Building a Virtual Globus Grid in a Reconfigurable Environment - A case study: Grid5000

Alexandru-Adrian Tantar, Nouredine Melab, Christophe Demarey, El-Ghazali Talbi

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Building a Virtual Globus Grid in a Reconfigurable Environment A case study: Grid5000

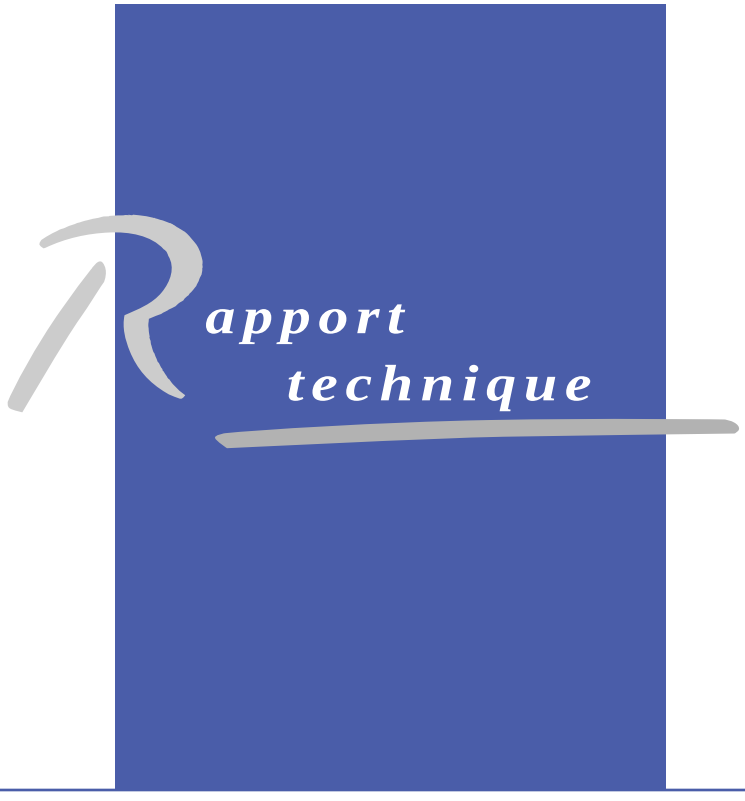
A. Tantar — N. Melab — C. Demarey — E.-G. Talbi

INRIA DOLPHIN Project Team, OPAC LIFL

N° ????

August 2007

Thème NUM



ISSN 0249-0803 ISRN INRIA/RT--????--FR+ENG

Building a Virtual Globus Grid in a Reconfigurable Environment
A case study: Grid5000

A. Tantar, N. Melab, C. Demarey, E.-G. Talbi
INRIA DOLPHIN Project Team, OPAC LIFL

Thème NUM — Systèmes numériques
Projets DOLPHIN

Rapport technique n° ???? — August 2007 — 21 pages

Abstract: With the continuous evolution of distributed computing grids and with the perpetuous development of the available computing resources and protocols, there is a *sine qua non* requirement to pass beyond the physical design of the grids. A viable solution is offered by virtual grids, having the advantage of flexible mapping and adaptation to live in-place resources. A software image is proposed, built with the use of the Globus Toolkit, the herein document describing the construction and configuratin phases as well as the deployment protocol in a live grid - Grid5000.

Key-words: virtual grids, reconfigurable grids, Globus Toolkit, distributed computing

Unité de recherche INRIA Futurs
Parc Club Orsay Université, ZAC des Vignes,
4, rue Jacques Monod, 91893 ORSAY Cedex (France)
Téléphone : +33 1 72 92 59 00 — Télécopie : +33 1 60 19 66 08

Building a Virtual Globus Grid in a Reconfigurable Environment
A case study: Grid5000

Résumé : With the continuous evolution of distributed computing grids and with the perpetuous development of the available computing resources and protocols, there is a *sine qua non* requirement to pass beyond the physical design of the grids. A viable solution is offered by virtual grids, having the advantage of flexible mapping and adaptation to live in-place resources. A software image is proposed, built with the use of the Globus Toolkit, the herein document describing the construction and configuratin phases as well as the deployment protocol in a live grid - Grid5000.

Mots-clés : virtual grids, reconfigurable grids, Globus Toolkit, distributed computing

1 Introduction

With the continuous development of highly reconfigurable grids and with the continously expansion of interconnected computing resources, virtualization and generalization over physical boundaries represents a *conditio sine qua non*. As constant change represents an immutable characteristic of nowadays grids, the design of a distributed computing system has to envision for scalability beyond the initial specifications. In the herein document we present the basic concepts of virtual grids, describing the steps to be followed for constructing and deploying software operating system images in live physical grids.

The core of the exposed virtual grid design is modeled by employing the Globus Toolkit[1] which is the current *de facto* standard for virtual grids development. In addition, the proposed design had as first target and as a test case exemplification, the deployment and execution of MPICH-G2[2]/Globus based applications. The proposed software image offers out of the shelf the ground base for the development of Globus and MPICH-G2 based applications. As case study we considered a live nation-wide distributed computing system - Grid5000[3].

As it will be later exposed in the following sections, Grid5000 is a highly reconfigurable grid with a large spectrum of methodologies and applications. The development and the evolution of Grid5000 stands as a combination of design patterns set in place (a) to overcome the limitations observed in simulators, emulators and real platforms and (b) as a result of the Grid community research topics investigations [1]. Furthermore, Grid5000 comes to meet user-centered specific requirements - setting up a virtual grid allows for the creation of a completely customized environment.

Extending the certainity of future multiple interconnected grids to the present, the advantage of having an on demand deployable environment represents the key for adaptibility and flexibility. Supposing a scenario under which it is required to connect Grid5000 with several different grids, even simultaneously, virtualization stands as an efficient and rapid solution.

While addressing a real specific case study, the exposed design elements and application target scalability issues, having in mind the imminent expansion towards multiple different interconnected grids. Surmounting physical limitations allows for the extension of available features, at the cost of constructing the underlying infrastructure. As this step, defining the basis for further development, represents a complex task to accomplish, a ready to use minimalistic environment is proposed in this document. The goal of the constructed environment is to offer the means for facilitating the mapping of virtual distributed computing systems over live ones. For further more complete details on the presented concepts and technical aspects refer to the Globus dedicated website (http://www.globus.org) as well as to the following documents: [2], [3], [4], [5], [6].

