On the use of clouds for grid resource provisioning[☆]

Constantino Vázquez, Eduardo Huedo, Rubén S. Montero, Ignacio M. Llorente

Departamento de Arquitectura de Computadores y Automática
Facultad de Informática
Universidad Complutense de Madrid, Spain
September 27th, 2010

Abstract

Cloud computing is being built on top of established grid technology concepts. On the other hand, it is also true that cloud computing has much to offer to grid infrastructures. The aim of this paper is to provide the ability to build arbitrary complex grid infrastructures able to sustain the demand required by any given service, taking advantage of the pay-per-use model and the seemingly unlimited capacity of the cloud computing paradigm. It addresses mechanisms that potentially can be used to meet a given quality of service or satisfy peak demands this service may have. These mechanisms imply the elastic growth of the grid infrastructure making use of cloud providers, regardless of whether they are commercial, like Amazon EC2 and GoGrid, or scientific, like Globus Nimbus. This technology of dynamic provisioning is demonstrated in an experiment, aimed to show the overheads caused in the process of offloading jobs to resources created in the cloud.

Keywords: grid, cloud, adaptable architectures, automation, dynamic provision

1. Introduction

Cloud computing was arguably first popularized in 2006 by Amazon's Elastic Compute Cloud, which started offering virtual machines (VMs) for $0.10/hour using both a simple web interface and a programmer-friendly API. Although not the first to propose a utility computing model, Amazon EC2 contributed to the popularization of the Infrastructure as a Service (IaaS) paradigm, which became closely tied to the notion of cloud computing. An IaaS cloud enables on-demand provisioning of computational resources, in the form of VMs deployed in a cloud provider's datacenter (such as Amazon's), minimizing or even eliminating associated capital costs for cloud consumers, allowing capacity to be added or removed from their IT infrastructure in order to meet peak or fluctuating service demands, while only paying for the actual capacity used.

Over time, an ecosystem of providers, users, and technologies has coalesced around this IaaS cloud model. More IaaS cloud providers, such as GoGrid, FlexiScale, and ElasticHosts have emerged. A growing number of companies base their IT strategy on cloud-based resources, spending little or no capital to manage their own IT infrastructure. Other providers offer products that facilitate working with IaaS clouds, such as rPath's rBuilder, which allows dynamic creation of software environments to run on a cloud.

In general, an IaaS cloud consists of three main components, namely: a virtualization layer on top of the physical resources including network, storage and compute; the virtual infrastructure manager (VIM) that control and monitor the VMs over the distributed set of physical resources; and a cloud interface that provides the users with a simple abstraction to manage VMs. In recent years a constellation of technologies that provide one or more of these components have emerged. Therefore, a variety of hypervisors have been developed and greatly improved, most notably KVM, Xen and VMWare. Also, several VIM technologies that cover the functionality outlined above have appeared, like Platform VM Orchestrator, VMware DRS, or Ovirt. On the other hand, projects like Globus Nimbus[1], Eucalyptus[2] or OpenNebula[3], or products like VMware vSphere, that can be termed cloud toolkits, can be used to transform existing infrastructure into an IaaS cloud with cloud-like interfaces.

Cloud computing has emerged as a very promising paradigm to simplify and improve the management of current IT infrastructures of any kind and, in particular, grid ones.

[☆]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 MEDIANET Research Program S2009/TIC-1468, by Ministerio de Ciencia e Innovación, through the research grant TIN2009-07146, and by the European Union through the research grant RESERVOIR Contract Number 215605

Email addresses: `tinova@fdi.ucm.es` (Constantino Vázquez), `ehuedo@fdi.ucm.es` (Eduardo Huedo), `rubensm@dacya.ucm.es` (Rubén S. Montero), `llorente@dacya.ucm.es` (Ignacio M. Llorente)

[1]http://www.nimbusproject.org
[2]http://open.eucalyptus.com
[3]http://opennebula.org

Clouds, in their IaaS form, have opened up avenues in this area to ease the maintenance, operation and use of grid sites, and to explore new resource sharing models that could simplify in some cases the porting and development of grid applications. The first works about the joint use of clouds and grids are exploring two main approaches, namely:

- The use of virtual machines and clouds as an effective way to provide grid users with custom execution environments. Therefore the same grid site can easily support Virtual Organizations (VOs) with different (or conflicting) configurations.
- The access to grid resources in a cloud-way. So, the users will access raw computing capacity by-passing the classical grid middleware stack. This approach is also being considered as a natural way to attract business users to our current e-infrastructures.

