Dynamic Provision of Computing Resources from Grid Infrastructures and Cloud Providers [*]

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

E-mail: {tinova, ehuedo}@fdi.ucm.es, {rubensm, llorente}@dacya.ucm.es

Abstract

Grid computing involves the ability to harness together the power of computing resources. In this paper we push forward this philosophy and show technologies enabling federation of grid infrastructures regardless of their interface. The aim is to provide the ability to build arbitrary complex grid infrastructure able to sustain the demand required by any given service. In this very same line, this paper also 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. Both these technologies of federation and dynamic provisioning are demonstrated in two experiments. The first is designed to show the feasibility of the federation solution by harnessing resources of the TeraGrid, EGEE and Open Science Grid infrastructures through a single point of entry. The second experiment is aimed to show the overheads caused in the process of offloading jobs to resources created in the cloud.

1. Introduction

In high performance computing there is the necessity to attend fluctuating and peak demands. 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 we will propose and evaluate an architecture to build a grid infrastructure in a very flexible way, being able to hire computer resources with potentially different interfaces; and we will show an experiment to prove its feasibility. This will be done with technologies available nowadays rather than the intention of developing standards for the future, i,e, using interoperation rather than interoperability [1]. Furthermore, we 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 [1] and GoGrid [2] (commercial) or Globus Nimbus [3] (scientific); and again we will show empirical data on an experiment showing this dynamic growth. This will provide with 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 abil-

[*]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, through the research grant TIN2006-02806, and by the European Union through the research grant RESERVOIR Contract Number 215605

[1]http://www.amazon.com/ec2
[2]http://www.gogrid.com
[3]http://workspace.globus.org

ity 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 therefore twofold, and it contributes with solutions to two related but different challenges:

- The interoperation of different grid infrastructures.
- The dynamic provision of resources using cloud providers.

The structure of this paper is as follows. Section 2 unfolds work related with this paper, while Section 3 unfolds a general architecture of the solution for the desired adapting grid infrastructure. Section 4 deals with the problem of grid interoperation, showing an experiment harnessing the computing power of resource pertaining to the TeraGrid, EGEE and Open Science Grid infrastructures. Section 5 presents a solution to dynamically grow an existing grid infrastructure using resources from cloud providers. Finally, Section 6 states plans for future work and summarizes the conclusions of this paper.

2. Related Work

Interoperation is an intrinsic characteristic of grid technologies, which basically consist on the aggregation of heterogenous resources in which interoperation obviously plays a crucial factor. Over the time, grid middlewares have been evolved without agreement between parts, resulting in incompatible interfaces for grid infrastructures. There are a numerous efforts focused on what is available nowadays, trying to provide federation solutions for the short term. This is exactly the aim of the Grid Interoperation Now [2] (GIN) group of the OGF. We can also remark efforts to interoperate different grid middlewares (for example, Globus and UNICORE [3]). It is also interesting to see efforts like InterGrid [4], proposing the creation of InterGrid Gateways (IGGs) to interconnect the existing different grid islands. Moreover, there are various works enabling interoperability between existing metaschedulers [5]. There is even an OGF group devoted to this research line, the Grid Scheduling Architecture (GSA) research group. Our solution takes advantage of the modular architecture of the GridWay metascheduler to use different adapters to access grid infrastructures with different interfaces, allowing to do so with a single point of access. Interoperation efforts are particularly important in the context of the EGEE infrastructure with respect to its National Grid Infrastructures (NGIs). Basically, each state contribute to the whole of the EGEE infrastructure providing resources grouped in a NGI. Therefore, we an think of the EGEE as a federation of NGIs, so interoperation becomes a key aspect between NGIs.