The following sections present the main steps to be considered when building a Globus virtual Grid. In Section 2 the main Globus related concepts are presented along with a

[1] http://www.globus.org
[2] http://www3.niu.edu/mpi/
[3] https://www.grid5000.fr

short introduction to grid infrastructure considerations from an architectural and security point of view. In addition, several Globus components are discussed - only the minimum detail has been considered, focusing only on the main required components. Section 3 offers a closer view on the technical elements required for setting up and interconnecting the presented components, starting from scratch. All the necessary steps are presented, starting from environment configuration, required packages, installation, compilation, etc. Security aspects are also considered, offering a brief example for generating and installing digital certificates. Section 4.1 includes a short description of the considered case study - Grid5000. A brief discussion is made on user policies, logical and physical environment, reconfigurability aspects, deployment, etc. The Globus deployment protocol is discussed in Section 4.2 - the deployment procedure is presented in a step-by-step manner. The final part of the section also presents deployment tests on five Grid5000 clusters as well as a broadcast messaging test with a number of processes of up to 300. In the end a few short conlusions are presented in Section 5, including references to future Grid5000 collaboration projects.

2 Globus Toolkit

The Globus Toolkit represents the outcome of an effort to construct the underlying support for service oriented distributed applications. As illustrated in Figure 2, the architecture of the Globus Toolkit includes several different levels, elementary entities for distributed systems being implementing as software components. Consecrated standards were adopted to stand as a base for designing security enhanced components, dealing with authentication, authorization and access control, encryption and data integrity - the main axes of the Grid Security Infrastructure (GSI).

The Globus Toolkit is constructed around the concept of interoperability having as backbone Web Services, authorization and security standards - Web Services Interoperability, SAML, XACML, SSL, X.509 digital certificates, etc. A schematic overview comprising the Globus Toolkit components is prestented bellow. In addition, cross-component tools may be considered, reunited under a development kit known as the Commodity Grid (CoG) which offers high-level framework support for Java and Python, for GT2 and GT4.

2.1 Grid Architecture and Security

As main aspects to be addressed when constructing a blueprint of a grid, computational and data storage requirements and constraints have to be considered. These initial specifications are further subject to the continuous evolution of the grid technologies, thus requiring high fault tolerance and adaptability capabilities. Another characteristic to be included as initial design factor is the envisaged collaboration, interoperability and inter-communication pattern to be supported.

Under the above considerations, grids can be classified as (a) computational grids and (b) data storage grids, employing a high demand in the form of computational-intensive applications or in data storage and management, respectively. Further differentiation may

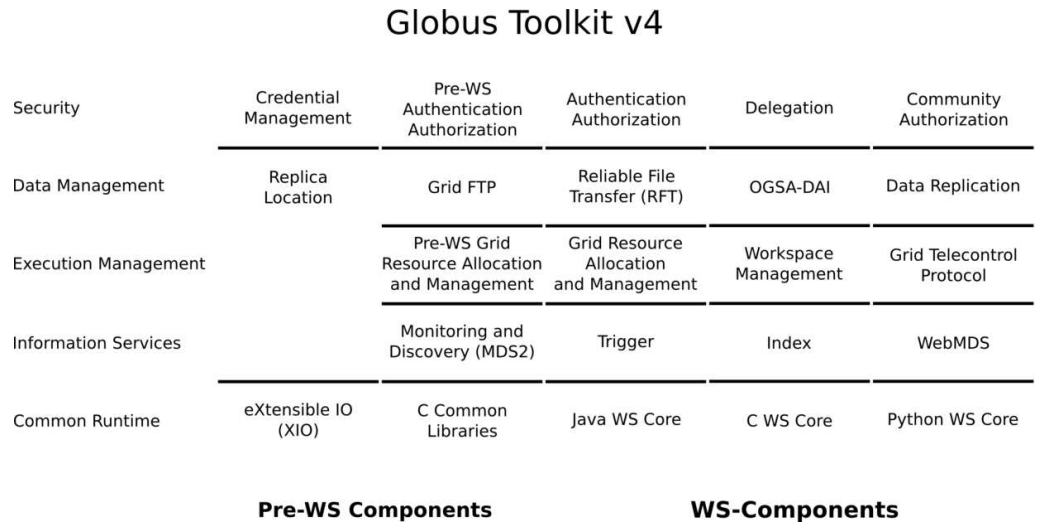


Figure 1: The Globus Toolkit components - Pre-WS and Web Services based components.

be implied by considering the chosen distribution model, distinguishing High Performance Computing as opposed to High Throughput Computing.

Computational grids allow for a high processing output obtained as the cumulative power of multiple low-level ordinary systems, heterogeneous collections of computational systems being rendered as a unified aggregate. As extensive processing power is demanded, closely coupled computing units may be unified under a cluster interface or as a cluster of clusters. At the opposite extreme loosely coupled systems can be considered, composed of domain-independent distributed volatile resources. In this scenario, each of the computational units represents an infinitesimal part of a distributed pattern disrupting data semantics awareness at the level of the processing nodes.

Data storage and management grids require secure and reliable manipulation of data over a distributed environment, acting when required as back-ends for computational grids. Performance measures relate in this case to data filtering, metadata based access, data retrieval and data relocation operations. Considering the semantics associated with the processing flow, different concepts may be introduced, *i.e.* multiple/unified query points, data consistency, resource monitoring and load balancing, etc. In this case the data stream may incur staging, translation and repartition over heterogeneous pools of data under federated database management.

As examples we may cite the case of Distributed Terascale Facility (TeraGrid), DOE Science Grid, NASA Information Power Grid (IPG), Earth System Grid (ESG), GRID5000, DAS-2, etc.

Securing a grid expanding over a large collection of heterogeneous resources, including different administrative domains, requires the authentication of the users and services as well as data transfer encryption mechanisms. Inside a virtual grid deployed via the Globus Toolkit authentication is performed by employing X.509 digital certificates delivered and signed by a Certification Authority (CA). A digital certificate encloses a digital signature, *i.e.* a public key, and descriptive elements linking the certificate to the identity of the owner - user(s), service(s), resource(s), etc. All the included descriptive attributes constitute as a unique distinguished name, specifying, for example, the name of the user and its coordinates inside the grid, for user certificates, or a Fully Qualified Domain Name (FQDN) for server certificates.

