ISTITUTO NAZIONALE DI FISICA NUCLEARE
Progetto INFN-GRID

INFN/TC-01-09
June 30, 2001

Report on the INFN-GRID Globus evaluation

Roberto Alfieri[9], Cosimo Anglano[12,11], Roberto Barbera[2], Massimo Biasotto[5], Piergiorgio Cerello[11], Andrea Chierici[3], Andrea Controzzi[10], Flavia Donno[10], Tiziana Ferrari[3], Luigi Fonti[3], Antonio Forte[11], Luciano Gaido[11], Francesco Giacomini[3], Alberto Gianoli[4], Claudio Grandi[1], Andrea Guarise[11], Ivano Lippi[8], Giuseppe Lo Biondo[6], Stefano Lusso[11], Lorenzo Marzola[4], Francesco Prelz[6], Silvia Resconi[6], Carlo Rocca[2], Franco Semeria[1], Andrea Sciabà[10], Massimo Sgaravatto[8], Fabio Spataro[9], Gennaro Tortone[7], Giulia Vita Finzi[3], Zhen Xie[10]

[1)] *INFN, Sezione di Bologna,* [2)] *INFN, Sezione di Catania,* [3)] *INFN, CNAF,*
[4)] *INFN, Sezione di Ferrara,* [5)] *INFN, Laboratori Nazionali di Legnaro,*
[6)] *INFN, Sezione di Milano,* [7)] *INFN, Sezione di Napoli,*
[8)] *INFN, Sezione di Padova,* [9)] *INFN, Gruppo collegato di Parma,*
[10)] *INFN, Sezione di Pisa,* [11)] *INFN, Sezione di Torino,*
[12)] *Università del Piemonte Orientale*.

Abstract

This a report on the Globus software evaluation activities that have taken place within the INFN-GRID project (INFN-GRID Work Package 1).

PACS:89.80

Published by **SIS-Pubblicazioni**
Laboratori Nazionali di Frascati

Contents

Chapter 1

Introduction

In order to facilitate the creation and the deployment of usable computational Grids, a certain number of core services need to be available. These should provide basic functionalities such as, for example, user authentication and authorization, resource and data management, "publishing" of information about the characteristics and the status of the various components of the Grid environment, etc. The toolkit provided by the Globus project [4] has been identified as a possible candidate for this Grid framework, in particular because its "layered bag of services" model seems suitable to support a wide variety of applications and environments, and the various services are distinct and have well-defined interfaces, and therefore they can be integrated into applications or tools in an incremental fashion. We therefore proposed, in the context of the work package 1 of the INFN-GRID project (Installation and evaluation of the Globus toolkit) [5], to investigate the functionalities of the Globus toolkit, evaluating the Globus packages for their effectiveness, completeness, robustness, ease of use, to find if and how these services or some of these services could be useful for our needs, what is missing, what are the open issues and how these could be addressed. These goals, set forth in the work plan [6], have been successfully met and fulfilled.

In order to perform a thorough and precise evaluation of the Globus functionalities, not based only on documentation, comprehensive tests have been performed, evaluating the various Globus services, considering different configurations and different use-cases. Besides simply evaluating the various Globus services as they are, we have also addressed some existing shortcomings. Moreover, we have also defined and implemented some specific configurations and customizations that we first tested and used in our INFN test environment, but can be very useful (or necessary) for wider environments. We also faced the problem of reducing the complexity for the Globus installation and maintenance, limiting the manpower required for these operations, implementing a specific tool, able to ease the process of install the Globus toolkit: we proved that this tool can be very useful

to effectively deploy the Globus software on a wide scale.

This document summarizes all these activities and the results of this Globus software evaluation, that took place between June 2000 and January 2001. Chapter 2 describes the experience in providing the INFN-GRID community with a tool to easy the process of installing and deploying the Globus toolkit. Chapter 3 reports on the evaluation of the Globus services that provide user authentication and authorization in a Grid environment, describing also how some existing shortcomings have been addressed. Chapter 4 summarizes the activities concerning the information service, a key element of a Grid system, since it must provide, in a common interface, information on system components of the dynamic Grid environment. The chapter, besides reporting on the evaluation of the Globus Grid Information Service (GIS), describes also how the GIS infrastructure and topology have been defined and implemented in the INFN test environment. Chapter 5 shows the activities related with the evaluation of the Globus services for resource management. In particular the Globus GRAM service, the component at the bottom of the Globus resource management architecture, has been tested. The chapter also reports on tests performed in order to evaluate the MPICH-G2 package, the implementation of MPI integrated with the Globus services, and includes also a preliminary evaluation of the GARA toolkit, the framework implemented by the Globus team, used for advance reservation of resources. Chapter 6 shows the results concerning the evaluation of some tools, implemented in the context of the Globus project, related to Grid data management. In particular tests have been performed on the functionalities and features of the so-called GlobusFTP, a wide implementation of the GridFTP protocol. Chapter 7 reports on the other Globus services (the HBM and the GEM) that we planned to evaluate, but that, even though reported in various Globus documents, are not seeing an active development. Chapter 8 describes the activities performed by some HENP experiments within INFN, in order to evaluate the Globus services with real applications and in real production environments, for a possible use in their activities. Our conclusions can be found in Chapter 9.

This report was also submitted to the Globus development team for comments. The comments that were received can be found in the *Response from Globus Team to the "Report on the INFN-GRID Globus evaluation"* that is made available and distributed along with the present document.

A short language note before we start: DataGrid (with capital D and G) refers to the European Union funded project ([3]), while INFN-GRID refers to the corresponding italian project (funded by INFN, [1]).

Chapter 2

Globus deployment and installation tools

In this chapter we describe the experience in providing the INFN-GRID community with a tool to ease the process of installing and deploying the Globus toolkit. We outline the requirements imposed by the INFN community, the common problems and intrinsic behaviour of the Globus tools, the INFN Globus Installation toolkit with the INFN customisations and future plans.

2.1 The Globus installation: common problems

The installation and deployment of the Globus toolkit, with the procedures provided by the Globus Team, has shown various problems. Before installing Globus, the administrator needs to install all the base products components, such as SSLeay and OpenLDAP. Such products need to be compiled sometimes with special options, depending on the platform and options in use. The toolkit comes with three main procedures to compile, configure and deploy Globus on one machine: *globus-install*, *globus-setup* and *globus-local-deploy*. *globus-install* is the procedure to compile Globus. It is a procedure that sits on top of the output provided by the GNU *autoconf* tools. *globus-setup* is used to configure a special DN space and/or a GIIS. *globus-local-deploy* partially configures the globus services, while further configurations are left to the manual intervention of the administrator. Some of the limitations of this procedure are:

- The installation and configuration are partially manual.
- A lot of compilation switches are available and some of them are required for certain platforms (but this info is not outlined in the documentation).

- Common user mistakes during the installation steps deal with manual services configuration, file locations and permission, local specific customisations, missing detailed documentation.
- Very long compilation times (mainly due to the fact that the entire toolkit is by default compiled multiple times) have caused frustration. Some users have given up Globus installation.
- No hooks for local customisations are provided.

In order to shorten the installation time of the Globus toolkit, avoid common mistakes and provide an easy tool for specific customisations, INFN has come up with an installation tool that has proven to be successful not only inside INFN but also outside at CERN and FNAL. Such a tool allowed to setup a GRID testbed inside INFN, granting uniformity of installations and quick problem solving and support.

2.2 Installation requirements

The HE(N)P community, especially the one with experiments based at CERN, has a long tradition of software distributed in a ready-to-use fashion via mirroring mechanisms and the use of the AFS (Andrew File System, currently marketed by Transarc) filesystem or very simple automatic procedures. In most cases, the software is distributed in a precompiled form for the OS platforms of interest with the possibility of binary file relocation. Support for special local environments and for the customisation of user startup files is always granted. The possibility of sharing an installation among several machines or installing (in user space) a further copy of a product for testing purpose is also a feature required and needed. Here's the list of requirements that led to the INFN-GRID Globus toolkit:

1. Distribution of binary files in order to avoid long compilation times.
2. Support for the most used platforms in the HE(N)P world (Linux RedHat and Solaris).
3. Allow for binary file relocation so that Globus can be installed anywhere in the system. This could be useful for testing purpose and to avoid assumptions on specific machine installations.
4. Support not only for the `fork()` based Globus jobmanager, but for multiple job managers such as LSF, Condor, PBS.

5. Configuration support for standard and HEPIX (CERN) user environments.
6. Quick distribution of Globus patches as soon as they become available, without waiting for the next release of Globus.
7. Distribution of the correct version of packages needed for installing/using Globus.
8. Support for shared installation of clients.
9. Support for multiple Certification Authorities.
10. Support for local customisations such as the INFN hierarchical GIIS structure (see Section 4).
11. Support for distribution of new tools or packages (*certretrieve*, *gsincftp*, *gsi-wuftpd*, *gdmp*).
12. Keep the distribution toolkit as general as possible and fully compatible with the standard Globus installation.
13. Support for multiple flavours of Globus (afs/kerberos/mpich-g2/etc.).
14. Provide upgrade and build procedures. Support for RPMs wherever possible.
15. Support for unattended installation.
16. Good documentation. Availability of a channel to keep users informed, of a distribution site with access control and of a WEB site. Creation of a support team.

INFN tried to satisfy the requirements described above with the *INFN-GRID Installation toolkit*. In the following section we describe our attempt to create such a tool and the difficulties found with the Globus toolkit.