The problem we are trying to address in this paper is the necessity to attend fluctuating and peak demands in high performance computing. For instance, a supercomputing center is subject to this needs, since projects can demand resources punctually for experiments, and they can do so even simultaneously. Moreover, meeting a Service Level Agreement (SLA) defining a Quality of Service (QoS) with the center's available resources can be challenging at certain times. The logical provisioning model will be to use the local infrastructure to attend all the existing demand, if possible (i.e., using their enterprise grid). If that is not enough, the excess load can be delegated to a partner grid, with which a previous arrangement has been made. This partner grid does not need to provide the same interface as the enterprise grid, but still a single point of access is desirable to the whole federated infrastructure. If the computing demand still overflows the existing resources, the center will need to use a cloud provider to perform temporary increase in its computing power. A unified point of access is even valuable as more heterogenous resources are added to the grid infrastructure.

In this paper will show a framework for monitoring service capacity and for growing grid infrastructures when it comes close to saturation, using cloud providers like Amazon EC2[4] and GoGrid[5] (commercial) or Globus Nimbus[6] (scientific); and we will show empirical data on an experiment showing this dynamic growth. This will provide the necessary components to build a grid infrastructure with a single point of access that can be adapted as in the aforementioned though experiment of the supercomputing center. One interesting characteristic of using the virtualization that conform clouds is the ability to provide resources with certain characteristics, that is, particular software libraries or specific configuration can be asked for in the requested virtual machines (VMs) to better fulfill the demands of the grid infrastructure.

The aim of this paper is to contribute with a solution to the challenge of the dynamic provision of resources using cloud providers. The structure of this paper is as follows. Section 2 references work related to this paper, while Section 3 unfolds a general architecture of the solution for the desired adapting grid infrastructure. Section 4 presents a solution to dynamically grow an existing grid infrastructure using resources from cloud providers. Finally, Section 5 states plans for future work and summarizes the conclusions of this paper.

2. Related Work

In the last decade we have witnessed the consolidation of several transcontinental grid infrastructures that have achieved unseen levels of resource sharing. In spite of this success, current grids suffer from several obstacles that limit their efficiency, namely:

- An increase in the cost and length of the application development and porting cycle. New applications have to be tested in a great variety of environments where the developers have limited configuration capabilities.
- A limitation on the effective number of resources available to each application. Usually different VOs require different software configurations, so an application can be only executed on those sites that support the associated VO. Moreover, the resources devoted to each VO within a site are usually static and cannot be adapted to the VO's workload.
- An increase in the operational cost of the infrastructure. The deployment, maintenance and distribution of different configurations requires specialized, time consuming and error prone procedures. Even worse, new organizations joining a grid infrastructure need to install and configure an ever-growing middleware stack.

This situation often leads to a struggle between the users, who need more control on their execution environments, and grid operators, who want to limit the heterogeneity of the infrastructure. As a result several alternatives to reconcile both positions have been explored in the past. For example, the SoftEnv project[7] is a software environment configuration system that allows the users to define the applications and libraries they need. Another common solution is to use a custom software stack on top of the existing middleware layer, usually referred as pilot-jobs. For example, the MyCluster Project [1] creates a

[4]http://www.amazon.com/ec2
[5]http://www.gogrid.com
[6]http://workspace.globus.org

[7]http://www.teragrid.org/userinfo/softenv

Condor or Sun Grid Engine (SGE) cluster on top of TeraGrid services; and similarly over other middleware we may cite: DIRAC [2], glideinWMS [3], PanDa[8] or the Falkon system [4]. These approaches essentially shifts the scalability issues from the application to the overlaid software layer, whereas the proposed solution transparently scales both the application and the computational cluster.

The idea of a virtual cluster which dynamically adapts its size to the workload is not new. It can seen being applied for instance in the cluster management software called COD (Cluster On Demand) [5], 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. Similarly, the VIOcluster [6] project enables dynamically adjusting the capacity of a computing cluster by sharing resources between peer domains.

