less than 1%. Added up with Xen performance lost, it remains under 10%. However, it provides attractive benefits like increased software robustness, easier cluster administration and enhanced security.

In the future, we will improve the decision making system on the resource manager, in order to optimize the deployment of virtual machines under different conditions and usage. We will also explore the possibilities that the proposed technology offers to outsourced grids, allowing dedicated service providers to supply resources on demand over the Internet.

References

1. http://www.amd.com
2. http://www.xen.org
3. http://www.vmware.com
4. Adabala, S., Chadha, V., Chawla, P., Figueiredo, R., Fortes, J., Krsul, I., Matsunaga, A., Tsugawa, M., Zhang, J., Zhao, M., Zhu, L., Zhu, X.: From Virtualized Resources to Virtual Computing Grids: The In-VIGO system. Future Generation Computer Systems 21(6) (April 2005)
5. Keahey, K., Foster, I., Freeman, T., Zhang, X.: Virtual Workspaces: Achieving Quality of Service and Quality of Life in the Grid. Scientific Programming Journal 13(4), 265–276 (2005)
6. Rubio-Montero, A.J., Huedo, E., Montero, R.S., Llorente, I.M.: Management of virtual machines on globus grids using gridway. In: IEEE International Parallel and Distributed Processing Symposium 2007, vol. 21, pp. 1–7 (2007)

7. Chase, J.S., Irwin, D.E., Grit, L.E., Moore, J.D., Sprenkle, S.E.: Dynamic virtual clusters in a grid site manager. In: HPDC 2003: Proceedings of the 12th IEEE International Symposium on High Performance Distributed Computing, Washington, DC, USA, p. 90. IEEE Computer Society, Los Alamitos (2003)
8. Sundararaj, A.I., Dinda, P.A.: Towards virtual networks for virtual machine grid computing. In: VM 2004: Proceedings of the 3rd conference on Virtual Machine Research And Technology Symposium, Berkeley, CA, USA, p. 14. USENIX Association (2004)
9. Keahey, K., Freeman, T., Rana, A., Norman, M., Wrthwein, F., Gardner, R.: Edge services framework for osg. OSG Document, 167-v1 (June 2005)
10. `http://www.amazon.com/ec2`
11. `http://gridengine.sunsource.net`
12. `http://www.eu-egee.org`
13. `http://www.gridway.org`
14. Huedo, E., Montero, R.S., Llorente, I.M.: Evaluating the Reliability of Computational Grids from the End User's Point of View. Journal of Systems Architecture 52(12), 727–736 (2006)
15. Barham, P., Dragovic, B., Fraser, K., Hand, S., Harris, T., Alex Ho, R.N., Pratt, I., Warfield, A.: Xen and the Art of Virtualization. In: Symposium on Operating Systems Principles, pp. 164–177 (October 2003)
16. Clark, B., Deshane, T., Dow, E., Evanchik, S., Herne, M.F.J., Matthews, J.N.: Xen and the Art of Repeated Search. In: USENIX Annual Technical Conference, p. 47 (2004)
17. Foster, I., Freeman, T., Keahy, K., Scheftner, D., Sotomayor, B., Zhang, X.: Virtual clusters for grid communities. In: CCGRID 2006: Proceedings of the Sixth IEEE International Symposium on Cluster Computing and the Grid (CCGRID 2006), Washington, DC, USA, pp. 513–520. IEEE Computer Society, Los Alamitos (2006)