2.3 A brief description of the INFN-GRID Installation toolkit

The INFN-GRID Installation toolkit comes with one main installation procedure plus an upgrade/uninstall and a build procedure. *grid-install* is the main installation script. The script can be run, both interactively or unattended, via a configuraton file. It must be executed by `root`, but the existance of the generic user `globus` is required. After downloading the software from the standard distribution site, the procedure installs the chosen flavor of the software in the defined `${GLOBUS_INSTALLATION_PATH}` tree and configures the job manager(s) of choice, as user `globus`. The installer can choose to install

optional software, to proceed with INFN specific customizations and to use INFN tools. INFN specific customizations include support for certificates signed by the INFN CA and the configuration of a hierarchical GIIS architecture with a top level central GIIS [10], although many GIIS architectures are supported as well. Several useful tools developed within the INFN-GRID project are included in the distribution toolkit, and in particular:

1. Procedures to generate the key and a request for obtaining a certificate signed by the INFN CA for both a user and a host.

2. Support for the automatic update of the INFN-CA CRL (Certificate Revocation List).

3. A tool to obtain user certificate DNs from a central LDAP server in order to update the local grid-mapfile.

The procedure is organized in steps. If one step fails, the administrator can verify and fix the cause of failure and start again from the point of the failure. Internally, *grid-install* invokes *globus-setup* and *globus-local-deploy*. *globus-root-setup* is the procedure invoked by `grid-install` and executed by `root`, which configures globus services, sets the right ownership and mode on system files and configures the user login files. Also with this procedure one can choose to perform various steps, or to skip a step. The HEPIX environment is also optionally supported. The steps executed by *globus-root-setup* would be executed by hand during a normal globus installation. Build and upgrade/uninstall procedures were also created. In particular, the build procedure helped us distributing the compilation burden at several INFN sites and using distributed resources. The upgrade procedure insures a smooth upgrade from one release of the Installation toolkit to the next. Whenever full functionality of the package cannot be granted with an upgrade (for instance if a major Globus release comes up), then a reinstallation is recommended.

This way, most of the requirements specified in Section 2.2 are currently met in the INFN distribution toolkit.

2.4 Creation of the INFN-GRID Installation toolkit

The first requirement we had to deal with was binary file distribution and relocatability. After solving all problems related to package dependencies and compilation switches, we had to setup a compilation bed of machines for the supported platforms. In setting up an automatic and unattended build procedure we found problems in the way the *globus-install* procedure tries to guess certain details of system setup. For instance, in the file `acmisc.m4`, for Sun machines, in order to find the C compiler, a `find /opt` is performed

and the first compiler found is taken. This operation can be executed more than once, thus slowing down the entire build process. Beside the fact that the find operation is really slow, the first compiler found could be not a good choice. On some of our machines the compiler used was the old version (the first found by `find`), causing the build procedure to fail. We had to find a more appropriate mechanism to solve problems like the one described above. In order to grant binary file relocatability, we looked into the build mechanism of Globus. While relocatability for SSLeay and OpenLDAP is granted, after compiling Globus on a specific directory tree, a lot of the code contains hard-coded paths. In the Configuration subdirectory one can find a set of `.m4` and `.in` files used by GNU *autoconf* wich define macros for packages configuration both at the level of C header files and initialisation scripts included by other scripts. While we ignored the problem of the header files (where the hardcoding is mostly in `globus_config.h`) since the first goal was to provide a binary distribution for users of Globus and not for developers, we tried to fix the hardcoding in the scripts. In particular, the script `globus-script-initializer.in`, is parsed by the build procedure to produce `globus-script-initializer`, substituting, among others, the macro `@installed_exec_prefix@` with the actual installation directory path. We found that, setting the variable `GLOBUS_ARCHITECTURE_INDEPENDENT` equal to `TRUE` (the variable is not really used for a specific purpose but is a left over by the Globus Team), we can force the script to use the `GLOBUS_INSTALL_PATH` variable as `prefix` or `exec_prefix` everywhere in the scripts, in order to obtain scripts relocatability. This is exemplified by the following code from `globus-script-initializer.in`:

```
if [ "X${GLOBUS_DEPLOYED}" = "XTRUE" ]; then
    export GLOBUS_DEPLOY_PATH
    prefix=${GLOBUS_DEPLOY_PATH}
    exec_prefix=${GLOBUS_DEPLOY_PATH}
    arch="`${prefix}/sbin/config.guess`"
    arch_sysconfdir="${prefix}/etc/${arch}"
else
    prefix="${GLOBUS_INSTALL_PATH}"
    exec_prefix=@installed_exec_prefix@
    if [ "X${GLOBUS_ARCHITECTURE_INDEPENDENT}" = "XTRUE" ]; then
if [ '@installed_exec_prefix@' != '${prefix}' ] ; then
    arch="`${GLOBUS_INSTALL_PATH}/sbin/config.guess`"
    exec_prefix="${prefix}/services/${arch}"
    arch_sysconfdir="${prefix}/etc/${arch}"
fi
```

```
    fi
fi
export GLOBUS_INSTALL_PATH
sbindir=${exec_prefix}/sbin
bindir=${exec_prefix}/bin
libdir=${exec_prefix}/lib
libexecdir=${exec_prefix}/libexec
includedir=${exec_prefix}/include
datadir=${prefix}/share
sysconfdir=${prefix}/etc
sharedstatedir=${prefix}/com
localstatedir=${prefix}/var
```

Another source of problems was the `globus-sh-commands.sh.in` script. This script is also parsed by the build procedure to localize all commands used by globus and provide full paths to them. This script is sourced in all other globus scripts. In our installation procedure we had to parse the output (`globus-sh-commands.sh`) produced by globus-build to adapt it to what found on the installation machine. Many other scripts needed to be parsed for hardcoded hostnames and directory paths. These installation problems in the Globus software were reported and shared with the Globus development team. They are planning to address these issues.

2.5 Documentation and support

In order to make the Globus installation and deployment easy, a very important part is the availability of a distribution site to facilitate software downloads, of a distribution list of registered users to inform them about problems and news, and of a WEB site with all relevant information and documentation about the software releases. The installation toolkit is available for downloads through two methods. The software is available to the INFN community via AFS (`/afs/infn.it/project/infngrid`) and to the rest of the DataGrid community via HTTP. [8]. The second method is available outside INFN and so the use license is shown. Access to the directories containing the files is not direct but filtered by a CGI procedure that allows a user to navigate the directory tree with an interface emulating a direct http access, keeping the root location hidden. This is done with the goal of denying direct access to the site and force users to execute the procedure. When a file is selected for downloading, the cgi procedure forces the user to fill out and submit an identification form. Only when this procedure is performed the file is downloaded. The identification data are registered in a database and the user mail

address is inserted in one mailing list (`infngrid-toolkits@infn.it`), used to inform users about news, problems, etc. related to the toolkit. Once the identification form is submitted, the user is associated with a cookie, which relieves the user from the task of refilling the form for each selected file during the same day. A WEB site [7] has been set up to keep users informed and to provide detailed documentation. A support team (`grid-release@pi.infn.it`) is always available to answer to questions and solve problems about the distribution toolkit and configuring Globus and related products.

2.6 Available versions and future plans

The installation toolkit has started with version 1.0 as a set of documents, to clarify certain installation steps and try to make easy the Globus configuration task. With version 1.1, automatic installation procedures were provided. New documentation specified not only the steps for fast installation but also a detailed list of bug fixes. Most of the requirements were already satisfied, offering support for the customisation of a central top-level GIIS, and INFN CA signed certificates. The platforms supported were Linux RedHat 6.1 and partially Sun Solaris 2.6, although some users reported the toolkit to work also for RedHat 6.2. The Condor and LSF job managers could be configured. In version 1.2, beside distributing auxiliary packages such as *gsincftp*, *wu-gsiftpd* and *gdmp*, upgrade and build procedures, support for the PBS batch system and full support for Solaris 2.6 were added. While using version 1.2, some memory leaks problems and bugs in some scripts were found. Patches to version 1.2 were reported to the Globus development team, and provided to the INFN community to automatically fix those problems. Version 1.3 is now ready for distribution, with the following list of features:

1. Fixes for Globus jobmanager memory leaks and various other minor bugs.
2. Support for Solaris 7.
3. Full support for GDMP v1.2.2.
4. Distribution of various Globus flavored compilations (Kerberos,AFS,MPICH).
5. Distribution of the INFN CA CRL tools.
6. Distribution of the INFN *certretrieve* tool to manage grid-mapfiles for Linux and Sun.
7. Distribution of the `infn-send-request` script for user certificates.
8. Support for unattended installations.

9. Support for multiple architectures of GIIS.

The Toolkit has been used up to now by, at least, 11 INFN sites, by some installations at CERN and FNAL and in Finland. It has been evaluated in U.K. and by the Globus Team, with the conclusion that this it is the most complete installation toolkit available at the moment. It has been proven to be successful although an installation of the main component and accessories via RPMs is among the most desired additions.

The installation toolkit will evolve in order to serve the needs of the DataGrid comunity. It will include support for unattended installation both via scripts and RPMs wherever possible. The support for source distribution will probably change, trying to encounter the needs for the setup of a developer environment for DataGrid, not addressed at the moment. There is a plan, by the Globus Team, to completely change the structure of the software, to allow for a more flexible modularisation of the software with support for distribution via tarballs, etc. For a list of desiderata by the Globus Team see [9]. The Toolkit will surely adapt to converge toward a generalized tool for use by the DataGrid community.

Chapter 3

Security services

In this section we evaluate what is possibly the most mature part of the Globus package, namely the services that provide user authentication and authorization. We also describe how the existing Globus shortcomings and security issues were addressed within the INFN test enviroment, and comment on the issue of configuring Globus for a national or institutional (i.e. non-Globus) Certification Authority.