The digital certificate ports as warrant from the CA's part for the owner as being part of the grid environment or domain. Furthermore, the certificate has to guarantee for facile fraud and integrity detection, while uniquely identifying the designated entity or the domain of resources, *e.g.* there should be no confusion between the identities of distinct components of the grid. Covering a large domain of concepts and enclosing several security notions, the following terms may be considered as introductory basis:

- **authentication** - the valid identification of a resource according to the afferent certificate which, in this case, stands for its identity inside the distributed environment context. For each action or request, pertinent inside the boundaries of a limited designated domain, authentication has to be achieved first before actually performing the specified task. Authentication may be also demanded in the case of delegated actions for remotely impersonated software components.

- **authorization** - while authentication validates the identity of a user, access limitations may be imposed defining a boundary enclosing the actions the requesting entity is acknowledged as having the right to perform.

- **encryption and data integrity** - as authentication and authorization control represent local security enforcement measures, no control is exerted over inter-domain communication, hence data encryption is required. Furthermore, data integrity mechanisms have to assure for no incidental data alteration during information exchange. In addition, encryption and data integrity mechanisms assure the confidentiality of sensitive information.

Already included in the Globus Toolkit, a *Simple Certification Authority* may be used for signing and delivering certificates - to no extent replacing a fully qualified CA, as it is designated for testing purposes only. The afferent certificate management tasks are accomplished as part of the Grid Security Infrastructure in conjunction with a Public Key Infrastructure (PKI). Enabling the CA requires as a first step the generation of a pair of private and public keys, the private key being used for signing the CA's certificate. The private key is further used for signing each of the requested and delivered certificates inside the designated domain hence representing one of the main security points. A digital certificate assigned to a specific entity represents the key it employs for SSL encryption when

transferring data, impersonating the owner specified by the certificate. While relying on the fulfillment of several constraints, cross certification may also be attained by assuring mutual recognition between distinct Certification Authorities.

An additional requirement for constructing a virtual Globus grid is having synchronized all the involved machines. As time stamps are used as marker for each delivered certificate, an action is considered valid and hence authorized only for its associated time span. As a consequence, a network time protocol (NTP) has to be configured requiring the use of a time synchronization server.

While to a given extent security enforcement may prove to be sufficient and self-sustained, vulnerabilities may persist due to several causes ranging from configuration issues to inappropriate environment operation and manipulation. Unauthorized access to grid resources may result as a consequence of interference with parts of the network not concerned by the virtual grid, impersonation attacks, compromised security points, etc.

For our specific case, GRID5000, different security levels may be considered, the constructed virtual grid being placed on top of the already in place security layer. Furthermore, given the reconfigurable nature of the environment, the virtual grid represents a volatile environment lasting only for the duration of a grid reservation. Further details concerning GRID5000 as well as the superposition of the two grids, the physical in place one and the virtual grid, will be presented in the following sections.

2.2 Globus Toolkit Components

In the followings, a few main components of the Globus Toolkit are briefly presented in order to offer a general perspective over the interacting parts of the framework as well as their role in constructing a distributed grid environment and in deploying a distributed grid application.

Globus Security Infrastructure (GSI) - designed around the Secure Socket Layer (SSL) protocol with the support of Generic Security Service API. The SSL protocol, alternatively known as the Transport Layer Security (TLS), represents a standard promoted by the Internet Engineering Task Force (IETF). The core security concepts of GSI employ as key element X.509 encoded digital certificates, also a standard sustained by the IETF. As out of scope for the current paper, no further details were considered for the mentioned elements - the target functionalities offered by the GSI include mutual authentication, secured communications, delegation and single sign-on.

Monitoring and Discovery Service (MDS) - decentralized monitoring service offering the support for collecting static as well as dynamic information, describing the distributed environment. It is mainly based on the Lightweight Directory Access Protocol (LDAP), the Grid Index Information Service (GIIS) and the Resource Information Service (GRIS), services which we will not discuss as being out of scope. While not active as part of the grid image architecture delivered on GRID5000, it represents an important module to be considered for further development.

Dynamically-Updated Request Online Coallocator (DUROC) - coordinates the resource manager transactions in order to guarantee the necessary resources for executing

distributed jobs. The DUROC module acts as a coallocator, the resource managers layering the interaction with job managers. As main interest we may point the possibility of synchronizing multiple jobs by imposing a startup barrier as well as the support for inter-job communications, hence allowing for the construction of coordination mechanisms.

Grid Resource Allocation Manager (GRAM) - allows for task remote execution and state management through gatekeepers, the afferent API providing submission, monitoring and cancellation/termination primitives. The context required for the specified task to be executed can be outlined by a job request file. The syntax of this request file has to follow the rules of the Resource Specification Language (RSL). The RSL file exposed below for exemplification purposes, specifies the environment and the location of an MPI program to be launched on two machines, namely, for the herein example, *unitA.unit.edu* and *unitB.unit.edu*:

```
+
( &(resourceManagerContact="unitA.unit.edu")
        (count=1)
        (jobtype=mpi)
        (label="subjob 0")
        (environment=(GLOBUS_DUROC_SUBJOB_INDEX 0))
        (directory=/home/repository/mpis)
        (executable=/home/repository/mpis/example)
)
( &(resourceManagerContact="unitB.unit.edu")
        (count=1)
        (jobtype=mpi)
        (label="subjob 1")
        (environment=(GLOBUS_DUROC_SUBJOB_INDEX 1))
        (directory=/home/repository/mpis)
        (executable=/home/repository/mpis/example)
)
```

GRAM is responsible for interpreting and processing the RSL file as well as for taking the appropriate actions. Remote execution is completed by the gatekeeper which is responsible for creating a secure communication channel between the interacting machines and for actually creating the job manager which, in turn, handles by delegation the execution of the job. Job requests are dispatched by the job manager to local resource managers which further manage callbacks and client cancellation requests. The local resource manager is also responsible for securely sending the output results back to the client through Global Access to Secondary Storage (GASS). The stages of the remote execution are modeled as an automaton, the job manager being responsible for signaling back the state changes.