On the other hand, regarding on-demand provision of computational services, different approaches have been proposed in the literature. Traditionally, these methods consist in overlaying a custom software stack on top of an existing middleware layer, like for example the MyCluster Project [6] or the Falkon system [7]. 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. Jeffrey Chase et al., from Duke University, describe [8] 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 VMs, and the system is studied from a grid perspective. The use of virtualization to provide on-demand clusters has been also studied in the context of the Globus Nimbus [9]. Globus Nimbus provides a WSRF interface to launch heterogeneous clusters on a cloud. However, these clusters can not be easily integrated with the local resources nor can be supplemented with other cloud providers.

Finally, Amazon Elastic Computing Cloud 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. UnivaUD is an example of a company offering solutions to virtual provisioning by monitoring applications, gathering and analyzing detailed performance metrics, and then driving appropriate provisioning actions based on these metrics to meet established customer SLAs [4].

Our approach uses the concept of a dynamically adapting grid infrastructure, but more aligned with the concept of a Service Manager like Hedeby [5], although using VMs with specific configurations rather than configuring physical servers. In this aspect, our solution is similar to that provided by RightScale [6], but it differs from it since it is not a completely virtualized solution, but rather a way to extend

[4] http://www.univaud.com/reliance/use-cases/provisioning.php

[5] http://hedeby.sunsource.net

[6] http://www.rightscale.com

a physical infrastructure by the punctual use of virtualized resources.

3. Architecture

Dynamic provisioning posses 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 a 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 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.

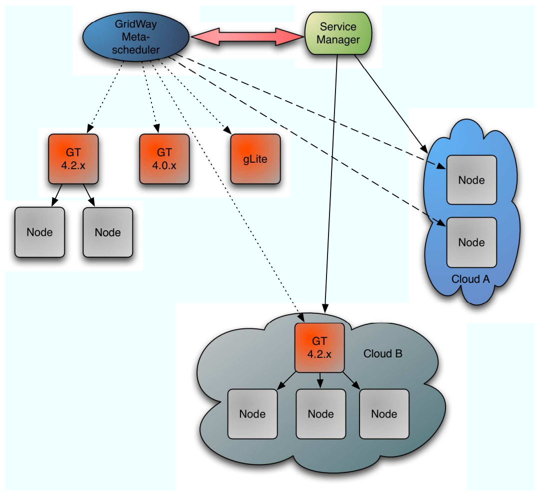

Figure 1. Architecture for an elastic grid infrastructure.

The flexible architecture of this metascheduler allows the use of adapters. Although sharing the same interfaces 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 interoperation. 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, called Middleware Access Drivers (MADs), that enable access to different production grid infrastructures. In Section 4, an experiment shows the metascheduler accessing the EGEE, TeraGrid and Open Science Grid, featuring different interfaces. Moreover, there are plans to provide SSH MADs, so access to local resources can be achieved with decreased overhead, and avoiding the need to have installed and configured in the nodes any grid software as, for instance, the Globus Toolkit.

Once described the federation approach, lets see the proposed architecture for dynamic provisioning, which principal component is the the Service Manager. It is used to monitor the GridWay metascheduler, and when the load of the system excesses 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. This front-end can then enroll itself to 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 non 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 Grid-

Way to access the cluster by talking directly to the local resource management system (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 waked with different configurations. In other words, if 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 awaken as nodes to increase the grid infrastructure capacity.

4. Provisioning from Heterogeneous Grid Infrastructures

We can think of *interoperation* as an immediate solution for the collaboration between two or more heterogeneous grids. On the other hand, *interoperability* focuses on the big picture and tries to bring together technologies that implement the grid infrastructure by means of standardization (like, for example, the Simple API for Grid Applications, SAGA, or the Basic Execution Service, BES, do). It is clear that this can not be achieved without a significant amount of effort and, more important to the point being made here, time. Thus, the need to provide interoperation and the justification of the GIN group within the OGF.

Since most common open standards to provide grid interoperability are still being defined and only a few have been consolidated, grid interoperation techniques, like for instance adapters, are needed. An adapter is, according to different dictionaries of computer terms, a device that allows one system to connect to and work with another. The aim of this section is to show the feasibility of the adapter-based approach to interoperation.