3.1 Evaluation of the Globus Security Infrastructure (GSI)

In order to evaluate the services provided by GSI we first tried to identify the typical security needs of applications and user communities within INFN:

- Experiments of the LHC era certainly[11] need to geographically distribute and transparently access and analyse datasets on the scale of many Tbytes.
- Accessing and transferring data over clear pipes, with traditional, username/password FTP authentication or AFS, is still common practice for many geographically spread communities, especially in the field of High Energy and Nuclear Physics (HE(N)P). Sometimes SSH is used.
- HE(N)P experiment communities show little concern for the intrinsic security of the data, since these are of negligible value without the appropriate analysis environment and collaboration expertise. This of course may not be true for other DataGrid customers and the grid at large, and may change as the self-describing content of experimental data increases.
- Some concern is shown for the protection of semi-final, unpublished processed data. It is important to be able to identify the group/experiment affiliation of grid users, and grant access permissions accordingly. In general, however, ease of data access

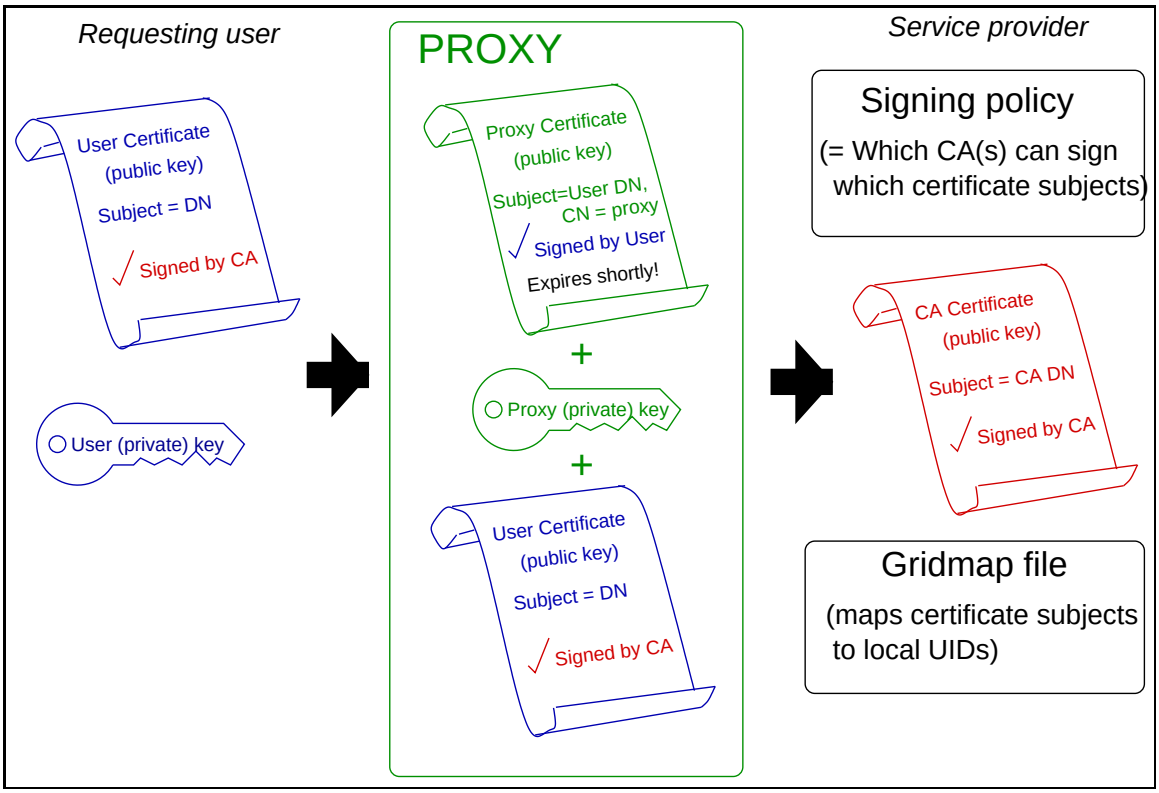


Figure 3.1: Basic elements of the GSI one-time-login model.

and reliability of the data transfer process are recognized as more important than data privacy.

- A lot of concern is shown towards increased security measures that complicate day-by-day work patterns. One-time-login mechanisms are welcome.

We will now briefly describe the operation of the GSI model and determine whether it fits these needs.

3.1.1 Operation of the GSI model

In order to describe the workings of GSI we will refer to Figure 3.1.

First of all, GSI is based on an implementation of the standard (RFC 2078/2743) Generic Security Service Application Program Interface (GSS-API). This implementation is based on X509 certificates and is implemented on top of the OpenSSL[12] library.

The GSS-API requires the ability to pass user authentication information to a remote site so that further authenticated connections can be established (one-time login). The

entity that is empowered to act as the user at the remote site is called a ”proxy”. In the Globus implementation, a user ”proxy” is generated by first creating a fresh certificate/key pair, that are associated with the certificate of the user who is supposed to be represented by the proxy. The certificate has a short (default value of 1 day) expiration time and is signed by the user. The user certificate is signed by a Certificate Authority (CA) that is trusted at the remote end.

The remote end (usually at some service provider's site) is able to verify the proxy certificate by descending the certificate signature chain, and thus authenticate the certificate signer. The signer's identity is established by trusting the CA, and in particular by understanding and accepting its certificate issuing policy (see Section 3.2 for an example of such policy).

The last remaining step is user authorization: the requesting user is granted access and mapped to the appropriate resource provider's identifiers by checking the proxy (or user) certificate subject (X.500 DN) and looking it up in a list (the so-called *gridmap file*) that is maintained at the remote site. This list typically links a DN to a local resource username, so that the requesting user can inherit all the rights of the local user. Many DNs can be linked to the same local user. Some issues of *gridmap file* handling are dealt with in Section 3.4.

3.1.2 GSI advantages

As we pointed out above, the INFN community current requirements for data privacy are not so stringent, and the authentication and authorization needs seem to be addressed in a suitable way by the GSI model.

The GSI model provides (via the GSS-API compliance) a one-time login mechanism. This matches the ease of access requirement, and has already proven to be useful in the existing applications of the Andrew File System (AFS).

An advantage of GSI over Kerberos authentication is the use of X509, which is by now a more widespread user identification and digital signature mechanism. X509 is also the only digital signature framework that is currently granted legal meaning in Italy.

GSI also provides a scheme (via the Globus Authentication and Authorization library - GAA) for extending relations of trust to multiple CAs without having to interfere with their X.500 naming scheme. See Section 3.2 for more details.

3.1.3 GSI shortcomings and ways to address them

1. The fact that authorization is based only on certificate subjects allows for the existance of multiple valid certificates for the same subject. This means that care must

be taken in making sure that revoked certificates (certificates that were compromised and revoked by the CA) are not accepted. The Globus GSS-API explicitly tries to find and check a CA Certificate Revocation List (CRL) when verifying proxy certificates. This means that the CRL must be present and up-to-date at every GSI-capable site. There are no specific tools in the current Globus toolkit to handle this (see Section 3.3 for specific measures that were taken in the INFN installations).

2. The authentication library is based on OpenSSL, but it is not currently in sync with the OpenSSL development (it links against OpenSSL 0.9.1c at latest). This was reported to the Globus development team in June 2000, but no new release of Globus appeared throughout the 6-month evaluation period.

3. The authentication library provides cryptic diagnostics (e.g. "certificate chain too long" when the CA policy check fails). Again, this was reported to the development team and assurance was received that the entire diagnostic machinery was under review, but no improved version of the software was released so as to verify this.

4. The model where generally valid (even if for a limited time only) private keys are available on remote hosts fits a world where all system administrators are honest and able to implement a seamless security model. Good security practices call for limited scope proxy certificates. The only current limitation is in the ability to further delegate a proxy (which already is a delegated form of user credential). We were told by the Globus developers that limited (by scope or purpose) proxies were in the works, but again haven't been able to evaluate these extensions.

5. The Globus v1.1.3 GAA library has proven (during the production tests made for the CMS experiment described in Section 8.3) to suffer from many memory leaks. Patches for these leaks were provided to Globus by INFN-GRID, and eventually incorporated in the Globus code repository. We haven't been able to test a new Globus release to make sure all these issues are actually taken care of.

6. The GSI infrastructure doesn't provide any tool to handle the association of user identities (certificate DNs) to activity (experiment) specific groups. As we pointed out in section 3.1 this is an important requirement, so some appropriate solution has to be provided, possibly in the framework of the CAS (Community Authorization Services) development.

3.2 The INFN CA

Soon after realizing that the GSI security model seems to satisfy the INFN enviroment requirements, one has to face the issues related with establishing user identities in an efficient way. The Globus project provides a Certification Authority (CA) at their development sites that implements a rather lax method of establishing certificate requestor identities. It was thus felt that a more local scope CA (such as the esisting X509 INFN CA[13]) could provide a higher quality service.

