ICEAGE Grid in a Box

Using virtualization to setup gLite training infrastructures

Author:

Paulo Ricardo Motta Gomes
Paulo@lsd.ufcg.edu.br

Supervisors:

Marc Ozonne
Marc.Ozonne@cern.ch

Erwin Laure
Erwin.Laure@cern.ch

CERN Openlab Student Programme 2007

1. Introduction

Grid education enables researchers to take more advantage of the immeasurable grid potential by providing them up-to-date and qualified information on the usage of grid tools and services. Besides spending less time searching for support information, well-trained scientists are able to enhance the productivity of their experiments by using acquired knowledge on advanced grid middleware features.

The International Collaboration to Extend and Advance Grid Education (ICEAGE) organizes several training schools and tutorials on grid computing, in order to incorporate grid education in academic courses throughout the European Research Area [1].

ICEAGE currently uses INFN's GILDA e-infrastructure on the grid training schools. The GILDA remote testbed is a series of grid sites and services spread all over the world and mainly based on EGEE's middleware gLite [2]. Since the GILDA online laboratory is ready to use, it's relatively simple to setup a grid school infrastructure using GILDA.

Nevertheless, the testbed can be suddenly overloaded with job submissions or go down because of a power cut. In that sense, it is safer to have local backup grid services. In this work we present a quickly deployable solution that reinforces the availability of the overall grid training infrastructure.

2. Concept

Setting up a gLite site from scratch is not a straightforward task. The site administrator has to install required libraries, edit configuration files, setup hostnames and IP addresses, install certificates for most of the services, synchronize the nodes, deal with unexpected installation problems, test the system intensively, etc. Moreover, once the site is installed it is quite difficult to move the snapshot of the system to other machines without having to do some complicated reconfiguration, sometimes even harder than installing the system again.

Having a set of pre-configured, plug-and-play, easy to transport, lightweight grid services is very convenient and suitable for grid education.

Trainers could always have a local backup grid infrastructure that they could control, and restart in case of problems; trainees could have a global view of a grid site and see, in concrete terms, how the different grid services interact together.

2.1. Grid in a Box

The idea is very simple: using virtualization technique we create in one physical machine one virtual machine (VM) per grid service, deploy the services and configure a working grid environment, save the state of the machines, and finally distribute the package with the virtual grid that can be deployed universally; the Grid in a Box.

The concept is not brand new; a few "grid in a box" projects have been developed so far, but each one with different objectives and results. The first known work on this field was lead by Trinity College Dublin, "A single-computer Grid gateway using virtual machines", and aimed at reducing hardware costs of an LCG site by running the gateway services (UI, CE, SE, Install Server) in one machine using virtualization, in order to increase the number of sites and, consequently, the grid accessibility in Ireland research institutes [3].

Our project aims at creating an easily deployable and lightweight virtual gLite site for grid education, without any special configuration needed by the school administrator. Even though there are services that deliver virtual machines with separate gLite grid services for customization, no similar gLite plug-and-play "Grid in a Box" was found [4].

2.2. Virtualization

Operating System Virtualization is basically a layer of software between the OS and the underlying server hardware. This layer manages the multiple OS instances, by scheduling resources of the single server. Each host OS believes that it has the resources of the entire machine under its control, but underneath it the virtualization layer (also known as the Hypervisor or the Virtual Machine Monitor) transparently ensures that the resources are being properly shared [5].

The virtualization support of our project will be supplied by the Xen Hypervisor. Besides being open source, Xen performs well enough to be indistinguishable from a real machine, according to the benchmarking realized by the Trinity College grid virtualization project [3].

Xen uses the para-virtualization technique. That is, the guest OS is ported to an idealized hardware layer which completely virtualizes all hardware interfaces. When the OS updates hardware data structures, the Hypervisor API is called by the modified OS device drivers. This allows the Virtual Machine Monitor (VMM) to keep track of all the changes made by the OS, and to optimally decide how to modify the hardware in any context switch [5].

