Deployment of Cloud Computing Services with open source technologies

M.A. Alvarez¹, A. Fernandez-Montes¹, J.A. Ortega¹, A. Silva², L. Gonzalez-Abril³, F. Velasco³

¹ Computer Languages and Systems Dept., University of Seville, Spain
² Scientific Computing Center of Andalusia, Seville, Spain
³ Applied Economics I Dept., University of Seville, Spain

{afdez, maalvarez, jortega, luisgon, velasco}@us.es, asilva@cica.es

Abstract. A new approach technology, called the CICA GRID solution is presented which enables the expansion and replication of resources on demand in the service given by the Scientific Computer Centre of Andalusia to the research community. This approach is based on the current Cloud Computing trend. With a minimal technical staff effort, it quickly provides the resources any for demand beyond the normal workflow of an HPC cluster, by presenting a beta version of ReCarta architecture and user interface. This system acts as an access interface for non-technical users to a private Cloud Computing system.

Key words: IaaS, Cobbler, Koan, Puppet, OpenNebula

1 Introduction

In recent years, a concept has been launched which has potential to transform the way in which computers are used and managed: “Cloud Computing”. This technology promises to realise the desired objective of transforming computing resources into a single utility that can be obtained in any quantity for the time needed, that is, it provides flexibility (elasticity).

For the HPC/Grid Computing/Scientific computing area, this resulting elasticity is especially interesting since it enables resources to be managed in a controlled environment. When a new project is considered, researchers estimate their computing needs. The resources available to these researchers often fail to include the features required. The researchers must therefore spend their time configuring their needs to the available resources, which can be a problem. Hence, a solution that allows to these researchers to count on personalized infrastructure must be developed. This solution could be based on Cloud Computing, used in the form of an Infraestructure As A Service (IaaS) which enables a supply of resources to be made available in answer to a demand peak.

The Scientific Computer Center of Andalusia (CICA), in response to the needs of researchers and to improve the Andalusian Supercomputing Network
(RASCI), has implemented a technology called CICA Grid solution, which enables the expansion or replication of resources depending on research demands. This solution of GRID environments, given by the CICA in collaboration with the Spanish National Grid Initiative, features a cloud of high scalability and quick configuration for new resources. The main demand of GRID environments is the provision of a solution to the project of Cloud Computing, born in CICA, whose aim is to be able to quickly attend to job demands that exceed the normal workflow expected for an HPC cluster, with a minimal technical staff effort.

The developed approach, called Recarta [14], incorporates three tools to carry out its function: Cobbler, to provide on-demand virtual machines; Puppet, for the management of the systems provided; and OpenNebula, for the deployment of these systems in a physical infrastructure. It is worth noting that these tools are hidden from the user by a web interface.

The paper is organized as follows: the Cloud Computing technology is briefly presented in the next section. In Section 3, the project motivation and system architecture are analyzed, and users’ tools and examples are given in Section 4. Benchmarks and features are shown in Section 5. Section 6 provides a final discussion and concludes this paper.

2 Cloud Computing

Cloud Computing refers to the hardware and software infrastructure which allows applications to be served across the web for end users, and provides computational resources and virtual hosting for users to build their own applications. It is worth noting that these services themselves have long been referred to as Software as a Service (SaaS). The datacenter hardware and software is what is called the Cloud.

There are two kinds of Cloud: Public Cloud [9], which is available for commercial purposes and pay-per-use, whose examples include Amazon EC2, Google AppEngine, ElasticHosts and Microsoft Azure [4]; and Private Cloud, which is found in an individual organization and whose access is allowed only to authorized members.

Cloud Computing systems can be classified as follows:

- IaaS (Infrastructure as a Service). The cloud provides the capacity of brute processing and hosting. The user has to choose which applications to install and the use of computational capability. Examples of IaaS include EC2 and ElasticHosts.
- PaaS (Platform as a Service). The cloud provides a software platform and middleware services where the users could develop their applications. Examples of PaaS include Google AppEngine and Microsoft Azure.
- SaaS (Software as a Service). This can be considered as the top abstraction level in the cloud. This model enables users to access an application hosted in the provider’s cloud. An example of this is SalesForce.
The advantages of SaaS to both end users and service providers are well-known. Service providers enjoy greatly simplified software installation and maintenance and centralized control over versioning; end users can access the service “anytime, anywhere”, share data and collaborate more easily, and keep their data stored safely in the infrastructure.