The GRAM scheduler automaton illustrated in Figure 2 corresponds to the remote execution phases. A few special states may be outlined - the **unsubmitted** state was introduced in order to model the case of a job manager restart before a job submission while **done** and **failed** represent final states. The **pending** state includes submitted jobs awaiting for the allocation of the required resources, leading to active jobs in case of success or ending in the **failed** state otherwise. The **stagein** and **stageout** states may become active only for jobs requires staging steps, *i.e.* which require staging the executable or input/output data. In

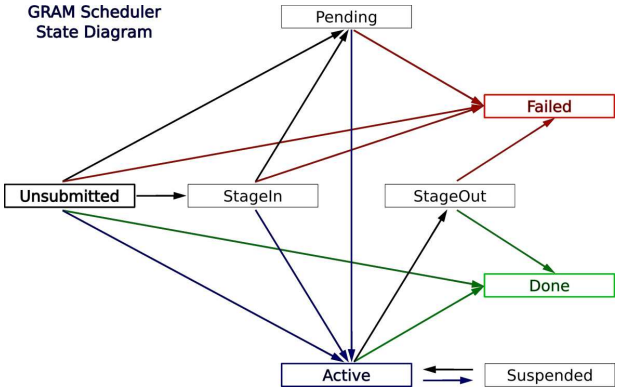


Figure 2: GRAM Scheduling Model - broker submitted jobs start as pending jobs undergoing state changes; jobs may end in only one of the final states, **Done** or **Failed**.

addition, a job may be temporarily placed by the scheduler in the **suspended** state. The **failed** state may also be reached as the result of a user or system job cancellation.

GridFTP - represents an extension of the FTP protocol on top of IETF RFCs designed to meet security requirements for the reliable transfer of data across wide-area networks. Furthermore, the implementation is optimized for high-bandwidth communication making use of multi-channel parallel transfers over GSI secured, authenticated data channels. The design of the GridFTP component sustains operations on large datasets, replicas management, indexing services, etc.

Global Access to Secondary Storage (GASS) - a service offering support for remote data access and transfer specifically adapted for wide area distributed environments and applications. The GASS service provides different grid optimized input/output patterns including caching support, staging, output simulation, real-time monitoring, data filtering, etc. The GASS also supports remote file access by offering cache-aware remote file manipulation primitives. In addition, GASS, under GSI secure environment, is used by the GRAM module for client-server output transfer mechanisms. For more details, refer to [5].

3 Installing and Configuring the Globus Toolkit

Several prerequisite software packages are required for installing the Globus Toolkit, *e.g. gcc*, *zlib*, *Apache Ant*, *Tomcat*, *PostgreSQL*, etc. As being out of scope, only the relevant phases are presented in the following sections. All the additional afferent details may be found on the Globus dedicated site (http://www.globus.org). For the specific case considered here, the post-installation process reduces to setting up the environment and security, *i.e.* obtaining certificates, creating container certificates and keys, configuring GridFTP, GRAM,

WebServices, etc. One has to notice that the herein presented steps stand only as general guidelines, while not representing by no means a substitute for the official Globus installation documentation. In addition the different presented packages may not scale for a different scenario - depending on each specific case, different versions or design patterns have to be considered. For addressing the proposed requirements, on a per-case basis, binary or source packages were chosen - this may also be different depending on the base environment, *e.g.* a *Linux Fedora Core 4* in this case.

3.1 Creating and configuring the environment

The first phase of the installation process requires creating environment variables pointing to the installation repositories of the different prerequisite components. In addition, a *globus administration* account has to be created as well as a *globus user* account - the account names were arbitrarily chosen, *e.g. globus* and *globususer*, respectively. Moreover, the PostgreSQL installation procedure creates a *postgres* administration account. The environment variables have to be replicated for each of the involved users, according to their specific role, by modifying, for **bash shell** for example, the **.bash_profile** file.

In the followings the performed steps are presented in commented snippets with a focus on the main phases of the installation/configuration process. Assuming to have already installed the packages, path variables have to be created indicating, were the case, the installation tree or the directories containing the executable files and/or libraries:

```
export JAVA_HOME=/usr/java/jdk1.5.0_04
export ANT_HOME=/opt/apache-ant-1.6.5
export CATALINA_HOME=/opt/apache-tomcat-5.5.12
export GLOBUS_LOCATION=/opt/globus/globus-4.0.1
export PATH=${PATH}:${ANT_HOME}/bin
```

Considering the repartition of roles for the created users, different parts of the installation tree have to be owned by different users, as required - we assume a binary installation of the *Apache Tomcat* package, the **/opt/globus/globus-4.0.1** representing at this step an empty directory:

```
chown -R globus:globus /usr/apache-tomcat-5.5.12
chown globus:globus /opt/globus/globus-4.0.1
```

Additionally, the PostgreSQL database has to be initialized under the *postgres* user - the output is also attached bellow:

```
su - postgres
initdb -D /var/lib/pgsql/data/


The files belonging to this database system will be owned by user "postgres".
This user must also own the server process.
The database cluster will be initialized with locale en_US.iso885915.
This locale setting will prevent the use of indexes [...].
Fixing permissions on existing directory /var/lib/pgsql/data/... ok
creating directory /var/lib/pgsql/data//base... ok
creating directory /var/lib/pgsql/data//global... ok
```

```
creating directory /var/lib/pgsql/data//pg_xlog... ok
creating directory /var/lib/pgsql/data//pg_clog... ok
creating template1 database in /var/lib/pgsql/data//base/1... ok
creating configuration files... ok
initializing pg_shadow... ok
enabling unlimited row size for system tables... ok
initializing pg_depend... ok
creating system views... ok
loading pg_description... ok
creating conversions... ok
setting privileges on built-in objects... ok
vacuuming database template1... ok
copying template1 to template0... ok
Success. You can now start the database server using:
        /usr/bin/postmaster -D /var/lib/pgsql/data/
or
        /usr/bin/pg_ctl -D /var/lib/pgsql/data/ -l logfile start
```

After initializing the database, the PostgreSQL daemon has to be started:

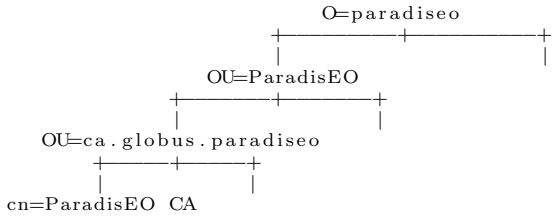
```
/usr/bin/pg_ctl -D /var/lib/pgsql/data/ -l logfile start
```