The Xen host machine is called dom0 (or domain zero), and it has this name because it's the first domain started by the Xen hypervisor on boot. This domain has special privileges, like direct hardware access and rights to create and destroy VMs. The unprivileged domains (or domUs) are the counterpart of dom0; they have no direct access to the hardware; their system calls are all made to the hypervisor's API. Both domains have the kernel patched with code provided by Xen developers.

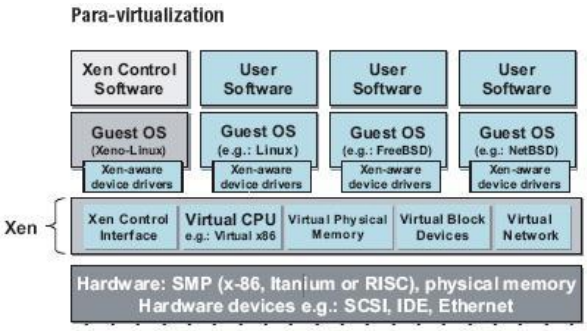

Diagram 1 – Xen para-virtualization

3. Architecture

In the first version of the project we will deliver a simple site in a box containing the following services: LCG-CE, gLite-WN, gLite-SE, and gLite-UI. The LCG-CE was chosen instead of the gLite-CE because it's considered more stable. In that sense we will need four different Xen virtual machines, each one running a separate grid service.

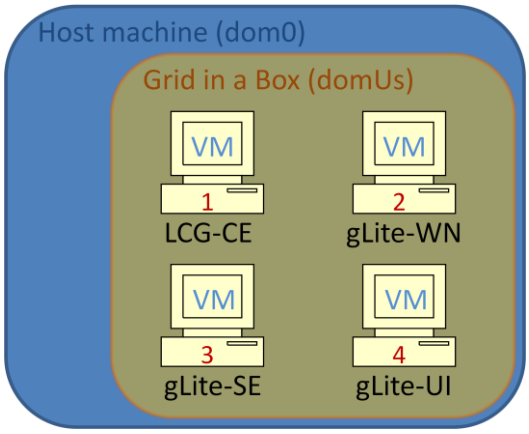

Diagram 2 – Grid in a Box overall architecture

3.1. Networking

The first big challenge of this project arose when dealing with networking. How to make a portable grid system that will not be affected by the change of network environment of the host machine? How to ensure that inbound/outbound access will always stay up on the guest domains, independent from where the host machine is located? After all, networking is a vital point of the grid, and it's desirable that the system works in any type of network without any extra effort from the system administrator.

Private Networking

Allocating public IPs to the virtual grid machines is not practical because for every IP address change, many system reconfigurations would be needed on the guest nodes on each installation. Moreover, the same would happen if the gateway or hostname changes. It's not interesting that administrators spend their time reconfiguring VMs.

To add maximum portability to our system we have to ensure that the entire networking configuration of the site is static, that is, it won't change when the host machine is changed. By using a private networking schema between the VMs, we make sure that the virtual machines have the same IP addresses in every host machine. That means they only need to be configured once, during the creation phase.

Outbound access

Private network ensures that the machines can communicate between themselves and the host, but how are the grid machines making requests to the outside world? After all, it's essential that a grid site can communicate to external services, or download external input. To ensure outbound connectivity on the private network without affecting the nodes static configuration we use the host machine as a gateway to the guest machines. As the host machine will be on the private network, the gateway address will also be static.

The IP Masquerading technique enables the host machine to act as a router from the internal network to the Internet. It uses Network Address Translation (NAT) to stamp the packets coming from the internal network to the Internet with the public IP of the host machine, and to forward back the responses to the private IP machine that made the request.