Cloud Computing does not change these arguments, but it does give more application providers the choice of deploying their product as SaaS without provisioning a datacenter: just as the emergence of semiconductor foundries gave chip companies the opportunity to design and sell chips without owning a manufacturing plant, Cloud Computing allows the deployment of SaaS and scaling on demand without building or provisioning a datacenter [5,6].

It should be pointed out at this point that the applied model in CICA is basically IaaS inside a private cloud which is accessed only by our users from the Andalusian Supercomputing Network (RASCI).

3 Project motivation and system architecture

A cluster for HPC, where Sun Grid Engine is currently used SGE as the LRM (Local Resource Manager) is supported by CICA. Furthermore, this cluster has about 30 machines which is part of the Spanish National Grid Initiative. The main motivation for our approach is the increase in requirements for access to computational resources in excess of the typical working scheme in a cluster with an LRM. Due to these requirements, a Cloud Computing and a virtualization as a way to satisfy that workload have been developed. Thus, a project, called “Recursos a la carta” (À-la-carte resources), was initiated where authenticated and authorized users, could design their own computational infrastructure, and then use and manage it across a comfortable and simple interface.

Clearly, to achieve these goals, some technical issues must be solved: i) Machine supply; and ii) How to distribute the available physical resources among these virtual machines which need them. Our proposal, called CICA GRID solution, is developed as follows.

3.1 Provisioning and management of large-scale virtual systems

The CICA GRID solution is a private cloud with 35 virtual machines that compose gLite’s working nodes and services. To make the management of this cloud easier, and more automated while maintaining control of the features and services in the production of each machine, it is essential to have a tool that enables easy and flexible administration of these machines. Hence, the problem of building the machines demanded by users has been resolved by using Cobbler/Koan [3].

This software implements a Linux server for machine installation which includes a set of easy commands. Furthermore, its configuration file allows the installation of virtual and physical machines by PXE and this server provides an
API which enables its integration with other systems. Cobbler/Koan can optionally be the manager of DHCP and DNS for new installed machines. This tool facilitates the provisioning of virtual machines according to the options given to the users at the time of selecting how they want to build their machines.

Hence, Cobbler establishes an object hierarchy which defines the configuration characteristics at the highest levels. These objects are, from the highest to the lowest level:

- **Distro**: Defines a kernel which has to boot the machine, the options with which this should boot, processor type (i386, x86_64, ...), and variables to guide the installation process.
- **Profile**: Defines the associated distro, the kickstart file that will guide the installation, variables whose values will be applied to the previous file, characteristics for a possible virtual machine that could be provided from this profile (CPU number, memory, virtualization type (Xen, KVM...). At this level, variables inherited from the higher level can be redefined.
- **Subprofile**: Modifications for an existing profile. The aim is to make the provision for the kinds of machines that share a lot of characteristics with other kind of machines, without having to make very similar profiles.
- **System**: Defines an installation scheme for a specific computer.

Node deployment is established by Cobbler/Koan which, by using a predefined image repository and a profile configuration, enables the creation of virtual machines in production as would-be needs and the physical environment allow. Once the user, using an interface built for this specific purpose, has decided which features (hardware and installed software between some allowed options) are wanted, a quickstart guide is built by means of Cobbler/Koan which provides these virtual machines. The relationship between objects that can be defined with Cobbler and the actual supplied machines are shown in Fig.1.

At the time of the installation of a computer, PXE boots the system while Cobbler serves as an expandable menu where the required installation type (profile, subprofile or system) can be set up. This step can be done automatically and the administrator does not have to choose the option manually. If a virtual machine is going to be supplied, then it is possible to use the Koan command over the physical machine to specify what kind of machine is needed.

In the private cloud, the CICA GRID solution, the user’s indications (or the staff responsible for the management of the NGI grid part) at moment of selecting the number of required machines and their features (CPU, memory, preinstalled software,...), initiate a guided installation through a Cobbler profile. In answer to this request, the designated virtual machines are kept in a shared space (machine repository, see Fig.2 where they are left available to OpenNebula for deployment.