Another area explored nowadays is the use of the existing grid infrastructure to build autonomic clouds [7]. This is achieved by creating virtual environments, with their associated benefits like flexibility and adaptation, across multiple physical domains, using a decentralized private overlay network system called IPOP (IP over P2P). In this fashion, a massive number of grid infrastructure could be turned into adaptable clouds in a non disruptive manner for the existing grid services, although they will be suitable for serving high throughput, high latency tolerant services.

However the most promising technology to provide VO with custom execution environments is virtualization. The dramatic performance improvements in hypervisor technologies made possible to experiment with virtual machines (VM) as basic building blocks for computational platforms. Several studies [8, 9] reveal that the virtualization layer has no significant impact on the performance of memory and CPU-intensive applications for HPC clusters. The first works in this area integrated resource management systems with VMs to provide custom execution environments on a per-job basis. For example Dynamic Virtual Clustering [10] and XGE [11] for MOAB and SGE job managers respectively. These approaches only overcome the configuration limitation of physical resources because VMs are bounded to a given resource and only exist during job execution. A similar approach has been implemented [12] at the grid level using the Globus GridWay Metascheduler. GridWay allows the definition of an optional phase before the actual execution phase to perform advanced job configuration routines. In this phase, the availability of the requested VM image in the cluster node is checked, transferring it from a GridFTP repository if needed. Then, in the execution phase, a script starts or restores the VM on a worker node and waits for its activation by periodically probing its services, and executes the user program after the VM is ready, the program copies all the input files needed to the VM, and executes the user program. This strategy does not require additional middleware to be deployed and is not tied to a given virtualization technology. However, since the underlying local resource management system is not aware of the nature of the job itself, some of the potential benefits offered by the virtualization technology (e.g.server consolidation) are not fully exploited.

More general approaches involve the use of virtual machines as workload units, which implies the change in paradigm from building grids out of physical resources to virtualized ones. For example, the VIOLIN [13] project proposes a novel alternative to application-level overlays based on virtual and isolated networks created on top of an overlay infrastructure. Also VMPlant service [14] provides the automated configuration and creation of VMs that can be subsequently cloned and instantiated to provide homogeneous execution environments across distributed grid resources. The InterGrid system uses as well VMs as building blocks to construct execution environments that span multiple computing sites [15]. Such environments can be created by deploying VMs on different types of resources, like local data centers, grid infrastructures or cloud providers. InterGrid uses OpenNebula as a component for deploying VMs on a local infrastructure. On the other hand, the In-VIGO [16] project adds some virtualization layers to the classical grid model, to enable the creation of dynamic pools of virtual resources for application-specific grid computing. Also in this line of work, several studies have explored the use of virtual machines to provide custom (VO-specific) cluster environments for grid computing. In this case, the clusters are usually completely build up of virtualized resources, as in the Globus Nimbus project [17], or the Virtual Organization Clusters (VOC) [18]. The latter allows, by means of real-time monitoring of LRMS queues, to launch VO-specific instances of Virtual Machines in order to serve the jobs belonging to the Virtual Organization that is in need of computing power. Once the jobs are completed, the Virtual Machines are powered down so the physical resources can be re-claimed. Aligned with this research direction, our research group has explored a hybrid model, so the cluster combines physical, virtualized and cloud resources [19].

It is important to note that the previous works also highlight that the use of virtualization in grid environments can greatly improve the efficiency, flexibility and sustainability of current productions grids. Not only by extending the classical benefits of VMs for constructing cluster, e.g. consolidation or rapid provisioning of resources [20, 21]; but also grid-specific benefits, e.g. support to multiple VOs, isolation of workloads and the encapsulation of services. Some EU projects like StratusLab[9] are studying a cloud-like provisioning model for grid-sites. In this way, a grid site exposes a typical cloud interface to

[8]https://twiki.cern.ch/twiki/bin/view/Atlas/Panda

[9]http://www.stratuslab.eu

instantiate and control virtual machines. This would allow accessing grid computing resources as cloud providers, complementing the existing grid middleware since the aim is for the cloud layer to be fully transparent to the layers above, and thus effectively taking benefit from the adaptability of service provisioning given by clouds applied to the batch processing goodness of grid infrastructures.

Our approach uses the concept of a dynamically adapting grid infrastructure, but more aligned with the concept of a Service Manager like Hedeby[10], although using VMs with specific configurations rather than configuring physical servers. In this aspect, our solution is similar to that provided by RightScale[11], but it differs from it since it is not a completely virtualized solution, but rather a way to extend a physical infrastructure by the punctual use of virtualized resources.