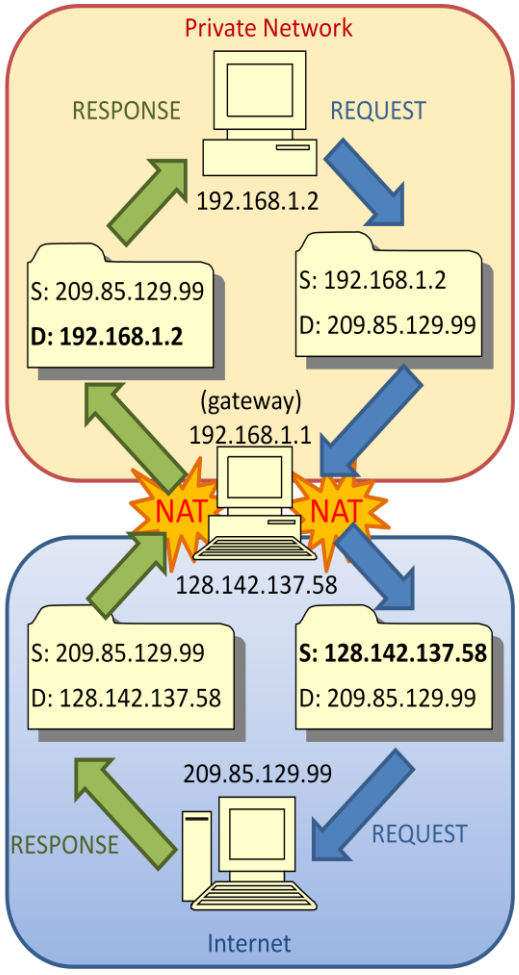

Diagram 3 – Packet flow when using IP masquerading technique to route the packets from the private network to the Internet. The bold IP addresses represent the fields that were changed by NAT on the gateway (host machine).

Inbound access

The network is now encapsulated in a box. The machines can communicate between themselves and can send requests to the outside world. It is possible to change host machines without having to reconfigure the guest machines networking. But there is still one network issue left: how can external users and services have access to the virtual grid?

Having private IP addresses means that the VMs cannot be accessed directly from the Internet. The only external access point to the virtual network is the public IP address of the host machine.

A good way to reach machines inside a private network through a single-IP public gateway is by Port Forwarding (or tunneling). This technique consists on forwarding a network port from one node to another, most of the times from one public node to a private node. In our case this technique allows network ports that are not being used on the host machine to be forwarded to the virtual machines. In that way, remote grid services running on the private machines can be reached from the outside world by the IP address of the host machine, through a specific port.

SERVICE*	FROM	TO
CE(JM)	HOST_IP:2119	192.168.1.2:2119
SE(RFIO)	HOST_IP:3147	192.168.1.4:3147
UI(SSH)	HOST_IP:2222	192.168.1.5:22

Table 1 – ICEAGE Grid in a Box port forwarding table

*Legend:
JM – Job Manager
RFIO – Remote File Input/Output
SSH – Secure Shell

The table above shows the port redirections that are present in the first version of this project (customizable).

If a user wants to use the Computing Element or the Storage Element from outside, then he/she should use the IP of the host machine as the service address and every message using these services default port (2119 and 3147) will arrive to the correct destination. On the other hand, if we forward the default SSH port (22) of the host machine to the User Interface machine, we will block SSH access to the host, because each message arriving on SSH port 22 of the host would instantly be forwarded to the User Interface. That is clearly the biggest limitation of the port forwarding schema. If the same port is being used simultaneously in more than one machine (host or guest), only one can receive messages from the outside world at a time. For now we don't see this as a critical problem, as most of the services we are currently using listen on different ports.

To allow SSH access to the UI without affecting SSH access to the host machine, we forwarded the port 2222 of the host machine to the port 22 of the UI. One can simply access the UI machine by typing *"ssh user@HOST_IP -p 2222"*.

Hostname certification

All the gLite service nodes (except UI, WN and BDII) require host certificates to work. These certificates are issued by a Certification Authority (CA) and are attached to the full hostname of the grid machine. Before any connection between the grid elements is established, some authentication is performed and it's verified if the hostname in the certificate resolves to the IP address of where the message is coming from. As the messages coming from the virtual grid machines are being stamped by NAT with the host machine's public IP address, the hostname of those machines should point to this IP address in order to be authenticated and certified.

If the hostnames of the VMs were bound to the host machine's network

domain, every time the location of the grid changed the hostname of the machines would need to be updated. That is not a practical solution to a plug-and-play grid because every time the hostname changed, new host certificates would need to be requested and the grid services would need to be reconfigured.