At this step it should be possible to compile and install the Globus Toolkit - the installation has to be performed as the *globus* user. Different flavors may be chosen at compile time, depending on the already installed software packages - for example, if a vendor MPI resides on the machine, an MPI flavor of Globus can be built, if desired. For the herein case no special flavor is specified, the *gcc64dbg* and *gcc64dbgpthr* being built by default:

```
su - globus
tar -xvzf gt4.0.1-all-source-installer.tar.gz
cd gt4.0.1-all-source-installer
./configure --prefix=$GLOBUS_LOCATION
make 2>&1 | tee build.log
make install
```

3.2 Setting up security

The security context is ensured by creating and employing certificates, the *globus* user being responsible for all the afferent administrative tasks - managing *SimpleCA*, starting and stopping the container, deploying services, etc. In order to accomplish all these tasks, read and write permission have to be granted for the *globus* user under the **$GLOBUS_LOCATION** directory. As a first step, generating a certificate requires to define a distinguished name - as an example, we may have the following structure:



Having chosen a distinguished name, globus specific scripts are employed for generating and signing certificates:

```
$GLOBUS_LOCATION/setup/globus/setup-simple-ca


            Certificate Authority Setup


This script will setup a Certificate Authority for signing Globus
users certificates. It will also generate a simple CA package that
can be distributed to the users of the CA.

The CA information about the certificates it distributes will be
kept in: ...

The unique subject name for this CA is:
cn=ParadisEO CA, ou=ca.globus.paradiseo, ou=ParadisEO, o=paradiseo

Do you want to keep this as the CA subject (y/n) [y]:n

Enter a unique subject name for this CA:
cn=Globus Simple CA, ou=ca.globus.paradiseo, ou=ParadisEO, o=paradiseo

Enter the email of the CA (this is the email where certificate requests
will be sent to be signed by the CA): <a_valid email_adress>
...
Enter PEM pass phrase:
Verifying - Enter PEM pass phrase:
creating CA config package...done.
...
*************************************************************************

setup-ssl-utils: Complete
```

Furthermore, as *root* user, the GSI has to be set up, creating a *grid-security* file and a trusted certificates directory - a *Globus CA certificate* is also generated at this step:

```
$GLOBUS_LOCATION/setup/globus_simple_ca_CA_Hash_setup/setup-gsi -default


setup-gsi: Configuring GSI security
Making /etc/grid-security...
mkdir /etc/grid-security
Making trusted certs directory: /etc/grid-security/certificates/
mkdir /etc/grid-security/certificates/
Installing /etc/grid-security/certificates//grid-security.conf.fa0f418f...
Running grid-security-config...
Installing Globus CA certificate into trusted CA certificate directory...
Installing Globus CA signing policy into trusted CA certificate directory...
setup-gsi: Complete
```

The second phase in securing the environment consists in creating certificates for the different roles which act inside Globus - host certificates, users, proxy, container certificates, etc. Each of the generated certificates has to be signed and sent back through a reliable secure medium to its respective owner.

3.2.1 Generating and signing host certificates

In order to exemplify the protocol required for setting up certificates, the sequence of commands performed for generating, signing and installing a host certificate is exposed bellow.

- **obtaining the host certificate** - as root, the following scripts are launched:

```
source $GLOBUS_LOCATION/etc/globus-user-env.sh
grid-cert-request -host `hostname`


A private host key and a certificate request has been generated
with the subject: ...

Please e-mail the request to the Globus Simple CA <ca@grid5000.fr>
You may use a command similar to the following:

cat /etc/grid-security/hostcert_request.pem | mail <ca@grid5000.fr>

Only use the above if this machine can send AND receive e-mail. if
not, please mail using some other method.

Your certificate will be mailed to you within two working days. If
you receive no response, contact Globus Simple CA at <ca@grid5000.fr>
```

- **sign and install the host certificate** - entitle the host as a recognized entity inside the grid. For this particular case, installing the certificate consists in only copying the file under the **/etc/grid-security** directory. In the followings, the existence of a certificate repository is assumed, here named *certrepository*, accessible by the *globus* user:

```
cp /etc/grid-security/hostcert_request.pem \
        /certrepository/`hostname`_cert_request.pem

sudo -u globus grid-ca-sign -in /certrepository/`hostname`_cert_request.pem \
        -out /certrepository/`hostname`_signed.pem -passin pass:CAmanager    \
        -dir ~globus/.globus/simpleCA/

cp /certrepository/`hostname`_signed.pem /etc/grid-security/hostcert.pem
```

While not exemplified, generating certificates for the users, proxies, etc. is similar in nature to a given extent with generating host certificates - please refer to the Globus documentation for further details.

3.2.2 Adding authorization statements

In order to authorize the *globususer* account for different later actions, performed on top of Globus, a **grid-mapfile** has to be created by the *root* user under the **/etc/grid-security** directory. In a second step, the appropriate lines have to be added to the file:

```
$GLOBUS_LOCATION/sbin/grid-mapfile-add-entry -dn
"/O=paradiseo/OU=ParadisEO/OU=ca.globus.paradiseo/
OU=machine.grid5000.fr/CN=ParadisEO CA" -ln globususer
```

3.3 Configuring the Globus components

Assuming no errors occurred during environment creation and configuration, the different Globus components have to be configured, as required. For the herein example, we will concentrate on GridFTP and GRAM only - refer to the Globus documentation for further details.