Not just the ability to fire VMs is enough for grid cloudbursting, it is also crucial the ability of the grid infrastructure to federate with other resources. For this, we used the GridWay metascheduler to architect this solution. It features a modular design, with adapters (Middleware Access Drivers in GridWay speak) that enables interoperation of different grid middleware stacks through the metascheduler, therefore making federation of grid resources feasible. This is explored in the work presented in [22], where adapters are used to access simultaneously grid infrastructures exposing different flavors of the globus grid middleware (pre Web Services and Web Services).Moreover, it can integrate resources being run by a local resource management system (LRMS) to the existing grid infrastructure cluster or it even has the potential to integrate with single computers via SSH.

3. Architecture

Dynamic provisioning poses various problems. One problem in this field is the importance of interoperability, i.e., being able to grow using any type of given resource, independently of what interface may be offering. Another problem is answering the question of when this dynamic growth is necessary and how to actually perform it. An even third issue can be the enforcement of a budget on this decision, taking into account expected CPU and network usage.

In this section, these problems are addressed by means of a grid infrastructure that can be flexibly built, that is aware of its load, featuring a single point of access and that can incorporate a new resource temporarily in an automatic fashion to satisfy heavy demands. Figure 1 sketches an architecture of such a solution. We can see that one of the building blocks is the GridWay metascheduler.

The flexible architecture of this metascheduler allows the use of adapters. Although sharing the same interfaces

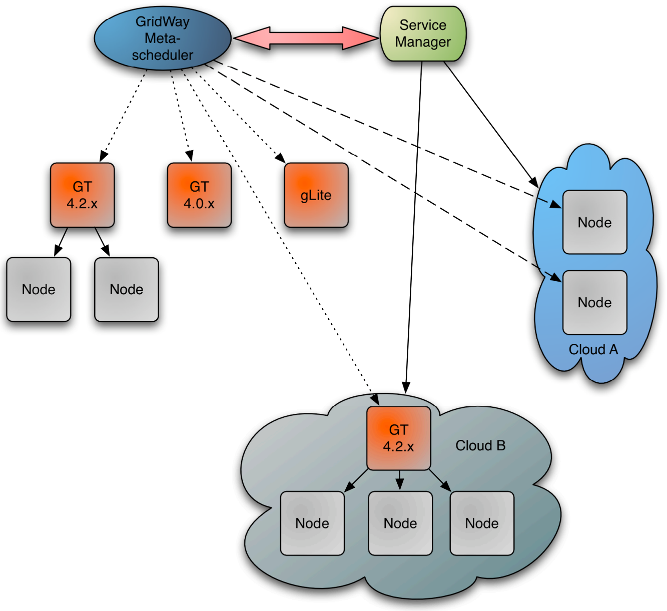

Figure 1: Architecture for an elastic grid infrastructure.

is the ideal way to achieve interoperation, sometimes there are different middlewares deployed in the sites to be federated, and it is unfeasible to unify them for a variety of reasons (politics, time constraints or ongoing migration or upgrades). This can be seen as the consequence of not having an accepted standard, and therefore, lack of interoperability. One possible solution to federate these different sites is to build a portal that uses different components (to submit jobs, gather information, transfer files, etc) to interface these sites. These components are designed specifically for a particular middleware stack or even version, and we can call them adapters.

GridWay already has a number of adapters (MADs), that enable access to different production grid infrastructures. Moreover, it also provides SSH MADs, so access to local resources can be achieved with decreased overhead, avoiding the need to have installed and configured in the nodes any grid software as, for instance, the Globus Toolkit.

Having described the federation approach, lets see the proposed architecture for dynamic provisioning, whose principal component is the the Service Manager. This component is used to monitor the GridWay metascheduler, and when the load of the system exceeds a threshold, detected using heuristics, it is responsible to grow the available grid infrastructure using specific adapters to access different cloud providers. This growth can be accomplished in two ways. The first one is by requesting a number (calculated with the aid of said heuristics) of single hosts. These need to have a previously defined software configuration that will then help them enroll in the available grid infrastructure. This corresponds to the use of Cloud A in Figure 1.

Another possibility is to deploy a full virtualized cluster, with a front-end controlling a number of slave nodes.