Keeping the configuration static on the grid nodes is essential to ensure the portability of the box. Providing hostnames to the VMs that are resolved to the host machine's IP address is necessary to keep the gLite authentication functional. In that sense, the hostname assigned to the nodes should be independent from the host machine's domain while still pointing to that machine's IP address.

We found a useful DNS service on the web where the administrator can create hostnames, and change the IP addresses associated with them at anytime, free of charge. The service is hosted on www.no-ip.org [6] and one account was created there for this project.

For this first version we created four hostnames: {xencenode, xenwnnode, xensenode, xenuinode}.sytes.net, and they need to be updated every time the host machine changes. This will ensure that there will be no authentication problems.

Host:	IP/URL
sytes.net	
xencenodes.sytes.net	128.142.137.56 [IP]
xenwnnode.sytes.net	128.142.137.56 [IP]
xensenode.sytes.net	128.142.137.56 [IP]
xenuinode.sytes.net	128.142.137.56 [IP]

Table 2 – www.no-ip.org redirection table; the IP to which the hostnames are pointing are the IP address of the host machine. Every time the host machine is changed, this table needs to be updated in order to keep the gLite host certificates valid.

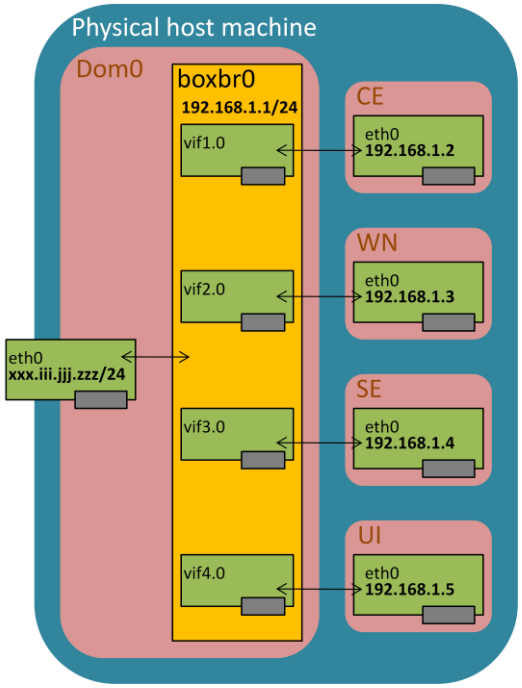

Diagram 4 – Grid in a Box Network Architecture

The hostnames {xencenode, xenwnnode, xensenode, xenuinode}.sytes.net point to the IP *xxx.iii.jjj.zzz*.

4. Implementation

To implement the solution we used a cluster node with double Intel Xeon 2.8GHz processors, and 2GB RAM memory. The operating system of the machine was Scientific Linux CERN SLC release 4.5, with Xen patched Linux Kernel 2.6.18-8.1.3.slc4xen, and Xen version 3.0.3.

4.1. Xen network script

For most configurations, Xen provides network scripts that perform the necessary steps to setup the desired networking schema for the host and the guest machines. There was no Xen default network script that attended all the networking needs of the Grid in a Box. In that way we decided to create our own script to setup the network configuration described on Diagram 4.

The *network-gridbox* script initially creates the bridge *boxbr0* on the dom0, and assigns the IP range 192.168.1.1/24 to it. When each domain is created, Xen

automatically adds the corresponding virtual interface to this bridge, through the *vif-bridge* Xen default script. In each domain, we set the default gateway to be the IP of the bridge (192.168.1.1). This bridge configuration guarantees the private networking between the virtual machines and the host.

After the private networking is set, next is to provide outbound/inbound connection to the outside world through the IP masquerading and port forwarding techniques, described in the last section. That is achieved through a set of iptables firewall rules. First, we enable IP forwarding on the dom0, and add the firewall rule that performs the IP masquerading to the NAT POSTROUTING chain. Finally, we read from a user-defined configuration file which ports of the host machine should be forwarded, to where, and add those rules to the iptables NAT PREROUTING chain.