For configuring GridFTP, considering basic requirements, as an example, an **/etc/x-inetd.d/gridftp** file has to be created with the following content:

```
service gsiftp
{
        instances = 100
        socket_type = stream
        wait = no
        user = root
        env += GLOBUS_LOCATION=/opt/globus/globus-4.0.1
        env += LD_LIBRARY_PATH=/opt/globus/globus-4.0.1/lib 2
        server = /opt/globus/globus-4.0.1/sbin/globus-gridftp-server
        server_args = -i
        log_on_success += DURATION
        nice = 10
        disable = no
}
```

Furthermore, gatekeepers can be set up by creating an **/etc/xinetd.d/gsigatekeeper** file on each server machine, as follows:

```
service gsigatekeeper
{
        socket_type = stream
        protocol = tcp
        wait = no
        user = root
        env = LD_LIBRARY_PATH=/opt/globus/globus-4.0.1/lib
        server = /opt/globus/globus-4.0.1/sbin/globus-gatekeeper
        server_args = -conf /opt/globus/.../etc/globus-gatekeeper.conf
        disable = no
}
```

In addition, the following two lines have to be added in the **/etc/services** file:

```
gsigatekeeper 2119/tcp  # Globus gatekeeper
gsiftp 2811/tcp         # Globus / GridFTP
```

The globus user may be empowered to run GRAM related commands without explicit authentication by adding the following corresponding lines in the *sudoers* file, hence enabling the *globus* user to submit jobs inside the Globus virtual grid:

```
globus ALL=(username1,username2) NOPASSWD:
/opt/globus/globus-4.0.1/libexec/globus-gridmap-and-execute -g
/etc/grid-security/grid-mapfile
/opt/globus/globus-4.0.1/libexec/globus-job-manager-script.pl *

globus ALL=(username1,username2) NOPASSWD:
/opt/globus/globus-4.0.1/libexec/globus-gridmap-and-execute -g
/etc/grid-security/grid-mapfile
/opt/globus/globus-4.0.1/libexec/globus-gram-local-proxy-tool *
```

As only a few details were offered, focusing on different parts of the installation and the configuration process, for further information refer to the Globus documentation.

4 A case study - Grid5000

4.1 GRID5000 - Environment and Functional Aspects

GRID5000 is a French nation-wide experimental grid, connecting several sites which host clusters of PCs interconnected by RENATER[4] (the French academic network). GRID5000 is promoted by CNRS, INRIA and several universities[5] [1]. At this time the grid is gathering more than 3000 processors with more than 2.5 Tb of cumulated memory and 100 Tb of non-volatile storage capacity. Inter-connections sustain communications of 10 Gbps, in addition, the clusters including high speed networks (Myrinet, Infiniband, etc.). The target point to be achieved is a marker-stone of 5000 processors for late 2007, regrouping nine centers: Bordeaux, Grenoble, Lille, Lyon, Nancy, Orsay, Rennes, Sophia-Antipolis and Toulouse.

The architecture and the infrastructure of GRID5000 are designed having as main criterion deep reconfigurability capabilities. The design envisages running different operating systems, *i.e.* Linux, Solaris, Windows, different OS layers, etc., as well as the use of different grid middlewares - Globus, Unicore, P2P middleware, etc. In addition a large range of research requirements, relating to different research domains, have to be satisfied. In order to accomplish all the specifications, a deep-reconfiguration mechanism is provided, allowing for system software images to be deployed and installed, (re)configured and retrieved for later experiments. The entire protocol requires (1) a reservation phase in order to obtain an ensemble of machines distributed across the grid, (2) the deployment of one of the available software images on a pre-specified target partition, (3) the machines are rebooted with the new environment and (4) the experimentations are carried out for the duration of the reservation.

As an optional phase, the entire environment can be retrieved as an archive and stored for later experiments. We should note that the user has complete control over the deployed image - no restrictions are imposed.

Each of the GRID5000 sites offers a default environment meeting most of the common experimentation requirements. The users also have the possibility of storing and installing different packages in the afferent home account which is replicated through LDAP+NFS across the machines. The management of the user home accounts and user policies is sketched in Figure 4.1.

The user accounts are created on all sites, as mentioned, a per-site LDAP+NFS server being responsible for account data replication services inside clusters. All sites contain the same tree, local administrators having management rights on per-site branches. No inter-cluster home account synchronization is provided by default - the user is responsible for

[4]Réseau National de Télécommunications pour la Technologie, l'Enseignement et la Recherche - http://www.renater.fr

[5]CNRS - http://www.cnrs.fr/index.html; INRIA - http://www.inria.fr.

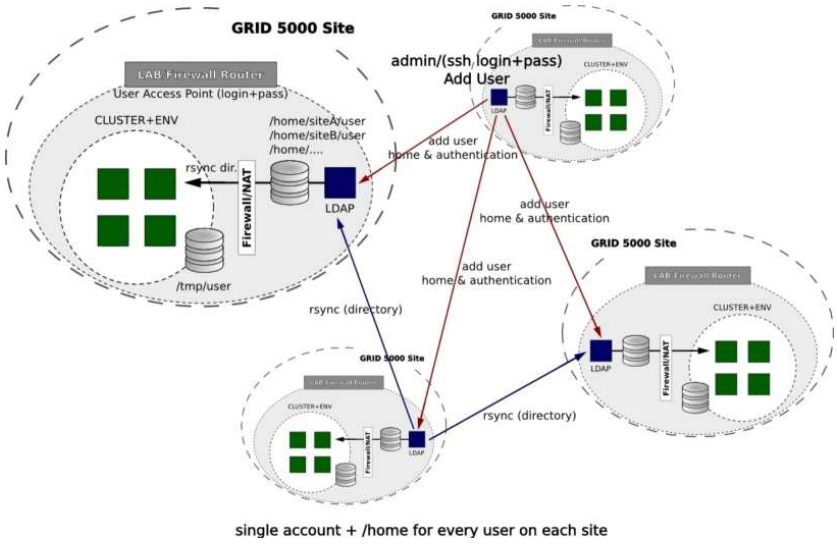


Figure 3: GRID5000 User Management

installing and customizing the account on the different sites. In case the default environment + the home account local environment proves to be sufficient for the requirements of a specific user, the experimentation protocol reduces to a reservation phase, at a cluster or grid level, specifying the duration, number of machines, sites, etc. followed by the experimentation itself.

Due to security considerations the GRID5000 sites are not directly connected to the Internet, providing an isolated domain, exterior access being possible through proxies. While restrictive to some extent, this security constraint allowed for inside enhancements - no limitations is imposed between the GRID5000 sites, thus a large scale confined cluster of clusters being built. Accessing the GRID5000 machines requires to pass strong authentication procedures which employ GRID5000 X.509 digital certificates and state-of-the-art security mechanisms. Furthermore, access is possible only through a set of designated front-end machines. A schematic overview of the GRID5000 security layer is offered in Figure 4.1.