[10]http://hedeby.sunsource.net
[11]http://www.rightscale.com

This front-end can then enroll itself in the existing grid infrastructure, adding its capacity. The GridWay metascheduler features mechanisms to dynamically discover new hosts or sites which can be used for this purpose. This corresponds to the use of Cloud B in the figure.

Therefore, the provisioning model we envision is twofold. The first mode adds one single computing resource to the grid infrastructure. This computing resource can be accessible through a GRAM interface, meaning that the VM that is going to be awaken needs to have the Globus Toolkit installed and correctly configured. An even more practical approach will be to use just SSH access to perform job execution in this kind of nodes, the GridWay metascheduler already has a prototype of such SSH drivers. In this way, machines from cloud providers can be used out-of-the-box, with little to no configuration needed, basically SSH access is the only requirement. On the other hand, a second mode of growing the existing grid infrastructure would be to use these cloud providers in a slightly different fashion. Negotiation with the cloud provider will grant access to a virtual cluster, accessible through GRAM and controlled by a LRMS like for example PBS or SGE. This cluster will then be added to the federated grid infrastructure the same way as one of the physical sites we saw in the last section. Future work is planned to enrich the flexibility of the grid infrastructure by removing the GRAM layer, enabling GridWay to access the cluster by talking directly to the LRMS.

Moreover, this solution has another advantage. Not only it can dynamically increase the size of the grid infrastructure, but the added computing nodes can be awoken with different configurations. In other words, the Service Manager can be built in such a way that it won't only concern itself with the need to increase the computing capacity, but it can do so in a service oriented way. If there is one specific service which is suffering from the peak demand, the Service Manager can decide to increase the number of nodes prepared to satisfy such a service. For instance, if the service is an application that requires specific mathematic libraries, virtual machines images containing that specific libraries be chosen to be awoken as nodes to increase the grid infrastructure capacity.

4. Grid resource provisioning using clouds

A technique to guarantee pre-accorded QoS and, therefore, meet SLAs even in cases of high saturation of the grid infrastructure is presented in this section. Also, it gives a solution for peak demands that occur without enough time for planning the extension through federation. This solution involves the elastic growth of the computing infrastructure by means of a cloud provider, being that commercial (Amazon EC2, GoGrid) or scientific (Globus Nimbus), being the charge model the only difference between these two type of cloud providers.

To enable our grid infrastructure to be able to meet a given QoS and so satisfying predefined SLAs we need a component that is aware of the load of said infrastructure. Our solution consists of a Service Manager component that monitors the metascheduler in order to find when it should elastically grow (or, conversely shrink) the available resources by waking up nodes or entire clusters (or shutting them down). In short, this component is responsible for adapting the grid infrastructure to dynamic computing demands.

This solution takes advantage of the chosen GridWay metascheduler. It employs a dynamic scheduling system and therefore it can detect when a new machine has been added or removed from a grid infrastructure, and redistribute the work load. It also features fault detection and recovery capabilities. Transparently to the end user, it is able to detect and recover from any of the grid elements failure, outage or saturation conditions.

The Service Manager is in charge of monitoring the metascheduler and to detect and excess of load for the available resources. In order to detect this excess, a set of heuristics have to be defined, so they define the threshold of number of pending jobs waiting for resources, and the load present on the available resources. A second set of heuristics is needed to decide which cloud provider is going to be used to elastically grow the cluster. These heuristics should be based on economics criteria, minimizing the total cost of CPU and network usage. In this line, a good starting point would be the work done in budget constrained cost-time optimization algorithms for scheduling [23].

Optionally, there is even a third set of rules that the Service Manager need to employ in case that it is aware of what service demands need to be satisfied. This rules will be used to decide which type of VM is going to be awaken to satisfy the excess of demand. For instance, in the context of a supercomputing center, VMs with a certain Virtual Organization (VO) configuration can be the ones chosen to be up, if that VO has suddenly increased its demand for computing power.

4.1. Description of the Experiment

This experiment is designed to evaluate the overhead incurred in the management of a new worker node in a virtualized cluster being executed in Globus Nimbus. Hence, for the purpose of this experiment, the virtual cluster is already being executed, and we are measuring the overheads between the different layers of our architecture in the process of adding a new worker node, i.e., the overheads caused by awaking the worker node, shutting it down, being it detected by the SGE, the MDS and GridWay, and the processing overhead incurred by this node since it is virtualized. This experiment is performed on a local cloud deployed in our laboratory at dsa-research.org, using Globus Nimbus as the cloud manager.