4.2. Virtual grid deployment

Once the network is up, it is possible to start deploying the virtual machines and the gLite services. To start, we downloaded an image of a Scientific Linux CERN 3.0.8 standard installation and replicated it to several LVM partitions, one to each host machine (CE, WN, SE, and UI). Afterwards we created one LVM swap partition to each node, each one with 256MB swap space. Then, we created a standard Xen configuration file for each virtual machine:

```
kernel = "/boot/xen/vmlinuz-
2.4.21-47.0.1.EL.cernxenU"
ramdisk = "/boot/xen/initrd-
2.4.21-47.0.1.EL.cernxenU.img"
name = "CEnode"
memory = 256
disk = [ 'phy:vg/CEnode,sda1,w',
'phy:vg/CEswap,sda2,w' ]"
vif = [ '' ]
ip = "192.168.1.2"
gateway = "192.168.1.1"
hostname = "xencenode.sytes.net"
extra = "fastboot"
root = "/dev/sda1 ro"
```

With the configuration files ready, the virtual machines can be booted from dom0, using the Xen management tool, and be used just as physical machines. From this step on, we can install the gLite components as we would normally do on ordinary nodes.

Before starting installation, it is necessary to get the host certificates for the Computing Element and the Storage Element. As our site will be using GILDA CA for authentication, we requested two host certificates: one for xencenode.sytes.net, and another one for xensenode.sytes.net. The User Interface and the Worker Node don't need host certificates.

To install each gLite service on the nodes we used the YAIM installation and configuration tool. YAIM is basically a set of scripts and configuration files that simplifies the installation process of gLite. We first created a YAIM site configuration file (site-info.def), gathering all the necessary information to the Grid in a Box site, and then copied this file to each node, and installed each service based on this same file.

Before starting the installation we had to make sure that the host machine was time synchronized (NTP). The guest machines are set with the clock of the host, and it's essential that they are synchronized to make the gLite services work properly.

User Interface

The first node to be installed was the User Interface, in order to be able to test the installation of the other elements. The version installed was glite-UI-3.0.22-2.

The installation went smoothly with no major problems. We basically installed YAIM, then installed the GILDA VO rpm, and finally configured the node through YAIM configuration tool.

Computing Element

The installation of the Computing Element was the most time consuming of all the services. We decided to install the lcg-CE_torque-3.0.13-2 since it is a stable version of the CE. Initially the installation was exactly done like the UI installation (YAIM, GILDA VO rpm and YAIM config). But after many debugging we found out that the certificates of the GILDA CA were missing, what was making the submitted jobs to fail. We could have saved some time on this issue if we had clearer gLite error messages. After installing the CA certificate the CE worked perfectly, and we could successfully submit jobs from the UI.

Worker Node

The installation of the Worker Node went well, apart from some configuration errors on the PBS client, which was making the node always busy for the Computing Element scheduler. After fixing the load properties on the pbs_mom configuration file (loadave and netload), the Worker Node finally appeared on "free" state, what allowed job submission from the UI. We installed glite-WN-3.0.22-2.

Storage Element

The Storage Element installation was the most straightforward. After installing all the other components, we were aware of the main issues that could happen on a gLite installation. In that sense, there was no problem when installing the glite-SE_classic-3.0.15-2. After that, we were able to successfully submit files from the UI to the SE. Finally the Grid in a Box deployment was finished.

4.3. Deployment tool

We finally have the Grid in a Box up and running in one machine, but still, it's not portable and generic. Creating a package with the necessary installation files and an installation tool that will transparently install and make sure the system will work on any machine is vital for this project success. We designed a few scripts that will make automatic the deployment process of the Grid in a Box package. More information on their usage and configuration will be given on the next section.

5. User Guide

You will now be guided through the deployment process of the Grid in a Box project package. After following the steps presented in here you should be able to have a grid in a box instance up and running.

Requirements

To install the Grid in a Box system you will need the following requirements:

- 22GB of free disk space, for the installation files or 22GB of free disk space on a separate partition, if you wish to use LVM partitions for the virtual machines file systems.
- 1280MB RAM memory; 256MB for each VM + 256MB for the host.
- Internet connection with a public IP address.
- Xen Hypervisor installed and running
- Root access, for modifying system configuration files.