This arrangement can be rather simply accomodated via the existing GSI tools. The GAA library accesses and interprets a rather detailed configuration file (*Globus deploy dir*`/share/certificates/ca-signing-policy.conf`) that allows to specify multiple CAs along with the certificate subjects that can be accessed by them, as in the following example:

```
# EACL entry #1|
 access_id_CA   X509    '/C=US/O=Globus/CN=Globus Certification Authority'
 pos_rights     globus  CA:sign
 cond_subjects  globus  '"/C=us/O=Globus/*" "/C=US/O=Globus/*" "/O=Grid/O=Globus/*"'
# EACL entry #2|
 access_id_CA   X509    '/C=IT/O=INFN/OU=Authority/CN=INFN CA (2)'
 pos_rights     globus  CA:sign
 cond_subjects  globus  '"/C=it/O=INFN/*" "/C=IT/O=INFN/*"'
```

As far as Globus is concerned then, there are no technical issues related with accepting certificates signed by the INFN CA. The focus then shifts to the process that is followed by the CA when signing a new certificate request. This process determines the quality of the CA and the level of trust that can be extended to it. Here is an excerpt of the RFC2527-based INFN CA Certificate Policy and Certification Practice Statement ([14]):

> **Authentication of Individual Identity**
> Procedures differ if the subject is a person or a server:
>
> - **Person** (listed in the official INFN phonebook).
>
> INFN CA staff call the subject by phone and check if the request was from him/her. The authentication procedure fails after five days of unsuccessful attempts.
>
> - **Person** (not listed in the official INFN Phonebook).
>
> The request **must** be accompanied by an e-mail to `infn-ca@fi.infn.it`, **signed by a valid INFN CA certificate**, certifying the identity of the subject.

- **Server**

  Requests **must** be signed with a valid INFN CA certificate.

3.3 CRL distribution

As we mentioned above, compromised X509 certificates are revoked by the issuing CA and included into a Certificate Revocation List (CRL). As there's no provision in the current release of Globus for automatically retrieving the CRL for the various trusted CAs, an appropriate PERL script (`distcrl.pl`) was added to the INFN Globus installation kit (see Chapter 2), and the Globus site administrator is given the option of running the CRL update as a *cron* job. The scripts retrieves the INFN CA CRL via HTTP and stores it with the appropriate name (*certificate hash*`.r0`) in the appropriate CA certificate area (typically *Globus deploy dir*`/share/certificates/`).

3.4 Centralized management of the `grid-mapfile`

One of the key characteristics of the distributed system that constitutes a cooperative Grid environment is the ability to leave the individual system administrators complete control over the authentication and authorisation policies. Two basic strategies can be implemented:

1. The same common ("group") local ID can be assigned to many different remote users.

2. A different local user ID (possibly generated on the fly and with a limited scope and lifetime) is assigned to every remote user.

The first approach simplifies the administration and handling of the membership of individuals to a specific group or activity (the group membership information is just basically the mapping of several user identities or certificate DNs to a local username). The downside is that it is virtually impossible to distinguish between files that are generated by different remote users, when these are mapped to the same local user. On the other hand, the second approach is much harder to automate, since a detailed rights database must be maintained in order to create the users correctly.

The Globus toolkit does not provide tools to handle this kind of group information, and does not include tools to replicate the authorisation information over a series of hosts or computing resources that are used for the same activity: this can be done only by manual editing of the `grid-mapfile` on every Globus host. This task can lead to misalignments of access policies and complicates the management of the `grid-mapfile`.

For this reason the INFN-GRID group has planned to implement a system that simplifies gridmap-files management, allowing Globus administrators to update their `grid-mapfile` with consistent information. This can be done implementing a central repository of users information to be used for authentication and authorization in the Globus environment. This information is then used to periodically build the users database (the `grid-mapfile`) on Globus hosts.

The server provides only information on the access policy, while the final authentication is done by the Globus host or computing resource. A *network* service for authentication could lead to possible denials of service in the grid environment.

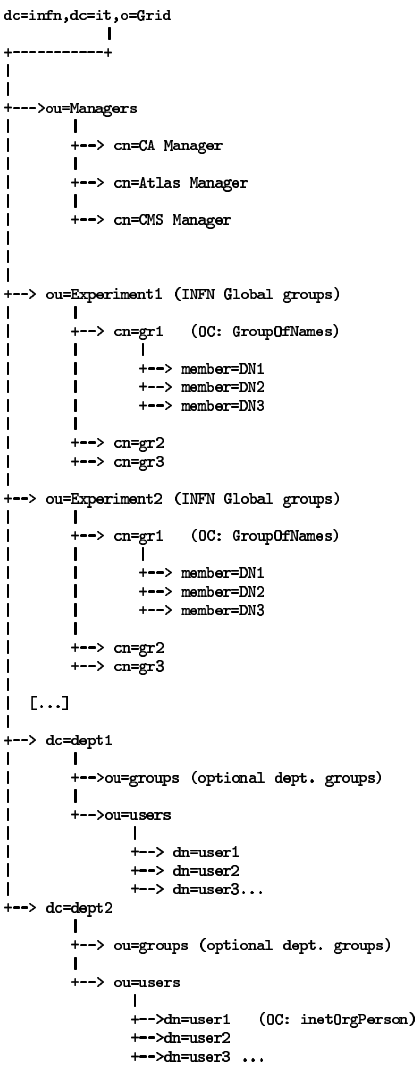
In the Globus model, authentication tokens are X.509 user certificates that in our case are produced by the INFN Certification Authority. Users are identified by their certificate subject, that is mapped to a local unix account by the `grid-mapfile`. The missing link is then a repository that acts as a server for user certificates (subjects) and has the ability to manage and share groupings of users to the INFN GRID testbed.

In order to easily integrate this service into the existing Globus information framework, the user/group information is stored in an LDAP server, that uses the GIIS namespace and that contains users description (with certificates) and users groupings. The Globus (standard) domain component based namespace is used.

The information in the server must also use RFC-standard objectclasses as far as possible, as this allows easier integration of the system with existing software (for, say, user authentication).

The objectclasses that best represents users in the INFN-GRID are the *person*, *organizationalPerson* and *inetOrgPerson* [15] [16] objectclasses. The combination of these objectclasses (see Appendix A for a detailed description) allows authentication with a password and the storage of an X.509 user certificate. Groupings of users can be defined using the *groupOfNames* objectclass. The Member is a multi valued attribute that contains a list of distinguished names of users belonging to the group. The *certificationAuthority* objectclass can also be used to store the certificate CA and the CRL.

The following tree layout can then be implemented:



This namespace allows a clean access control list implementation, and a directory partitioning based on a geographical model.

3.4.1 Maintaining the repository

The process of maintaining the repository involves the following steps and actors:

1. The **CA manager**, produces authentication information (certificates) and publishes this info in the repository.

   The CA manager MUST be able to add/modify users in the directory, this can be done using a tool that accepts a certificate and publishes it to the directory. A

mapping rule between the certificate and the DN under which it will be published is a common technique to simplify this process.

At the moment INFN uses X509v3 certificates and, since X509v3 certificates must contain an email address attribute (not the one in the subject, needed only for X509v2 compatibility). Since the email address uniquely identifies an user and contains the same domain component information as the standard DNs that are used by the Globus or the INFN CAs, the email contained in the certificate will be used to produce the DN as in the following example:

`Giuseppe.LoBiondo@mi.infn.it`

becomes

`dn: mail=Giuseppe.LoBiondo@mi.infn.it,dc=mi,dc=infn,dc=it,o=Globus`

(note that this DN uses the Globus namespace)

2. **Organizational Unit Managers** are responsible of editing OUs Groups, creating new ones and editing memberships. As discussed above, Groups are implemented in the directory through the groupOfNames objectclass. Groupings can be used to produce gridmap files as well as for other administrative purposes.

3. **LDAP Managers** have full access to the directory, create the Directory layout and assign privileges to Groups Managers and to the CA manager.

3.4.2 Using the repository

The service is used by **Globus administrators** that can use the information on the directory to update periodically the gridmap file using their preferred policy.

A tool for Globus administrator should be able to:

1. connect to the server and download selected certificates choosing a policy to implement (all,group,domain etc..).

2. Check if the certificates are valid using the CA certificate and the Certificate Revocation List.

3. Produce grid-mapfile lines.

3.4.3 Integration with Globus

Using the command switch `-publish-users` of *globus-gram-scheduler* it is possible to publish gridmap files on the web (this feature didn't work correctly in the Globus v1.1.3 distribution, but INFN-GRID provided a fix). This can be used in conjunction with the *grid-mapfile* update system we are describing, so that users may know which resources are accessible to them.

3.4.4 Security Issues

Even if the grouping information of the repository is used only indirectly for authentication, this sub tree must follow a restrictive security policy.

1. Accessible only from Globus hosts (registered in the MDS)
2. Transport Layer Security (TLS) should be used for maintenance operations (cert publishing, group editing, where password are sent over the net) and for searches when possible.
3. Even If the X.500 model provides an objectclass to contain CA certificates and CRLs, the repository is not the right place to hold this information that should be provided by the CA independently. (An attack to the server that changes the CA certificates and users certificates can lead to false authentication assumptions).

Access control lists (ACLs) to estabilish managers privileges on the Directory Information Tree (or DIT) must be implemented. Unfortunately no standard ACL schemas exist yet, (standardization is ongoing), so the software specific ACL schema must be used.

3.4.5 Tools

Two tools were prototyped to deploy this directory service: one that allows the CA to publish certificates and one that allows Globus administrators to create gridmap files automatically. Prototypes for these tools were developed in the framework of the INFN-GRID Globus Evaluation activity.

Here is the prototype syntax:

`certpublish` publishes user certificates, it can be used by the CA manager to publish certificates.

```
certpublish
```

```
-in <filename>   : DER Encoded Certificate to publish
-host hostname   : Name of the server, (default: gridmap.infn.it)
-port integer    : Port Number, (default: 389)
-base DN         : Base for searches, (default: dc=infn,dc=it,o=Grid)
-DN DN           : Bind dn (default: cn=Manager,dc=infn,dc=it,o=Grid)
-help            : This help
```

certretrieve can be used by globus administrators to create gridmap-files on the fly using information contained in the LDAP server.

```
certretrieve

-host hostname   : Name of the server, (default: gridmap.infn.it)
-port integer  : Port Number, (default: 389)
-base DN  : Base for searches, (default: dc=infn,dc=it,o=Grid)
-DN DN    : Bind dn (default: anonymous)
-groupDN groupDN : If present return only user in the group
-lcluser user  : User to map certificates (default: glbusr)
-CAfile filename : If present check certificate validity
-CRL  filename   : If present check CRL
-help  : This help
```

Group Managers can edit groups using many existing LDAP tools. No development is needed here.