However, the provision and deployment of virtual machines does not resolve all maintenance problems of the infrastructure: the daily administration of these systems remains unresolved. Hence, a system for automating system administration tasks is required, and Puppet [7] has been chosen. Puppet is based on:
A client-server scheme. Puppet server stores tasks that should be executed by clients. In our case, the clients are virtual machines which are built in the cloud.

A declarative language which specifies administration tasks that have to be applied by the clients. Although it is not necessary to specify how to carry out tasks in this language, it must be composed of normal concepts which are used every day, for example, node, file, and service.

Puppet is used in the CICA GRID solution to configure and ensure that the NTP service of machines works correctly, that users are authenticated by LDAP, and that a basic backup configuration, security updates and certain file systems are set up. Through Cobbler profiles, each newly supplied virtual machine has a Puppet client installed which is queried every thirty minutes, so that all system administration tasks can be applied.

It is worth noting that in our experience to date, both Cobbler/Koan and Puppet have proved to be capable of providing support for hundreds of machines, and depending on the required bandwidth and CPU consumption, these systems can be used on a larger scale.

### 3.2 Virtualized systems

Open Nebula has been chosen to solve the problem of finding a system for efficient deployment of virtual machine deployment over private Cloud Computing
infrastructure (IaaS) into the physical infrastructure in such a way that the users think that they are working with a Cloud Computing environment. This is an open-source virtual infrastructure engine that enables the dynamic deployment and re-placement of virtual machines using a pool of physical resources. OpenNebula has been chosen since:

- it works with different kinds of virtualization systems (Xen, KVM),
- it has the capacity to establish requirements that must be satisfied by physical machines to execute the work given to virtual machines,
- it is free software whose deployment is unattached to any manufacturer; and
- its source code is available.

Hence, the benefits of virtualization platforms from a single physical resource to a pool of resources, by decoupling the server not only from the physical infrastructure but also from the physical location [10], are obtained with OpenNebula.

Therefore, once Cobbler has provided the machines requested by the user and these have been saved in the repository, the system will build the needed files to enable OpenNebula to launch the deployment of these machines as shown in Fig.2.

Fig. 2. Architecture: Connection Scheme
4 User tool

The CICA GRID solution has a modular design to facilitate its development and has been implemented in the Python[16] language. Each module presents a well-defined interface so that it can be easily used by the other parts of the program.

**Cobbler module:** This module implements the supply of the machines and is responsible for: i) Defining profiles (profiles in Cobbler terminology), ii) Defining systems (specific machines that could be provided), iii) Providing machines: Creating VM from all the previous machines; and iv) Reporting on profiles and systems that a user has defined.

**DHCP and DNS management module:** This module has to assign IP and name data. The Cobbler system can be responsible for this automatically but needs DHCP and DNS servers to run on the same machine as Cobbler, since this is rarely the case, another module was built which is responsible for the connection of machines so these services run and modify the DHCP and DNS configurations.

**Puppet module:** Defines policies of system management for machines built by users.

**Open Nebula module:** Creates templates using the syntax that Nebula understands to carry out the deployment of the machine. This template is made from the configuration file that the configuration system of virtualization creates when carrying out the supply. Furthermore, this module communicates with Nebula to trigger the deployment of virtual machines.

**User Interface Module:** This is the main module of the application. It’s where the users interact with and is responsible for communication with all the other modules in order to translate user requests into low level instructions. Since multiple users can use the application at the same time some concurrency problems may arise. How many connections can be made to Cobbler to ask it to build a machine? What if there are simultaneous changes of configuration files of DHCP or DNS zones? The best solution appears to be the separation of the user interface into two blocks: one which speaks with each user while the other block serializes and sequentially executes each request from the users [1].

4.1 ReCarta

A minimalist approach, which attempts to show the user the least possible options, is required. Therefore, the main focus is not on writing less code, but providing the user with a useful system. This system is called Recarta. The machine creation is composed of only 2 steps: when a user wants to define a new group of machines, the user must first define hardware and software features. This is what causes ReCarta to create a new profile (profile) in Cobbler. In the second step, the user must indicate how many machines are to be defined with these characteristics and the names of each.

At the end of the process, the user data of the systems created is shown, along with the information needed for their connection and start-up. Therefore,
users have a project control panel at their disposal from which they can see what systems have been defined and they may initiate the deployment or stop the deployment if it is no longer needed.