GridWay's architecture is flexible enough to allow for the use of adapters to achieve interoperation between infrastructures exposing different interfaces. The experiment shown later in this section proves the feasibility of this approach. Notwithstanding, this solution in turn posses a new problem that requires the development of new heuristics for the optimal scheduling of jobs across resources of different infrastructures. Lets see an example in order to clarify what these heuristics will have to decide. For clarity's sake, we are going to assume that GridWay is configured to access two different infrastructures, one with 30 free nodes and another with 20; and GridWay receives a job array of 18 elements. Currently, GridWay scheduler doesn't take into account the notion of array for scheduling, so it will probably begin scattering jobs across both infrastructures. Nonetheless, it is intuitively clear that we would want to send the whole array to, for instance, the 20-node infrastructure, making space for a possible next 30 job array. Some work has been done in this direction [10], and it can be taken as a good starting point for new heuristics for metascheduling.

GridWay's adapters are the Middleware Access Drivers (MADs), enabling the metascheduler to access different infrastructures simultaneously and seamlessly. GridWay has evolved over time to take advantage of the adapters technique, it is interesting to see its evolution, from a first tentative effort to harness both EGEE and the IRISGrid infrastructures [11], to the seminal work that produced its full blown current modular architecture [12], motivated by the interoperation between grid resource management services provided by Globus. There are three types of adapters: execution, transfer and information MADs.

4.1 Description of the Experiment

GridWay metascheduler was configured to access four different infrastructures. Not only different interfaces were the problem, but also different versions of the same middleware posed their own issues. Furthermore, different configurations of even the same version of the same middleware stacks are troublesome for the correct interoperation of the four infrastructures. GridWay was set to use different adapters especially configured to access the following infrastructures:

- *Open Science Grid*: This infrastructure offers two versions of the Globus Toolkit, deployed using VDT[7]: the pre Web Services (preWS) and Web Services (WS) versions.

- *TeraGrid*: Again, two versions of Globus are offered. Configuration for the file transfer was different than in other infrastructures since the Storage Element (SE) was a separate machine (sharing homes with the Computing Element).

- *EGEE*: This infrastructure uses the gLite middleware stack, which is based on Globus preWS. The preWS MADs are used, with a special file staging configuration. An information MAD for the Berkely Database Information Index (BDII) using the Glue scheme is used for host monitoring.

- *UCM:* This is dsa-research.org group local infrastructure at Universidad Complutense de Madrid. Probably due to the fact that the Globus WS MADs were developed against this very cluster, no extra configuration was needed for GridWay to access it.

[7]http://vdt.cs.wisc.edu/

Three iterations of two hundred jobs each were sent to the GridWay metascheduler, which in turn distribute them along the four infrastructures, in order to show the federation feasibility, using the Embarrassingly Distributed (ED) benchmark from the NAS Grid Benchmark suite.

Figure 2 depicts the set-up of the experiment. Resources are accessed by different adapters (dotted arrows represent preWS interfaces, solid arrows represent Web Services, WS, interfaces). For illustrative purposes, one resource from each infrastructure was chosen for this paper experiments, so they could be performed in a more controlled environment. A complete resource listing for the adapters scenario can be seen in Figure 3. To show different adapters in action, WS MADs were used to access the TeraGrid (with Host Identifier , HID, 0 in the figure) and the UCM (HID 2) infrastructure, while the preWS MADs were chosen to access the Open Science Grid (HID 1) and the EGEE (HID 3) infrastructure. It is worth noting that WS versions 4.0.x and 4.2.x of the Globus Toolkit are incompatible, meaning that clients from one version cannot access servers from another. GridWay is able to use MADs from toolkits of both versions, and therefore access seamlessly Globus containers of both versions, solving neatly the incompatibility between them and allowing a smooth upgrading process. In this experiment, GridWay accesses version 4.0.x of the Globus Toolkit to use resources from the TeraGrid, and version 4.2.x to use UCM resources.