Certificate subjects can be retrieved for example with the following command:

```
certretrieve -groupDN "cn=gen,ou=CMS,dc=infn,dc=it,o=Grid"
```

This will retrieve certificate subjects of users in the gen subgroup.

NB: since users can belong to several groups sometimes it may be necessary to merge the results.

3.4.6 Group Editing

Groups for a specific activity (for instance an experiment) are created by the appropriate administrators and look like this:

```
# gen, CMS, dc=infn, dc=it,Grid
dn: cn=gen, ou=EXP, dc=infn, dc=it,o=Grid
objectClass: top
objectClass: groupOfNames
cn: gen
owner: mail=manager1@xx.infn.it,ou=people,dc=xx,dc=infn,dc=it,o=Grid
owner: mail=manager2@yy.infn.it,ou=people,dc=yy,dc=infn,dc=it,o=Grid
owner: mail=manager3@zz.infn.it,ou=people,dc=zz,dc=infn,dc=it,o=Grid
ou: EXP
o: Grid
businessCategory: HEP Experiment
member: mail=user1@zz.infn.it,ou=people,dc=zz,dc=infn,dc=it,o=Grid
member: mail=user2@yy.infn.it,ou=people,dc=yy,dc=infn,dc=it,o=Grid
member: mail=user3@tt.infn.it,ou=people,dc=tt,dc=infn,dc=it,o=Grid
```

The `owner` attribute determines who is allowed to update the `member` attribute. Using OpenLDAP tools group Administrators are able to:

1. Add new members to the group.

   First it is necessary to create an LDIF file (for example `modify.ldif`) such as the following:

   ```
   dn: cn=gen, ou=EXP, dc=infn, dc=it,o=Grid
   changetype: modify
   add: member
   member: mail=user1@hh.infn.it,ou=people,dc=hh,dc=infn,dc=it,o=Grid
   member: mail=user2@kk.infn.it,ou=people,dc=kk,dc=infn,dc=it,o=Grid
   ```

   then the following command can be used to add the new member attribute to the group entry.

   ```
   ldapmodify -h bond.cnaf.infn.it \
   -D"mail=manager1@xx.infn.it,ou=people,dc=xx,dc=infn,dc=it,o=Grid" \
   -W \
   -f modify.ldif
   ```

2. Modify an user password.

   The manager password can be modified using the `ldappasswd` command:

```
ldappasswd -x -h ldapserver.xx.infn.it \
-D"mail=Manager1@xx.infn.it,ou=people,dc=xx,dc=infn,dc=it,o=Grid" \
-W -S
```

Note that the command syntax for `ldapmodify`/`ldappasswd` is from OpenLdap 2.0.7. Many other graphical or web interfaces can be used to edit the LDAP database contents. In particular we tested five GUI Ldap Clients Tools in order to find one to recommend to the Organizational Unit Managers. They are:

1. LDAP Browser/Editor by Jarek Gawor (Mcs/Anl)
2. GQ by different authors (`biot.com`)
3. Frood by Jenni Bennet (`sourceforge.net`)
4. Kldap by Oliver Jaun (`mountpoint.ch`)
5. Ldap Browser by Softerra

We chose the tool developed by Jarek Gawor because it satisfies these requirements:

- It is operating system independent because is written in Java.
- It supports SSL.
- The browser can be run as a signed or unsigned applet within a web browser.

3.5 AFS tests

As many current experimental activites within INFN use the distributed Andrew File System (AFS), we also needed to evaluate the compatibility of the existing Globus software with AFS. We will start by noting that AFS is also based on a one-time-login mechanism (Kerberos Version 4), where user ”proxies” (also known as AFS ”tokens”) are issued by an appropriate granting entity (network server) upon verification of a user password. AFS tokens also give general scope access to the distributed filesystem with the same rights of the requesting user, so we need to understand how an AFS token can be obtained by a remotely-executing Globus job.

3.5.1 What can be done with the existing Globus tools ?

The Globus ”gatekeeper” is able to execute an arbitrary ”login” command immediately after starting. This is done via the `globus-k5` program, which is invoked when the gatekeeper is started with the `-k` option. The ”login” command is associated to a Globus user ID (X509 DN) via another mapping file, the so-called *globuskmap* file. The name of the *globuskmap* file needs to be specified as a gatekeeper option as well (`-globuskmap` *filename*). In the standard Globus installation the options have to be added to the *Globus deploy dir*`/etc/globus-gatekeeper.conf` file.

When an AFS token is needed, one could just naively obtain it via an appropriate command in the *kmapfile*. For instance:

```
"/C=IT/O=INFN/L=Milano/CN=Francesco Prelz/Email=francesco.prelz@mi.infn.it"
/usr/afsws/bin/klog -principal prelz -password ******* -cell infn.it
```

The password needs to be specified in cleartext. The *kmapfile* can be made only readable by root, but this is obviously not an acceptable solution. The token lifetime can be specified with the -lifetime option once for all only by the remote system administrator. There’s no provision for reissuing tokens (the Kerberos V4 token validity cannot be ”extended”).

Another caveat: even if the system is able to obtain an AFS token, it’s up to the user to issue the approriate unlog command to remove valid tokens from the executing machine cache.

This way of accessing AFS was tested with the current Globus toolkit and was found to be working. This is however quite unfit for any real need, and a very bad idea.

3.5.2 ‘Correct’ ways to obtain an AFS (Kerberos V4) token

Kerberos V5 provides ways to transfer and extend tickets, plus a GSS-API implementation that could be used in place of the Globus GSS-API (see Section 3.1.1). The trouble is that Transarc (commercial AFS) has no plans to adopt Kerberos V5, and it’s not clear whether OpenAFS [17] will ever get there.

A rather makeshift tool, derived from MIT work (`ak5log`) can build a Kerberos V4 AFS token based on Kerberos V5 authentication information. This assumes there’s a Kerberos V5 authentication infrastructure in place, and INFN doesn’t have one in place. We have shown in Section 3.2 that the task of running *one* institutional certification infrastructure is already challenging enough, and definitely not worth an effort duplication.

We could however use another Globus tool (`sslk5`) that provides a client/server pair where the server generates Kerberos V5 tickets based on Globus (X509) credentials. If access to the user information AND to the AFS/Kerberos private key is available, one

could in principle generate a V5 ticket based on the AFS private key, then convert it to a V4 AFS token. We couldn't test this tool within the INFN AFS infrastructure, but tests are underway at CERN.

3.5.3 The most 'correct' way for accessing distributed data

Kerberos and GSI are actually two solutions to the same problem. In an ideal world we should have just one of them. The grid "should" (and possibly will) provide the same level of delocalized file access that AFS provides.

Chapter 4

Information Services for the Grid

The Information Service (IS) plays a fundamental role in the Grid environment, since resource discovery and decision making is based upon the information service infrastructure.

Basically an information service is needed to collect and organize, in a coherent manner, information about grid resources and status and make it available to consumer entities.

Areas where information services are used for the grid are:

- **Static and dynamic information about computing resources**: this includes *host configuration and status*, a list of services, protocols and devices that can be found on hosts, access policies and other resource related information.
- **User Information**: User descriptions and groupings need to be available to client systems for authentication purposes. In the Grid environment, this is mostly a service provided by the Public Key Infrastructure of every participating institution.
- **Network status and forecasting**: Network Monitoring and forecasting software will use an information system to publish network forecasting data.
- **Software repositories**: Information on where software packages are located, what they provide, what they need etc. needs to be available on the net.

Hierarchical directory services are widely used for this kind of applications, due to their well defined APIs and protocols.

Anyway, the main issue with using the existing directory services is that they are not designed to store dynamic information such as the status of computing (or networking) resources.

4.1 The Globus Grid Information Service

The Globus *Grid Information Service* provides an infrastructure for storing and managing static and dynamic information about the status and the components of computational grids. The Grid Information Service implemented by INFN is based on the Globus package, so here we'll refer to the *Globus* GIS software.

The Globus Grid Information Service is based on LDAP directory services. As we pointed out, LDAP is not the best choice for dynamic data storage, but provides a number of useful features:

- A well defined data model and a standard and consolidated way to describe data (ASN.1).
- A standardized API to access data on the directory servers.
- A distributed topological model that allows data distribution and delegation of access policies among institutions.

In the next sections we'll discuss directory services related topics and how they apply to the Globus grid information service that was set up for INFN.

4.1.1 Data Design

There are many issues on describing the data that the information system must contain. Basic guidelines for the GIS are:

- The information system must contain useful data for real applications (not too much, not too few).
- It is important to make a distinction between static and dynamic data so that they are treated in different ways by the various servers in the hierarchy. This can be done with the Globus package assigning different time to live to entries.

Unfortunately there are some lacking features in the Globus data representation: though LDAP represents data using ASN.1 with BER encoding (a good reference book for ASN.1 is [21]), the Globus v1.1.3 GIS implementation has no data definition types tied to attributes: all the information is represented in text format and the proper handling of attribute types is left to the backend.

As a consequence of this, it is not straightforward to apply search filters to data (for example numerical values are compared as they were strings).

4.1.2 Schema Design

LDAP schemas are meant to make data useful to data consumer entities. The schema specifies:

- An unique naming schema for objectclasses (not to be confused with the namespace).
- Which attributes the objectclass must and may contain.
- Hierarchical relationships between objectclasses.
- The data type of attributes.
- Matching rules for attributes.