In order to better understand the chain of events and where the overheads occur lets see the actions taking place when the Service Manager decides to add a new worker node to the grid infrastructure. A graphical depiction of

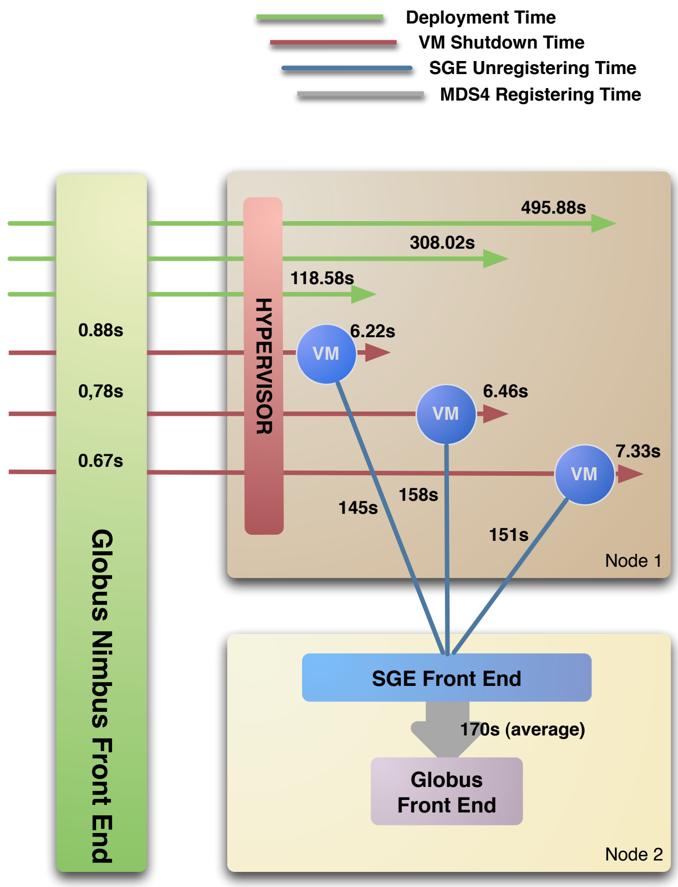

Figure 2: Grid resource provisioning using clouds overhead experiment.

the experiment, with measured times, can be seen in Figure 2. First, the Service Manager requests a new VM to Nimbus, which determines the best node to run the virtual machine, based on the resources requested (e.g memory). Afterwards, 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. 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. When these operations conclude, the VM is booted. When the VM has been deployed and is running, it registers on LRMS front-end as an execution host. After a given time, the grid information system (MDS) detects the new node and publishes it. Finally, GridWay the metascheduler will refresh the information of the available grid resources, and detect the new worker node. Then, according to the scheduling policies, it will allocate jobs on this new resource by interfacing with the grid execution service (GRAM). 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.

4.2. Evaluation of Overheads

Several experiments where run in the testbed in order to analyze the overheads caused by virtualization and the software layers corresponding to the proposed architecture.

The first overhead considered is caused by the *deployment* of a VM under several conditions. The experiment consisted in the deployment of one, two and three VMs within the same physical machine. Respectively, total times of deployment in seconds were 118.58, 308.02, 495.88 (**Deployment Time** in the figure). We would like to remark that deploying more than 3 VMs simultaneously resulted most of the times in error situations. Also, 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).

In the case of *shutting down* a VM (Table 1), we have measured three relevant values. **Command received** measures the time passed between the request is made to the Globus Nimbus front end and its arrival to the underlying hypervisor. **VM destroyed** expresses the time it takes to completely shutdown a VM since the shutdown order arrives to the hypervisor. Both times are consolidated in the figure's **VM Shutdown Time**. 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 (**SGE Unregistering Time** in the figure) 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 it increments network usage. System administrator must tune this value according to the number of nodes in the cluster and average VM uptime.

Once the VM is being booted, time is needed until it is enrolled for use in the original grid infrastructure, and complementarily, time is also needed to plug the node out

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

Table 1: Times (in seconds) when a worker node is shut down

of the infrastructure. These overheads can be called *grid integration* overheads. The time to start a virtual worker node (time since the Service Manager requests a worker node, 114 sec., till it is registered in the LRMS, 2 sec.) is roughly 2 minutes. The time to register the new slot in the Grid Information System (MDS4) is about 170 seconds (**MDS4 Registering Time** in the figure). 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 shutting 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 metascheduler could assign jobs to the cluster during the worker node shutting down time. In this case the metascheduler should be able to re-schedule this job to another resource.