4.2 Code example

A example of code is given in order to illustrate the set of calls to the API [6] defined by the different modules and how these calls carry out the task that a user has requested via the web interface. This example is an adaptation from the real code that is executed in the module which serves requests from the user interface. A new Cobbler profile (a new project in the terminology shown to users) is created that defines machines with 1 CPU, 512 MB for RAM with 4 GB for disk and Java language support (see\(^4\) Fig.3).

![imagenes/fig3.pdf](imagenes/fig3.pdf)

**Fig. 3.** A screenshot of the process

From this profile three systems are provided:

```python
# Import ReCarta modules
import mod_cobbler
import mod_dhcpDns
import mod_puppet
import mod_nebula

# Build a connection with Cobbler for the user John
miCobbler=mod_cobbler.Cobbler('john')
```

\(^4\) All text is in Spanish since it is the users’ native language.
The Cobbler profile for the machines has now been built. Physical characteristics for virtual machines are those requested by the user:

```python
miCobbler.setProfile({'nombrePerfil': 'proyecto1', 'kickstart': 'vm-kickstart.template', 'diskSize': '4', 'ram': '512', 'cpus': '1', 'comment': 'Perfil para el proyecto 1', 'software': ['X-WINDOW', 'JAVA-SUPPORT']})
```

Report to Cobbler that 3 systems will be provided with these characteristics. The method setSystems returns a list with pairs MAC/system name which has been defined in Cobbler. This information is used to associate an IP to the name and the MAC in the DHCP server:

```python
macAddressesNames = miCobbler.setSystems([{'nombre': 'proyecto1-vm1', 'comentario': 'VM1 del proyecto1', 'perfil': 'john-proyecto1'}, {'nombre': 'proyecto1-vm2', 'comentario': 'VM2 del proyecto1', 'perfil': 'john-proyecto1'}, {'nombre': 'proyecto1-vm3', 'comentario': 'VM3 del proyecto1', 'perfil': 'john-proyecto1'}])
```

Once systems are defined, the DHCP server configuration is modified to associate these MACs and IPs:

```python
mapIpNames = mod_dhcpDns.addSystemsDHCP(macAddressesNames)
```

Now the mapping between IPs assigned in the previous step and its names in a DNS server is carried out:

```python
mod_dhcpDns.addEntryDns(mapIpNames)
```

Now the machines are provisioned:

```python
miCobbler.provisionSystems(['john-proyecto1-vm1', 'john-proyecto1-vm2', 'john-proyecto1-vm3'])
```

It is important to look at the two ReCarta [14] design features that are fully deliberated: one is the high level abstraction offered by the different methods which make up ReCarta. Details, about how the template of the kickstart is modified to adapt it to the different requirements of users, appear nowhere, neither how the configurations of DHCP/DNS servers are modified to assign a place in the network to new systems, nor how they interact with OpenNebula to deploy the new systems, nor the numerous other details necessary to make the requested machines of users work.

The other feature is that ReCarta use no database system to save the information about which projects have been defined, which users have defined them, etc. What ReCarta does is to put the username as a prefix to data of profiles and also a project name to names of systems defined by users. This information is saved in Cobbler. When a user needs information about projects and systems, there are methods in the module to interact with Cobbler that query that information, parse and show it to the user. In the same way, the basic monitoring information about working systems is extracted by directly consulting to OpenNebula. This design decision has been taken in order to keep ReCarta as simple as possible.
5 Benchmarks and features

Nowadays, ReCarta creates systems which are composed of virtual machines of paravirtualized Xen. Although OpenNebula supports the launching of virtual machines over other virtualized systems such that KVM or VMWare [8], this capacity is not implemented in ReCarta. Xen [7] has been used to date, since it is proved that a paravirtualized virtual machine only loses 5-10% of CPU performances with respect to the equivalent physical machine.

Table 1 presents the results of the execution of benchmarks HPCC [9], while Table 2 and Table 3 show the results of the execution of Bonnie++ [10] for memory of a virtual machine and that of equivalent physical machine. In these latter cases, a significant decrease in the performance between the virtual machine and the physical machine can be seen for the action of writing to disc.