The schema is then what makes data valuable to applications. Globus implemented its own schema for describing computing resources.

As with other LDAP servers, the Globus schema can be easily modified. Anyway this process must be lead by an authority, in order to make schema modifications consistent among various GIS's infrastructures.

From our perspective the Globus schema should become a standard schema to describe computing resources with the aim to make it worth for a wide range of applications and to allow an easier application development (applications must refer to a schema to know how to treat data).

4.1.2.1 The Globus Schema

The Globus schema is used to describe computing resources on a Globus host.

We think that the current schema needs to be integrated at least with a description of:

- Devices (and file systems)
- Services (other than the services provided by Globus)

The definition and standardization of information about computing resources is largely work in progress ([18]) and the upcoming grid development efforts will have to make sure to feed back their proposals for extensions to the information schema into the standards process.

4.1.3 INFN Namespace

The domain component (dc) based naming schema used by Globus was adopted by the INFN-GRID project since it fully maps our geographical arrangement of servers and administrative policies. The analogy with the DNS also makes it generally more familiar.

4.1.4 INFN Infrastructure and Topology

Since the beginning of the INFN-GRID project, we had to cope with two different incarnations of the Globus MetaDirectory System (MDS), namely those included in Globus releases v1.1.2 and v1.1.3.

At the beginning, the Globus US-based MDS server (a Netscape Directory Server), seemed too slow and unreliable for transatlantic access, so we decided to have a child INFN server referred from the main server in order to gain better performance. The INFN MDS server was also based on the Netscape Directory Server.

This first MDS infrastructure was used by INFN sites, on a national scale, however the testbed size was small (about 20 hosts).

At that size the service was very fast and reliable, anyway the MDS v1.1.2 model had an administrative overhead, since every client site had to receive a username and password from the LDAP manager in order to access the INFN LDAP server (every INFN site had to enroll to the central MDS). In the v1.1.2 model, the server was also overloaded by client data publishing (each host pushed their data every 5 minutes).

With the v1.1.3 release, the Globus Project made major changes to the MDS architecture, to achieve better scalability. Open source LDAP servers (OpenLDAP) were used.

No MDS enrollment is needed in Globus v1.1.3 and no authentication mechanisms are provided by the LDAP servers. The multi-central MDS model (one LDAP server per organization) was replaced by the new, distributed MDS (GIS) architecture.

In Globus 1.1.3 every resource runs an LDAP server (GRIS) that publishes only a referral pointer (the registration) used to locate resource related data.

GRIS's automatically register themselves to site index LDAP servers (GIIS) that are used to collect information about resources.

Registration are also sent periodically (5 minutes by default) to inform the higher lever GIIS about the reachability of lower level GIIS's and GRIS's.

Using Globus 1.1.3 the INFN-GRID pilot has then implemented a hierarchical partitioning based on geographical entities (INFN departments). Each site runs a Grid Information Index Server (GIIS) that registers itself to a top level INFN GIIS.

Local GRIS's are registered at the site GIIS (and not directly to the root server).

Registrations to superior knowledge servers (site and root GIIS's), provides the superior GIIS a hook used to implement chaining on the corresponding tree partition.

Chaining results are cached by GIISes when a query is performed and returned to the clients querying the server. The cache expires less often at higher tree levels giving only a less dynamic view of the Grid: more up-to-date data can be provided by lower level ("department") GIIS servers and by GRISes.

See also [25] for deeper technical details of the INFN GIS configuration.

Though using this *cached chaining* may seem inadequate, it is at the moment the best option to deal with dynamic data within an LDAP directory service.

Other techniques like LDAP smart referrals (that put more burden on the clients) and chaining with no caching (that overloads the server) are likely to be poor in performance.

GRISes (tree leafs) can also register themselves on other institutions GISes depending on usage policies. GIIS's instead can register themselves only on a superior server that uses a consistent namespace. This means that a GIIS cannot register itself to a superior knowledge server that uses a different naming schema.

GRISes registering themselves to different GIISes allow the creation of several virtual organizations (such as HEP experiments) within the same set of resources.

Although possible, topological loops should be avoided and its usage is discouraged since it goes against the LDAP naming model.

4.1.5 Extending the GRIS

A good flexibility in extending the set of data returned by each GRIS in the GIS architecture is a definite requirement.

The GRIS currently ([19]) uses *trigger* programs to fetch data from the system. Such data is then cached for a configurable amount of time. The output of trigger programs must be in LDIF format and must respect the Globus schema. The schema can be extended to represent information other the those provided by default. In some case we concluded that a different definition of some standard information "keys" was needed. In the short term, we hope that such changes can be negotiated in the framework of our collaboration with Globus. In the long term the entire information modeling schema will probably evolve.

The Globus LDAP schema, at the moment, is not checked at runtime by the GRIS (LDAP server) to ensure data consistency.

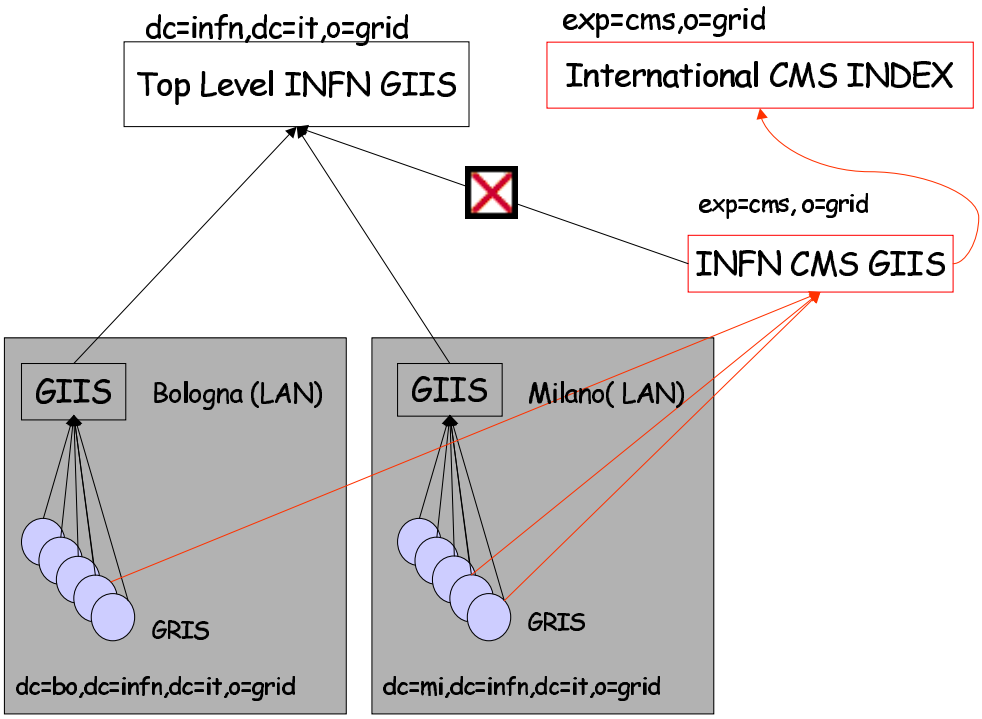


Figure 4.1: Hierarchy of Globus v1.1.3 (GIS) information services within the INFN-GRID test organization.

4.1.6 Service Reliability

When the geographically distributed configuration described in Figure 4.1 is implemented, it is important, in terms or service reliability, that each INFN site GIIS be able to operate independently from other INFN sites and from the root INFN GIIS server. This means that GIS clients at each site must be able to rely only on the local GIIS (as it happens, for example, with DNS clients, that don't lose the local visibility of the DNS when the root name servers are not reachable).

In the LDAP standard way to obtain this, the site GIIS server can hold a superior knowledge reference to its ancestor (a default referral to the root INFN GIIS) and return it to the clients (that can point themselves to the local GIIS) if they wish to know about global (INFN) resources.

This reference can be contained in the root DSE (DSA Specific Entry, a server private entry containing operational information) of the local LDAP GIIS server (see Figure 4.2. Clients can look there to learn where the INFN root GIIS is located (using an LDAP URL that specifies the host FQDN and the LDAP port) if they wish to know about non-local resources. This increase in reliability is easy to implement using LDAP native capabilities, but is not mentioned in the server documentation.

4.1.7 Data Caching

At the moment the root GIIS server contains the same set of information available at lower levels, but the caching of dynamic data at the top level is often useless. Furthermore, there is some information that is convenient not to publish for security reasons: users on a host, software installed etc.

The simplest way to limit the propagation of this "classified" (and useless, if it is too dynamic) information on higher hierarchy levels is to implement access control on GIIS's in such a way that superior GIIS servers can access only a portion of the information, while authorized hosts can access the whole information. At the moment, the GIS does not implement any security mechanism.

As stated earlier, the time to live of static information that should be available at higher levels has to be greater than the time to live of dynamic information.

As of now, the data provided by the root server is the same that inferior servers can provide, but expires less often (1 hour versus 5 minutes).