Additional Notes

- It is assumed that the host machine time is synchronized with a time server. A more than 5 minute synchronization delay between the gLite virtual nodes and the GILDA servers will affect the communication between the services.
- To avoid conflicts between the firewall rules of the Xen network script and the system rules, it is strongly advisable to disable the firewall of the host machine in order to ensure network connectivity to the box.

Installation package

ICEAGE Grid in a Box is distributed in Gunzip compressed package: icebox.tar.gz. This file has 7GB, and after unpacking will reveal the following directory structure:

```
icebox/
…boot/
……initrd-2.4.21-47.0.1.EL.cernxenU.img
……vmlinuz-2.4.21-47.0.1.EL.cernxenU
…images/
……CEnode
……SEnode
……UInode
……WNnode
…conf/
……network-gridbox
……gridbox-ports.cfg
……xend-config.sxp
…icebox
…icebox.properties
…setup.sh
```

To start the installation, execute *"setup.sh"* script. A screen will be prompted and the install options will be displayed.

Configure XEN

The first of the install options will automatically configure Xen environment to support ICEAGE Grid in a Box. Initially the necessary configuration files are copied to the Xen configuration directory (by default */etc/xen*). Then it is asked to the administrator if he wants to restart the Xen daemon. Restarting the Xen daemon is essential to make the system work, but it can only be done if the user is using a terminal on the machine. Otherwise, when the networking is restarted, any open connection is closed, including the one the admin is using to access the machine. In that case the best way to activate the changes is by restarting the machine remotely through the *"reboot"* command.

Port forwarding

The default port forwarding enabled by the installation script is described in the table 1. In order to forward additional ports from the host machine to the virtual machines, the */etc/xen/scripts/conf/gridbox-ports.cfg* file should be edited, and new entries should be added using the following syntax: *host_port>dest_ip:dest_port*.

Install using LVM partition

Using a separate partition to mount the virtual machines file systems improves considerably the guest machines disk access performance. It is possible to deploy the Grid in a Box on an empty partition using LVM by selecting the second option of the install menu. It will be asked to the user in which partition should the logical volumes be created. After selecting the partition, a volume group is created on that partition, and one logical volume is created to each VM file system. Once the virtual disks are created, the images are copied to the logical volumes. The virtual grid file systems are now ready to be mounted on the virtual machines.

Finally, each machine custom configuration file is created, on the "nodes" directory of the installation root. The system now is fully installed and ready to be executed.

Install using Image files

Another option to setup the Grid in a Box is using image files as the virtual machines file systems. The overall performance of the machines is slightly reduced, but on the other hand, an extra partition is not needed. The only action performed by this installation step is to generate the Xen VM configuration file for each machine on the nodes directory of the installation path. The image files used on this configuration files come on the installation package.

Icebox management tool

An extra script is provided on the package to support the easy management of the Grid in a Box virtual machines. In the first version we only provide the start and the stop commands, but we intend to add more management functionalities on later versions, like status monitoring and automatic update. The syntax is very simple: to create all the virtual machines, the administrator should type *"./icebox start"*, to shutdown all the machines the command is *"./icebox stop"*.

Testing installation

After the system is deployed using the installation scripts and started through the icebox tool, it is advisable to test the installation by running some commands on each virtual machine, testing each of the grid services. Once the machines are started, they can be accessed through "xm console <id>" command, where <id> can be retrieved from "xm list" command. They also can be reached through SSH from the host machine. Remember to login as root, to run the tests successfully.

- User Interface test:

First, create a user account and copy your GILDA user certificate to it:

```
adduser testuser

su - testuser

scp -pr user@machine:~/.globus ~
```

Now start the proxy, by typing:

```
voms-proxy-init -voms gilda
```

Now try the following tests:

```
lcg-infosites --vo gilda ce

lcg-infosistes --vo gilda se
```

If the information is displayed correctly and no error message is shown, the user interface connectivity is working fine. Some other tests will be performed on the UI when testing the other services.

- Computing Element test:

Still from the User Interface with an active proxy, type the following commands:

```
globus-job-run xencenode.sytes.net /bin/hostname
```

If the hostname *"xencenode.sytes.net"* is displayed as the job result, the CE is able to run remote jobs. Now let's test if the LDAP information of the CE is being correctly published. Type from the UI:

```
ldapsearch -x -H ldap://xencenode.sytes.net:2170 -b mds-vo-name=local,o=grid
```

If the information on the CE displayed, then the LDAP server on the CE is OK.