The interconnection between sites is constructed on top of RENATER technology, using a combination of DiffServ and MPLS. For security, network layer VPNs may use tunneling or network layer encryption (layer 3 VPN), while at link layer VPNs like MPLS, VPNs are directly provided by network service providers (layer 2-3 VPN). The advantage of the MPLS VPN over IP VPN (Ipsec) is performance. As the GRID5000 sites are connected to the same NREN, the multi-domain issue of the MPLS technology is avoided here. This MPLS

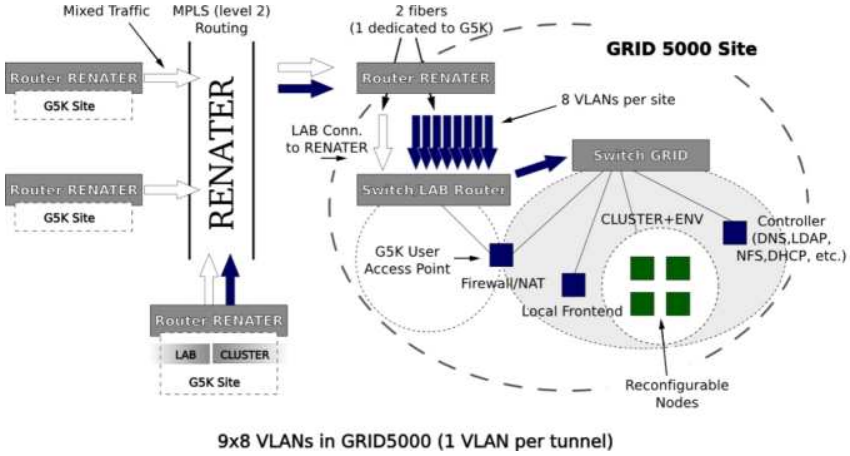


Figure 4: GRID5000 Security Architecture

based Grid architecture allows creating a trust context that even enables to experiment new security solutions for IP VPNbased Grids.

4.2 Deploying Globus on GRID5000

Node reconfiguration operations on GRID5000 are performed through a deployment tool called *Kadeploy2*[6] which allows the users to deploy customized environments on a pre-specified disk partition of a set of selected nodes. After the deployment phase, the user has complete access to all the levels of the operating system up to the kernel core components - no restrictions are imposed. A trace over the status of the reservations is maintained with the support of several specialized operating components enforced by database support. Different deployment aspects are specified and stored in the database, either as dynamic or static information, describing the coordinates of the environment to be created (disk partition schemes, deployed environment), user rights over the reserved nodes, environment description (kernel, initrd, custom kernel parameters, desired filesystem for the environment, post-installation scripts location) and logs of the deployment operations.

Assuming a reservation has been already performed, a simple deployment protocol requires the following steps:

- the user has to initiate a deployment operation by providing an environment name, a deployment partition and the name of the machines to perform the deployment upon;

[6]Complete details on the *Kadeploy2* deployment system may be found at the following address: https://www.grid5000.fr/mediawiki/index.php/Grid5000:Software#Kadeploy_2.

- for the specified environment name, if valid, the associated information is retrieved from the database - pre-specified and registered by the builder of the software image;
- at this step, the physical deployment can be initiated. The reserved nodes are rebooted on a minimal system through a networking booting sequence. The target disk partition is prepared for the deployment, *i.e.* partitioning, formatting and mounting operations are carried out, *etc.*;
- having the nodes set up, a broadcast is performed, transmitting in a pipe-line fashion the designated environment to all the specified nodes;
- after completing the broadcast phase, the post-install scripts stage-in, modifying the environment parameters in order to comply with node and site policies (mounting tables, authentication keys, etc.);
- the last phase consists in rebooting the installed environment under the new configuration, a network bootloader being employed along the process.

Having an environment deployed on one of the nodes allows for further reconfiguration. At the end of the reconfiguration phase, the environment can be saved back as an archive and registered in the database for later (re)deployment. The above presented steps outline the procedure undertaken for constructing the Globus image, on top of a basic back-bone environment, deployed and later reconfigured. A Globus installation has been performed inside the basic deployed environment, the final image containing all the required scripts[7] and components for the image to act as a master or a slave inside a virtual Globus grid. In this scenario, the front-end node runs Globus Toolkit services (GRAM, GSIFTP, gatekeeper, certificates, etc.) while a batch-scheduler is executed on each of the slave computing nodes - an overview is offered in Figure 4.2.

The deployment procedure of the Globus Toolkit image follows the above exposed steps, the configuration scripts included in the archive offering the possibility to construct a virtual Globus grid in an automatic manner, requiring only to specify the list of machines to be used. At the end of the reservation, *Kadeploy* is also the component responsible for rebooting the machines back under the default system image.

An example of typical deployment time(s) on five clusters of Grid5000 is illustrated in Table 1 - note that this may vary depending on the size of the deployed image, for example if customized, etc. In addition the deployment time may be influenced by local factors, depending on the logical/physical configuration of the cluster, which may result in protocol deployment failures, etc. In order to perform the grid deployment, on all the five clusters, namely Azur, Parasol, Paravent, Sagittaire and Orsay, the deployment procedure has been started simultaneously on all the five sites. An overall number of CPUs ranging from 20 to 300 CPUs has been considered - no significant variance or difference has been noticed between the considered sites under the specified conditions. For each of the tests a bellow 10 minutes deployment time has been obtained.

[7] All the required details and resources may be found at the following address: https://gridlille.futurs.inria.fr/dokuwiki/doku.php?id=start