4.1.8 WEB Tools

To simplify the GIS directory tree navigation , a very simple web interface has been made available at the URL:

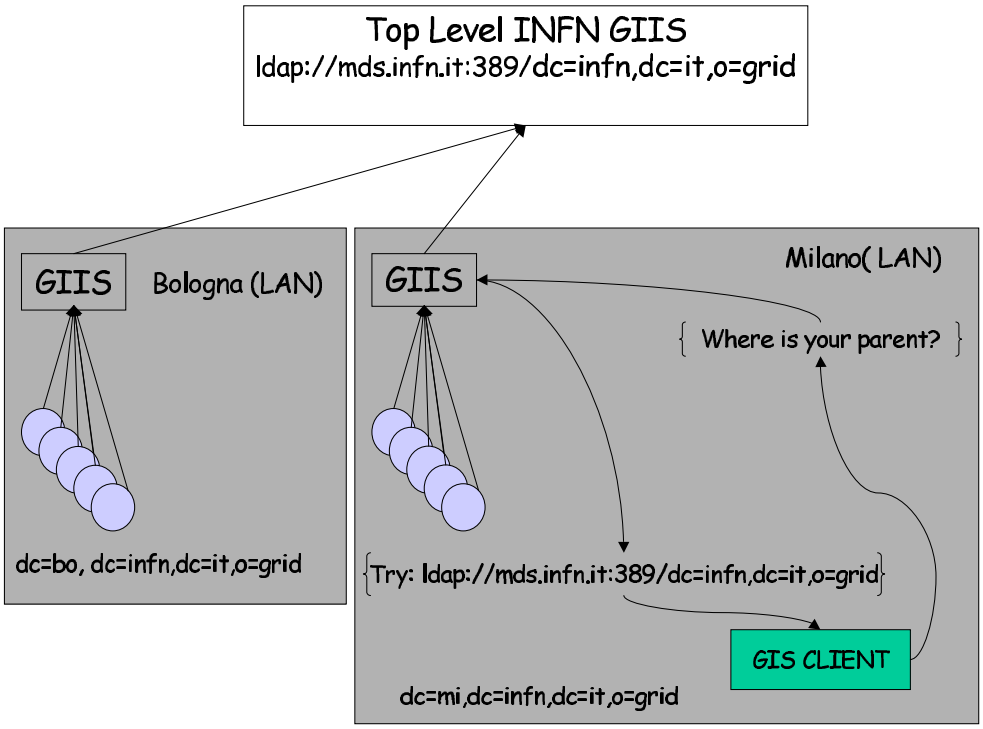


Figure 4.2: When building a geographically distributed (directory based) information tree, it is important to make sure that each site can get a reliable and consistent view of the local information in case the geographical links fail.

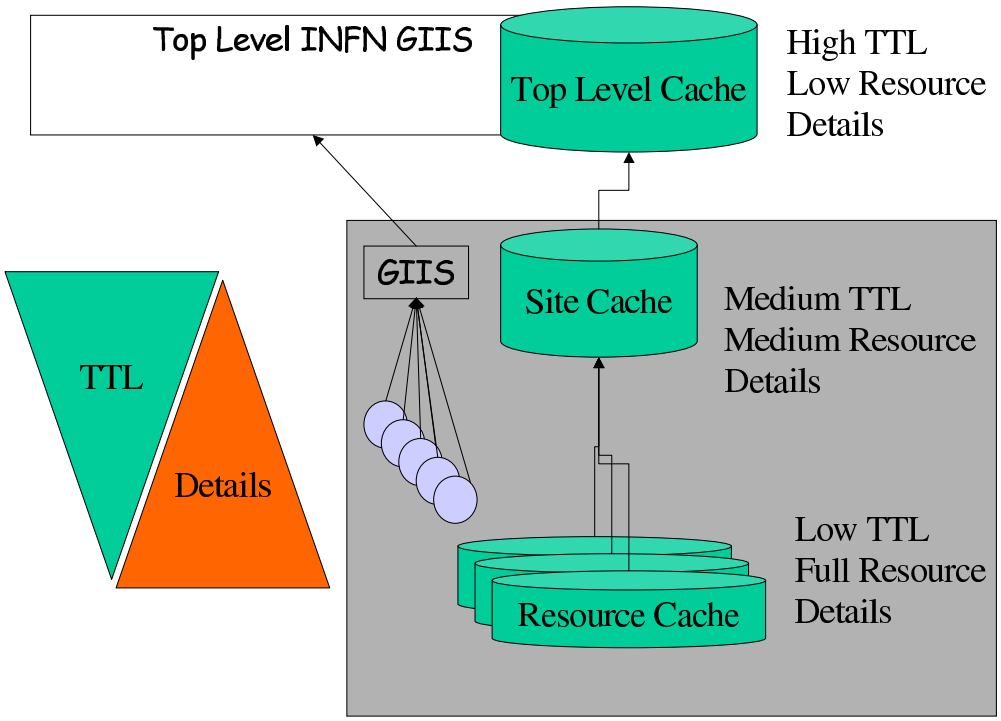


Figure 4.3: Detailed view of the GIS cache.

Figure 4.4: Snapshot of the GIS Directory Information Tree.

`http://bond.cnaf.infn.it/cgi-bin/mdsbrowse1.pl`

The CGI perl script uses the perl Net::LDAP module ([20]).

4.1.9 A Possible Approach to Resource Discovery

The main customer of the GIS framework is a Grid resource discovery system.

Typical queries on the Grid involve a search on the whole Directory Information Tree, causing on the worst case the entire set of servers to be queried. (For example a typical query may look for ”Linux PCs with 128 MB of RAM”)

Information retrieved may also not be up-to-date enough.

A proposed mechanism to obtain the information that we want can be described as follows (compare Figure 4.5):

1. Once the client client found the root server, it submits its query to it.

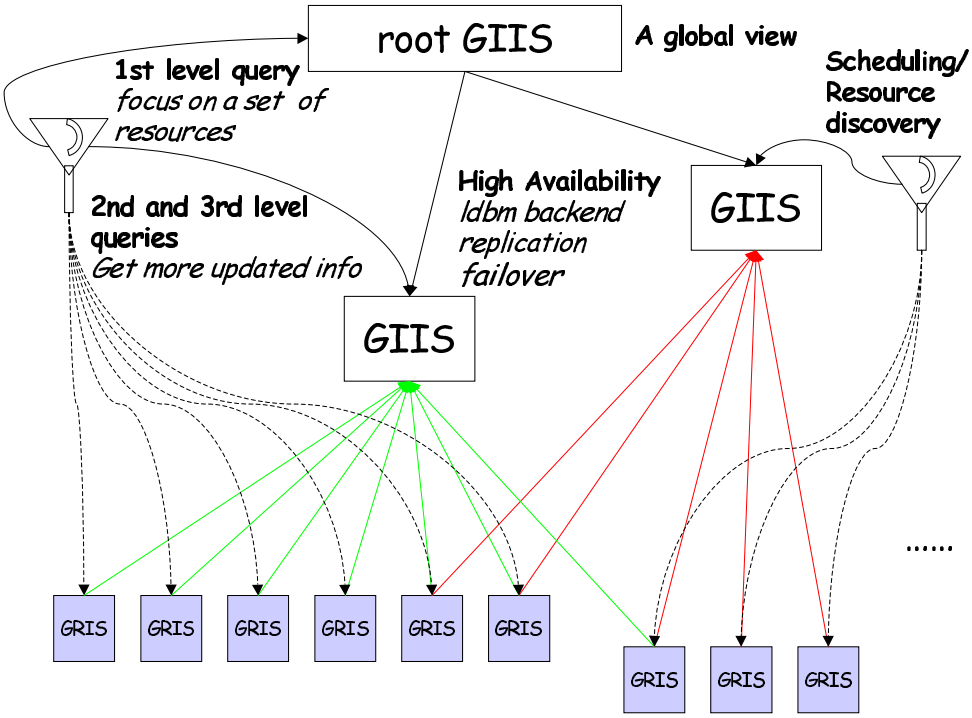


Figure 4.5: Role of the GIS in a possible approach to resource discovery. A “top” level query is made to the “root” (organizational) index server just to get a list of available resources. The search is then refined with lower level (“focusing”) queries made to information servers that are closer to the actual information providers.

2. The top level GIIS, can then return a set of possible candidates, using the static attributes that the query contains.

3. The client uses this set of information to narrow the search and to obtain more valuable data querying the root GIIS children’s.

4. The client uses the new set of data to further narrow the search on GRIS’s and to obtain privileged data inaccessible at higher levels.

Anyway at the first query instance the whole namespace has to be searched, Since GIS servers have no knowledge on where data can be found (or mined for) the first query

needs to be directed to the root server. This causes the whole namespace to be searched, and this means that the children of the root server need to cooperate for the final result. In the current GIS implementaton, this can take unacceptable times when the root server cache is expired. As we confirmed during our tests, the slowest GRIS found during the DIT upload into the root server determines the query response time.

Techniques to implement *search space pruning* could be investigated to reduce the number of servers contacted to answer a query. A possible approach can consist in maintaining a centroid index to be used to restrict the search scope. Such index can be used to point searches to the servers where data that we are actually searching is stored (we may index, for example, GIIS's using the host architecture attribute to speed up searches based on the computer architecture).

The Common Indexing Protocol (CIP) [24] may be used as the standard way to distribute indexing information among GIS servers.

4.1.10 Data Replication

Replication as a mechanism for automatic clients failover must be available at all the levels of the servers tree.

At the moment, anyway, LDAP replication is not standardized, although it may work with some arrangements between different servers. Another approach is server synchronization using custom scripts.

An automatic failover mechanisms based on the DNS can be in place to properly use the replications. A proposed approach is in RFC 2782 [22].

Synchronization will be implemented at the root level initially, as a backup for the INFN GRID to address possible GIS problems that may arise in this deployment phase. The top level synchronization server is a Netscape LDAP server, a synchronization script fetches the data from the MDS and pushes them into the Netscape server periodically. This script should be server independent and not rely on server specific replication mechanisms.