4.2 Analysis of Results

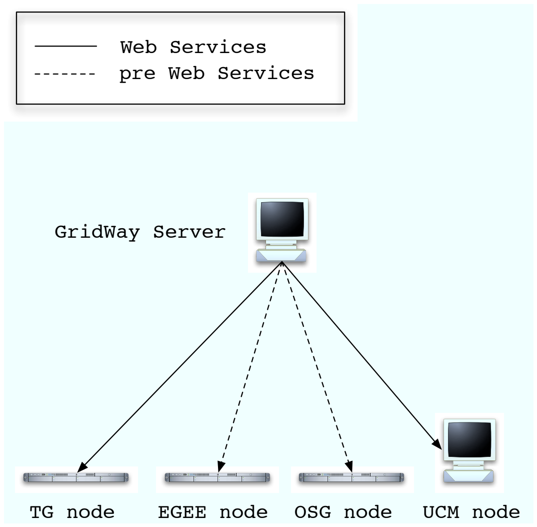

Figure 2. Interoperation across infrastructures

Figure 4 shows the number of jobs against the infrastructures where the jobs were executed. There is an approximately even distribution between our local cluster (*UCM*), the Open Science Grid preWS resource (*OSG preWS*) and the TeraGrid WS one (*TG WS*), while the *EGEE* resource shows a smaller ratio of jobs completed. This is due to the chosen EGEE site for the EGEE, ramses.dsic.upv.es, having lower computing power (both CPU and memory) than the resource chosen for this experiment in the other infrastructures.

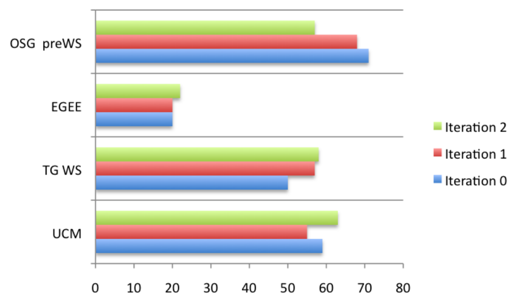

Figure 4. Job distribution across infrastructures

The capability to add new resources with potentially different interfaces to an existing grid infrastructure is therefore shown by this experiment, maintaining a single point of access, in this case the GridWay metascheduler.

5. Extension to Cloud Providers

Last section model provides a single point of access to a grid infrastructure that can be extended using different providers with potentially different interfaces. In this section it is presented a technique to guarantee pre accorded QoS and, therefore, meet SLAs even in cases of high saturation of the grid infrastructure. 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.

```
HID PRI  OS              ARCH   MHZ %CPU MEM(F/T)   DISK(F/T)  N(U/F/T)  LRMS               HOSTNAME
0   1                                                             0/5/5  PBS                tg-grid.uc.teragrid.org
1   1    Linux2.4.21-32. i686  2665  189 964/2006 62787/73964   0/10/10  jobmanager-condor  cmsgrid01.hep.wisc.edu
2   1    Linux2.6.24-17  x86_6 1995  100   18/499 10169/21817     0/5/5  SGE                aquila.dacya.ucm.es
3   1    ScientificSL4   i686   866    0  513/513         0/0   0/21/22  jobmanager-lcgpbs  ramses.dsic.upv.es
```

Figure 3. Resources as provided by the `gwhost` command.

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 & 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 [13].

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.

5.1 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 that takes place when the Service Manager decides to add a new worker node to the grid infrastructure. 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.

5.2 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 con the same physical machine. Respectively, total times of deployment in seconds were 118.58, 308.02, 495.88. We would like to remark that deploying more than 3 VMs simultaneously derived 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. 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.