<table>
<thead>
<tr>
<th></th>
<th>Intel 6400 Physical</th>
<th>Intel 6400 Virtualized</th>
</tr>
</thead>
<tbody>
<tr>
<td>PTRANS(GB/s)</td>
<td>0.65</td>
<td>0.54</td>
</tr>
<tr>
<td>HPL(Gflops)</td>
<td>14.26</td>
<td>13.01</td>
</tr>
<tr>
<td>MPI Latency(ms)</td>
<td>0.00043</td>
<td>0.00053</td>
</tr>
<tr>
<td>MPI Bandwidth</td>
<td>1471.17</td>
<td>1477.64</td>
</tr>
</tbody>
</table>

Table 1. HPCC Benchmarks

<table>
<thead>
<tr>
<th>Server type</th>
<th>Sequential Output</th>
<th>Sequential input</th>
<th>Random</th>
</tr>
</thead>
<tbody>
<tr>
<td>Virtual</td>
<td>14014 K/sec</td>
<td>34851 K/sec</td>
<td>150.7 Seeks/sec</td>
</tr>
<tr>
<td>Physical</td>
<td>45678 K/sec</td>
<td>49719 K/sec</td>
<td>64.7 Seeks/sec</td>
</tr>
</tbody>
</table>

Table 2. Bonnie++ test Write - Blocks : 2Gb

<table>
<thead>
<tr>
<th>Server type</th>
<th>Sequential Create</th>
<th>Random Create</th>
</tr>
</thead>
<tbody>
<tr>
<td>Virtual</td>
<td>0.0000 sec</td>
<td>0.0000 sec</td>
</tr>
<tr>
<td>Physical</td>
<td>1621 sec</td>
<td>891 sec</td>
</tr>
</tbody>
</table>

Table 3. Bonnie++ test Create - Blocks : 1Gb

Furthermore, the CICA GRID solution is a private cloud with 35 virtual machines that compose gLite’s working nodes and services, with the following physical (virtualized) features: 1 core and 1 GB RAM. To virtualize, 6 physical servers are used with the following features: 2 cores and 4 GB RAM.
In Table 4, it can be seen that the use of virtualization allows the power consumption to be reduced to 39%. Thus, from a point of view of consumption, the CICA GRID solution is a good alternative.

<table>
<thead>
<tr>
<th>Type</th>
<th>Physical</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Physical</td>
<td></td>
<td>7791 KWh/year</td>
</tr>
<tr>
<td>Virtualized</td>
<td>516 KWh/year</td>
<td>3096 KWh/year</td>
</tr>
</tbody>
</table>

**Table 4.** Consumption of 6 physical nodes and of 35 virtual nodes

### 6 Conclusions

Although the CICA GRID solution is still in its experimental phase, some case studies have been carried out: the creation of a small virtual cluster to be used with Hadoop, and the building of virtual machines to be used as working nodes to temporarily amplify our participation in the NGI grid. In both cases, the project objectives were achieved since this solution has enabled more job requests to be served that exceed the normal workload for a cluster expected for HPC with an LRM and this demand has been attended to quickly and with a minimal effort of technical staff.

Furthermore, from the point of view of the energy-saving involved in virtualized environments, the CICA GRID solution renders research advances less expensive for the research groups who collaborate with CICA by releasing electrical consumption and everything that implies.

It should be kept in mind that cloud technology currently involves some drawbacks in its grid applications, which implies that it is unlikely that cloud technology could replace or could be the exclusive method for building resources in the HPC. Nevertheless, we believe that it is a valuable tool for certain types of jobs. Thus, we have learned, during the launching of our pilot project of Cloud Computing: the CICA GRID solution, that our users appreciate these two advantages: i) The illusion of having a huge computing resource reserved exclusively for them; and ii) The possibility of increasing and decreasing the resources according to their needs.

Furthermore, Recarta is designed to supply RedHat-based systems. This has enables Cobbler (albeit to a limited extent) to support other distributions, specifically Debian. The Cobbler development roadmap is expected to provide better support for these distributions. Therefore Recarta can also improve the ability of these distributions to supply systems.
Acknowledgements

This research is supported by the Spanish Ministry of Science and Innovation R&D project ARTEMISA (TIN2009-14378-C02-01).

References

3. Available from: https://fedorahosted.org/cobbler/ "Cobbler is an install server; batteries are included", Online, (2008)