Virtualization technology imposes a performance penalty due to an additional layer between the physical hardware and the guest operating system. This penalty (that we can call the *processing* overhead) depends on the hardware, the virtualization technology and the kind of applications being run. As some studies suggest [24, 25], Xen performs 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. Nimbus development team measured its performance in a real-world grid use case [26], a climate science application, achieving about a 5% performance loss. Previous results [12, 27] indicate that, in general, the virtualization platform has no significant impact on the performance of memory and CPU-intensive applications for HPC clusters.

5. Conclusions and Future Work

We have shown an architecture to build any type of arbitrary complex grid infrastructures with a single point of access, which is able to dynamically adapt its size (and therefore, capacity) using a cloud provider to react to peak demands and/or meet SLAs.

This paper opens the path for a number of research lines to be followed and developments to be done. On one hand, there is room for the development of new adapters for the GridWay to access a greater number of different types of grid infrastructures. On the other hand, the problem posed by the need to schedule workloads across several grid infrastructures needs heuristics be developed. Said heuristics need to be implemented in the metascheduler, and use information gathered by it to decide which infrastructure to send the job to, taking into account their workload, their hardware characteristics, job execution history, and even user or group quotas.

Moreover, there is also work to be done in the heuristics needed to tune the Service Manager. This component needs to be aware of the characteristics of the service being offered to correctly adjust its capacity in case of a demand increase. Based on history, the Service Manager could even anticipate a peak demand and readies the service for it, adding resources to the grid infrastructure. It is also crucial that the heuristics work upon an economic model that knows different cloud provider characteristics to reach a trade off between service efficiency and price.

References

[1] Walker, E., Gardner, J., Litvin, V. and Turner, E.: Creating Personal Adaptive Clusters for Managing Scientific Jobs in a Distributed Computing Environment. In Proceedings of the IEEE Challenges of Large Applications in Distributed Environments. (2006) 95–103

[2] A. Tsaregorodtsev, V. Garonne, and I. Stokes-Rees: DIRAC: A Scalable Lightweight Architecture for High Throughput Computing. In Fifth IEEE/ACM International Workshop on Grid Computing (GRID'04), 2004, pp. 19-25.

[3] I. Sfiligoi: Making science in the Grid world: using glideins to maximize scientific output. Nuclear Science Symposium Conference Record, 2007. NSS '07. IEEE 2, Honolulu, HI, USA, 2007, pp. 1107-1109.

[4] Raicu, I., Zhao, Y., Dumitrescu, C., Foster, I. and Wilde, M.: Falkon: a Fast and Light-weight tasK executiON framework. In Proceedings of the IEEE/ACM SuperComputing, November 2007.

[5] Chase, J., Irwin, D., Grit, L., Moore, J. and Sprenkle, S.: Dynamic Virtual Clusters in a Grid Site Manager. In Twelfth IEEE Symposium on High Performance Distributed Computing (HPDC), Seattle, Washington, June 2003.

[6] P. Ruth, X. Jiang, D. Xu and S. Goasguen: Virtual Distributed Environments in a Shared Infrastructure. IEEE Computer, Special Issue on Virtualization Technologies, May 2005.

[7] Murphy, Michael and Abraham, Linton and Fenn, Michael and Goasguen, Sebastien: Autonomic Clouds on the Grid, In Journal of Grid Computing, Springer Netherlands (1) vol 8, pp 1-18I (2010)

[8] L. Youseff, R. Wolski, B. Gorda and C. Krintz: Paravirtualization for HPC Systems. Workshop on XEN in HPC Cluster and Grid Computing Environments (XHPC), held in conjunction with The International Symposium on Parallel and Distributed Processing and Application (ISPA 2006), December 2006.

[9] L. Youseff, K. Seymour, H. You, J. Dongarra and R. Wolski: The Impact of Paravirtualized Memory Hierarchy on Linear Algebra Computational Kernels and Software. High Performance Distributed Computing (HPDC), Boston, June 2008.