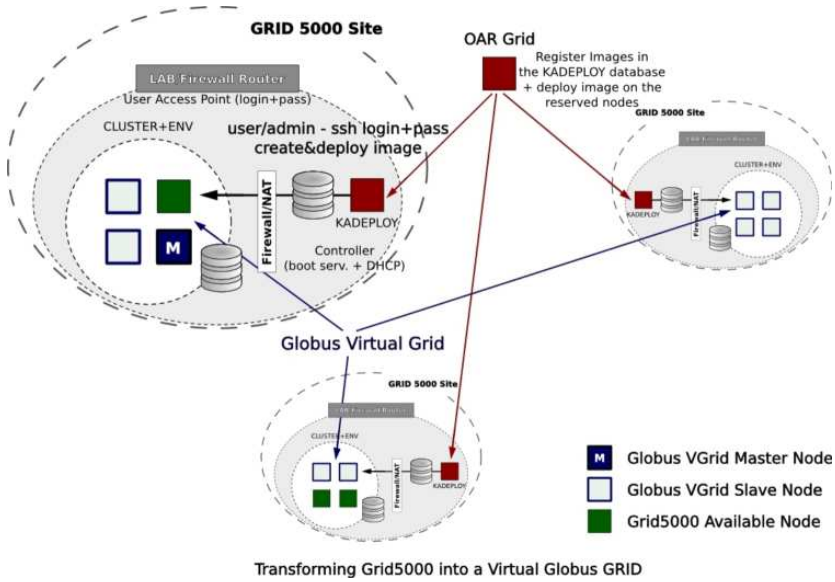


Figure 5: Globus Virtual Grid on Grid5000

			Number of CPUs per site					
			×4	**×6**	**×12**	**×20**	**×40**	**×60**
Cluster	**Azur**	Depl. time	8m47s	7m49s	5m22s	5m19s	5m19s	5m38s
	Parasol		4m49s	4m48s	4m49s	9m18s	9m41s	6m48s
	Paravent		5m6s	5m16s	5m23s	5m12s	5m44s	9m48s
	Sagittaire		4m14s	4m19s	4m28s	4m40s	8m9s	8m1s
	Orsay		4m28s	4m31s	4m35s	4m39s	4m44s	4m51s
			20	**30**	**60**	**100**	**200**	**300**

Table 1: Deployment times on five Grid5000 clusters. The upper line of the table (in bold face) represents the number of processors per site while the lower line (in bold face) represents the total number of CPUs for the overall deployment.

In addition, in Table 2, broadcasting time(s) for an MPICH-G2/Globus based application are exposed - the tests were performed on the Orsay Grid5000 cluster. While being strongly dependent on the communication infrastructure built internally by the MPICH-G2 routines (on top of Globus) the included information may offer a few performance measures of the environment. The obtained broadcast time(s) may also stand as a comparison basis for different further communication patterns, etc.

Number of CPUs		Broadcast time	Broadcast message size (in bytes)								
			64	**128**	**256**	**512**	**1024**	**16384**	**65536**	**524288**	**1048576**
	5		12s	12s	12s	12s	12s	12s	12s	12s	12s
	15		20s	17s	21s	20s	20s	17s	16s	16s	20s
	30		31s	32s	30s	30s	31s	33s	31s	30s	32s
	50		46s	45s	48s	46s	47s	48s	46s	45s	45s
	100		83s	81s	82s	82s	84s	81s	82s	82s	87s
	150		117s	115s	120s	117s	118s	119s	121s	121s	116s
	300		234s	232s	230s	233s	235s	231s	233s	239s	232s

Table 2: Broadcasting times for an MPICH-G2 based program. The first column of the table (in bold face) represents the number of CPUs/processes (one process per CPU). The upper line (in bold face) represents the size of the message, in bytes, to be broadcasted to all the processes.

For each test run, the overall execution time has been considered - initialization, broadcast, termination. As it can be observed from the table, the size of the broadcasted message is a negligible factor as compared to the number of considered processors, *i.e.* it does not influence significantly the overall execution time. This effect is due to the fact that, for MPICH-G2 based programs, the initialization phase requires interconnecting each process with all the other processes. Network interferences may also be responsible for the slight variation in execution time.

5 Conclusions

While being a complex infrastructure, a virtual grid has the advantage of being easily extendible over the boundaries of the physical in-place grid. In addition, considering the reconfigurable nature of Grid5000, the possibility of having a completely user-reconfigurable Globus environment opens the pathway for a broad range of experimental setups. As the main difficulty in constructing a Globus virtual grid consists in constructing and configuring the environment, having a ready to use Globus deployable image, eliminates most of the techincal aspects, hence offering the possibility to set a focus on the experimental part.

The built Globus image represents a testbed for further improvements and development, having also the main advantage of being user-(re)configurable. This is particularly important when considering the imminent future interconnection of Grid5000 with external grid/distributed computing projects like the Dutch DAS-3[8] (*Distributed ASCII Supercomputer, third generation*) or NAREGI[9] (*National Research Grid Initiative* - University Science Malaysia).

[8] http://www.cs.vu.nl/das3/overview.shtml
[9] http://www.naregi.org/concept/index_e.html

References

[1] Cappello, F.; Caron, E.; Dayde, M.; Desprez, F.; Jegou, Y.; Primet, P.; Jeannot, E.; Lanteri, S.; Leduc, J.; Melab, N.; Mornet, G.; Namyst, R.; Quetier, B.; Richard, O., *"Grid'5000: a large scale and highly reconfigurable grid experimental testbed,"* Grid Computing, 2005. The 6th IEEE/ACM International Workshop on , vol., no., pp. 8 pp.-, 13-14 Nov. 2005.

[2] Ian Foster, *"A Globus Toolkit Primer"*, www.globus.org/primer,. 2005.

[3] Ian Foster, Carl Kesselman, and Steve Tuecke. *"The anatomy of the grid: Enabling scalable virtual organizations"*. International Journal of Supercomputer Applications, 2001.

[4] Foster, I., Kesselman, C., Nick, J. and Tuecke, S. *"The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration"*. Globus Project, 2002, www.globus.org/research/papers/ogsa.pdf.

[5] Bester, J., Foster, I., Kesselman, C., Tedesco, J. and Tuecke, S. "GASS: A Data Movement and Access Service for Wide Area Computing Systems", Proc. IOPADS'99, ACM Press, 1999.

[6] N. Bieberstein, C. Gilzean, and J.Y. Girard et al. *"Enabling Applications for Grid Computing with Globus"*. IBM Corp., 2003.

Contents

Unité de recherche INRIA Futurs
Parc Club Orsay Université - ZAC des Vignes
4, rue Jacques Monod - 91893 ORSAY Cedex (France)

Unité de recherche INRIA Lorraine : LORIA, Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex (France)
Unité de recherche INRIA Rennes : IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex (France)
Unité de recherche INRIA Rhône-Alpes : 655, avenue de l'Europe - 38334 Montbonnot Saint-Ismier (France)
Unité de recherche INRIA Rocquencourt : Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex (France)
Unité de recherche INRIA Sophia Antipolis : 2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex (France)

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-0803