- Worker Node test:

From the Computing Element, first change the user:

```
su - gilda001
```

Then, try the following command:

```
qsub –I
```

If an interactive shell on the Worker Node is opened, the system works fine. Now let's see if the job retrieving is working fine from the CE to the WN. Type from the CE:

```
echo "hostname; sleep 20" | qsub –q
short
```

Now let's test if the CE is scheduling the job correctly to the WN. From the UI, with a valid proxy, type:

```
globus-job-run
xencenode.sytes.net:2119/jobmanager-pbs
-q short /bin/hostname
```

Finally let's submit a gLite job from the UI to the CE/WN. First create a simple Hello World jdl:

```
vi hw.jdl

Executable = "/bin/echo";
Arguments = "Hello World";
StdOutput = "hw.out";
StdError = "hw.err";
OutputSandbox = {"hw.out", "hw.err"};
VirtualOrganisation = "gilda";
```

Then submit it to the CE:

```
glite-job-submit –r xencenode.sytes.net
hw.jdl
```

Check the status of the job with glite-job-status and afterwards retrieve the output with glite-job-output. If the output was correctly retrieved the system is working fine.

- Storage Element test

To test the installation of the SE, please login an account with a valid proxy on the UI, and type:

```
edg-gridftp-ls --verbose
gsiftp://xensenode.sytes.net/storage

edg-gridftp-mkdir
gsiftp://xensenode.sytes.net/storage/myd
ir

globus-url-copy file:/bin/hostname
gsiftp://xensenode.sytes.net/storage/myd
ir/myfile
```

If the operations above are executed successfully, the SE is OK. In order to submit files using LFC, the SE needs to be registered on the GILDA BDII. When the SE xensenode.sytes.net is registered on the BDII, execute this command on the UI:

```
lcg-cr –v --vo gilda –d
xensenode.sytes.net –l
lfn:/grid/gilda/mozonne/testfile
file:/bin/hostname
```

Check if the file was transferred successfully, and now the system should be ready to use.

6. Conclusion

Providing a transportable, easy deployable and functional grid infrastructure package was the main objective of ICEAGE Grid in a Box project. This objective was achieved using virtualization technology, provided by open source project Xen.

Grid schools have now available a backup grid infrastructure that can be deployed locally. That means that a training session is now able to happen using local resources without the need to deploy a whole infrastructure from scratch.

Improvements and next steps

Even though the objective of providing a portable grid infrastructure was achieved, yet we do not have the ideal solution. Now there is no need for a real gLite site to organize a grid school, but our virtual solution still relies on some GILDA central services to work. That means that if these services go down, our virtual site will not be able to work. Providing a fully independent site, with virtual central services (BDII; RB; etc) would be something to work on the next versions of this project.

Keeping the configuration static was essential in this version of the project to ensure the portability of the system. On the other hand, hard-coded properties make hard to have multiple instances of the Grid in a Box. For example, we only have a set of hostnames, what makes impossible to have two virtual sites running at once. Creating a new set of hostnames would mean changing many configuration properties on each service. In that sense it would be interesting to develop a tool that could easily change properties of the site while making the system still functional and transportable.

7. References

[1] http://www.iceage-eu.org/

[2] http://gilda.ct.infn.it/

[3] A single-computer grid gateway using virtual machines – Childs, S.; Coghland, B.; Oapos; Callaghan, D.; Quigley, G.; Walsh, J. – Advanced Information Networking and Applications, 2005. 19th International Conference on – Volume 1, 28-30 March 2005 Pages 310-315

[4] http://twiki.cern.ch/twiki/bin/view/virtualization/GLite

[5] Xen and the art of virtualization – Bahram P.; Dragovic B.; Fraser K.; Hand S.; Harris T.; Ho A.; Neugebauer R.; Pratt I.; Warfield A. – Operating System Principles, 2003, 19th ACM Symposium on – Pages 164-177

[6] http://www.no-ip.org/