[10] W. Emeneker, D. Jackson, J. Butikofer and D. Stanzione: Dynamic Virtual Clustering with Xen and Moab. Lecture Notes in Computer Sience, 2006, 4331 pp. 440-451 Proc. of the Frontiers of High Performance Computing and Networking, ISPA 2006 Workshops.

[11] N. Fallenbeck and H.J. Picht and M. Smith and B. Freisleben: Xen and the Art of Cluster Scheduling. First International Workshop on Virtualization Technology in Distributed Computing (VTDC), 2006.

[12] Rubio-Montero, A.J., Montero, R.S., Huedo, E. and Llorente, I.M.: Management of Virtual Machines on Globus Grids Using GridWay. In Proceedings of the 4th High-Performance Grid Computing Workshop, in conjunction with 21st IEEE International Parallel and Distributed Processing Symposium, pp 1–7 (2007).

[13] X. Jiang and D. Xu: Violin: Virtual internetworking on overlay infrastructure. In Proceedings of the 2nd International Symposium on Parallel and Distributed Processing and Applications, Dec 2004.

[14] I. Krsul, A. Ganguly, J. Zhang, J. A. B. Fortes, and R. J. Figueiredo: VM-Plants: Providing and managing virtual machine execution environments for grid computing. In Proceedings of the 2004 ACM/IEEE Conference on Supercomputing, 2004.

[15] di Costanzo, A. and de Assuncao, M.D. and Buyya, R.: Harnessing Cloud Technologies for a Virtualized Distributed Computing Infrastructure, In IEEE Internet Computing (5) vol 13, pp 24-33 (2009).

[16] 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):pp. 896909, June 2005.

[17] T. Freeman and K. Keahey: Flying Low: Simple Leases with Workspace Pilot. In Euro-Par, 2008.

[18] Michael A. Murphy and Sebastien Goasguen: Virtual Organization Clusters: Self-provisioned clouds on the grid, Future Generation Computer Systems, 26 (8), pp. 1271-1281, February 2010.

[19] I. M. Llorente, R. Moreno-Vozmediano and R. S. Montero: Cloud Computing for On-Demand Grid Resource Provisioning. In: Advances in Parallel Computing, Volume 18 (2009): High Speed and Large Scale Scientific Computing, pp. 177 - 191. IOS Press, 2009

[20] H. Nishimura, N. Maruyama and S. Matsuoka: Virtual clusters on the fly - fast, scalable, and flexible installation. In CCGRID 2007, Seventh IEEE International Symposium on Cluster Computing and the Grid, May 2007.

[21] W. Emeneker and D. Stanzione: Dynamic virtual clustering. In IEEE Cluster 2007,Austin, TX, September 2007

[22] E. Huedo, R.S. Montero and I.M. Llorente: A modular metascheduling architecture for interfacing with pre-WS and WS Grid resource management services Future Generation Computing Systems 23 (2), pp 252-261 (2007)

[23] R. Buyya, M. Murshed, D. Abramsin and S. Venugopal: Scheduling Parameter Sweep Application on Global Grids: A Deadline and Budget Constrained Cost-Time Optimisation Algorithm. International Journal of Software: Practice and Experience (SPE) **5** (2005) pp 491–512

[24] P. Barham., B. Dragovid, K. Fraser, S. Hand, T. Ahrris, R.A. Ho, I. Pratt, A. Warfield: Xen and the Art of Virtualization. In Symposium on Operating Systems Principles. pp 164–177 (October 2003)

[25] Clark, B., Deshane, T., Dow, E., Evanchik, S., Herne, M. and Matthews, J.: Xen and the Art of Repeated Search. In USENIX Annual Technical Conference, pp 47–47 (2004).

[26] Foster, I., Freeman, T., Keahy, K., Scheftner, D., Sotomayor, B., Zhang, X.: Virtual clusters for grid communities. In Proceedings of the Sixth IEEE International Symposium on Cluster Computer and the Grid (CCGRID 06), IEEE Computer Society pp 513–520 (2006).

[27] Rodriguez, M., Tapiador, D., Fontan, J., Huedo, E., Montero, R.S. and Llorente, I.M.: Dynamic Provisioning of Virtual Clusters for Grid Computing. In Proceedings of the 3rd Workshop on Virtualization in High-Performance Cluster and Grid Computing (VHPC 08), in conjunction with EuroPar08, 2008.