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

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 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. 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. Two good performance comparisons of VMware and Xen were conducted by the computer science departments at University of Cambridge [14]. and Clarkson University [15]. 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. Nimbus development team measured its performance in a real-world grid use case [16], a climate science application, achieving about a 5% performance loss. Previous results [17, 18] indicate that, in general, the virtualization platform has no significant impact on the performance of memory and CPU-intensive applications for HPC clusters.

6. 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. Regarding the provisioning from heterogenous infrastructures, there is room for the development of new adapters to access a greater number of different types of grid infrastructures. Currently, there are already two new drivers being developed, one for the new job execution service from gLite, called CREAM, and one set of drivers to interoperate with the UNICORE middleware. Complementary to these developments, it will be interesting to add SSH adapters to the GridWay metascheduler so it is able to interface with "raw" machines, i.e., without the need of installing a grid middleware layer. This will suit perfectly the use of VMs from cloud providers such as Amazon EC2, since the only configuration to be done would be to obtain SSH access. Also, adapters for direct access to LRMS like SGE or PBS will add a great flexibility in the grid infrastructures that can be built using this architecture. One good option for this adapter will be to develop it against Distributed Resource Management Application API (DRMAA), since then it will be possible to plug it to the multiple LRMS that supports this API.

Heuristics for the problem posed by the need to schedule workloads across several grid infrastructures need to be develop. Moreover, there is also work to be done in the heuristics needed to tune the Service Manager. From an economic model of the cloud provider to a set of rules to aid in the decision of adding resources to the grid infrastructure, all is needed to properly develop a Service Manager that is aware of what service is being offered and what is needed to properly satisfy its demand.

References

[1] Field, L.: Getting Grids to work together. CERN Computer Newsletter **41**(5) (Nov-Dec 2006) 8–9

[2] Riedel, M., et al.: Interoperation of World-Wide Production e-Science Infrastructures. Concurrency and Computation: Practice and Experience (2008) (in press).

[3] Breuer, D., Wieder, P., van den Berghe, S., von Laszewski, G., MacLaren, J., Nicole, D., Hoppe,

H.C.: A UNICORE-Globus Interoperability Layer. Computing and Informatics **21** (2002) 399–411

[4] Assunção, M.D., Buyya, R., Venugopal, S.: Intergrid: A Case for Internetworking Islands of Grids. Concurrency and Computation: Practice and Experience **20**(8) (July 2008) 997–1024

[5] Bobroff, N., Fong, L., Kalayci, S., Liu, Y., Martinez, J.C., Rodero, I., Sadjadi, S.M., Villegas, D.: Enabling Interoperability among Meta-Schedulers. In: Proceedings of 8th IEEE International Symposium on Cluster Computing and the Grid (CCGrid-2008). (2008) 306–315

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

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

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

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

[10] Leal, K., Huedo, E., Llorente, I.M.: Dynamic Objective and Advance Scheduling in Federated Grid. In: International Conference on Grid computing, high performance and Distributed Applications. Volume 5331. (2008) 711–725

[11] Vázquez-Poletti, J.L., Huedo, E., Montero, R.S., Llorente, I.M.: Coordinated Harnessing of the IRISGrid and EGEE Testbeds with GridWay. Journal of Parallel and Distributed Computing **5**(65) (May 2005) 763–771

[12] Huedo, E., Montero, R.S., Llorente, I.M.: A Modular Meta-scheduling Architecture for interfacing with pre-WS and WS Grid Resource Management Services. Future Generation Computing Systems **23**(2) 252–261

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

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

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

[16] 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 (2006) 513–520

[17] Rubio-Montero, A.J., Montero, R.S., Huedo, E., Llorente, I.M.: Management of Virtual Machines on Globus Grids Using GridWay. In: In Proceedings of the 4th High-Performance Grid Computing Workshop, in conjunction with 21st IEEE International Parallel and Distributed Processing Symposium (IPDPS-07). 1–7

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