4.1.11 Security and Access Policies

Security is an important issue in GIS deployment. Anyway, the GIS at the moment doesn't implement any security mechanism:

- Any host can Register itself to a superior server having a consistent naming schema
- No access control is implemented when searching the GIS

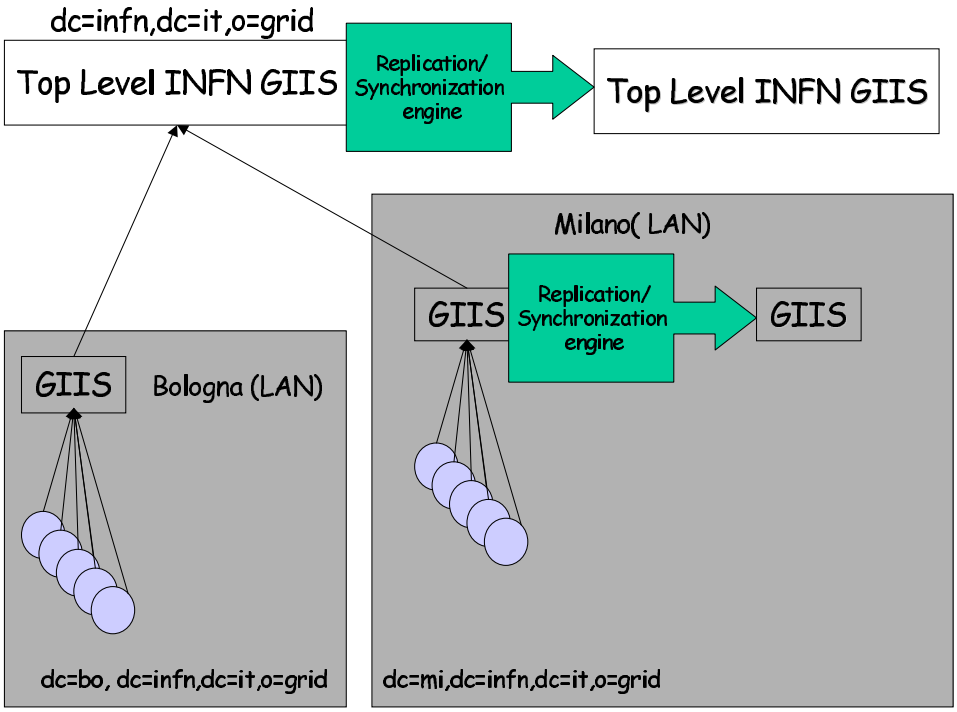


Figure 4.6: Access to the Grid index information servers must be a failsafe process. Appropriate failover (and data replication) mechanisms should be implemented.

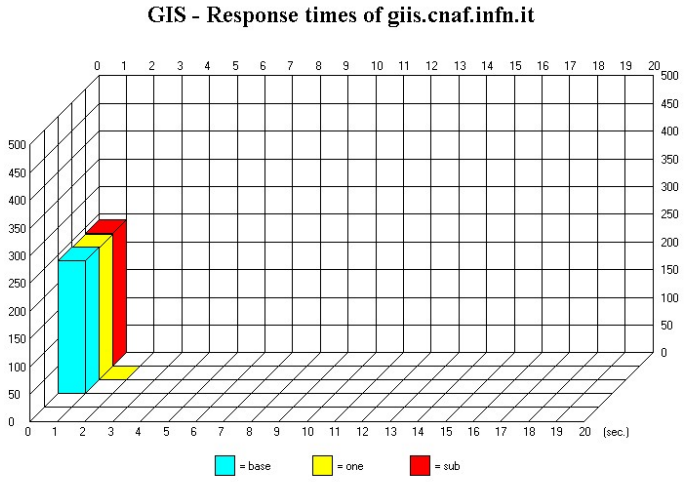


Figure 4.7: Response times measured from the CNAF GIIS.

The ultimate solution to this problem would be a Transport Layer Authentication (SSL+GSI) capable OpenLDAP server that uses a local public certificates database to authenticate the clients.

4.1.12 Monitoring and Performance Tests

Monitoring and performance evaluation is an important issue in this phase of GIS development and deployment. Monitoring systems and a standard GIS benchmark suite (to be shared by GRID projects) can provide valuable information in the GIS development and deployment.

4.1.12.1 Monitoring

A MRTG ([27]) system has been adapted to monitor LDAP servers (GIIS and GRIS) operation in order gather information about server usage.

This web tool uses statistical information published by OpenLDAP servers on a private Directory Information Tree (cn=monitor).

In particular, the “Entries returned per second” and “Connections per second” values are plotted. Unfortunately, no real life usage of the GIS has happened since now, so no useful statistics have been collected.

The MRTG plots are available at the URL:

`http://bond.cnaf.infn.it/gismonitor`

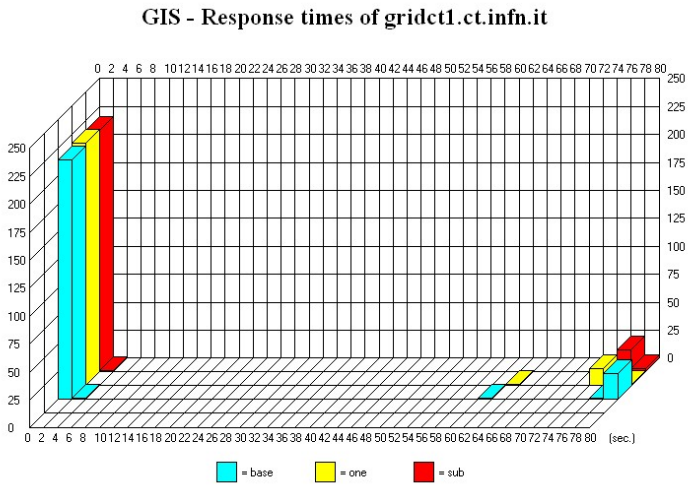


Figure 4.8: Response times from the Catania GIIS.

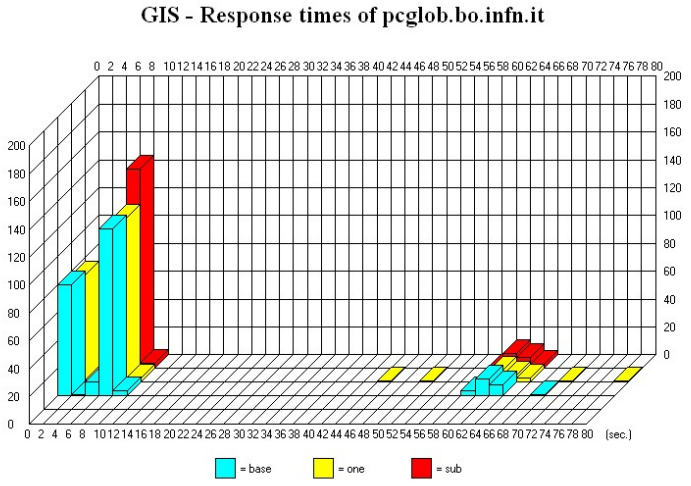


Figure 4.9: Response times from the Bologna GIIS.

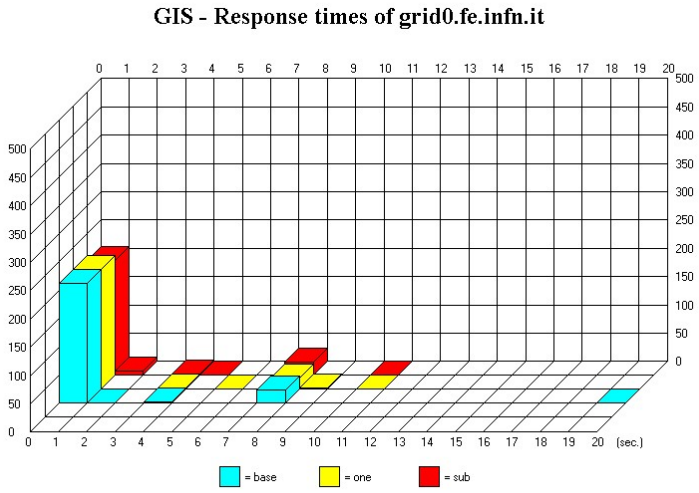


Figure 4.10: Response times from the Ferrara GIIS.

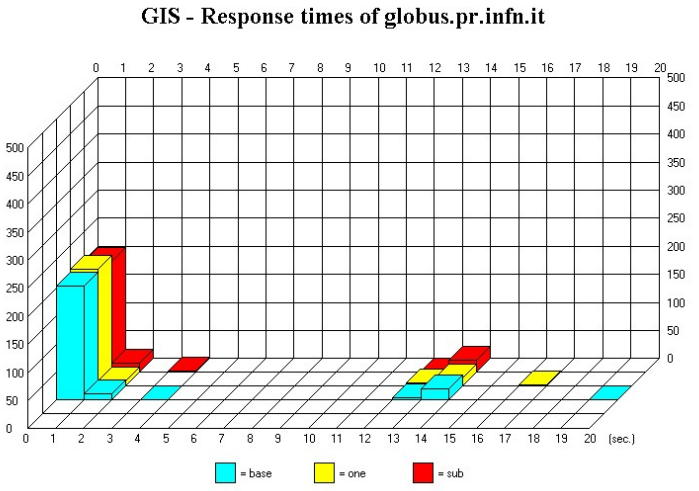


Figure 4.11: Response times from the Parma GIIS.

4.1.12.2 Performance and Scalability Test

In order to test the performance of GIS and how it is affected by the caching mechanisms, we have run the following standard LDAP query from a machine in Bologna (CNAF, were the root GIIS is located) to 4 GIISes located in Bologna,Catania,Parma,Ferrara

```
ldapsearch -h host -p 2167 -b"o=grid" "objectclass=*" -s scope
```

where *host* is the name of the GIIS and *scope* (levels of depth in the ldap tree) is alternatively *base,one,sub*.

We followed this schema for the test:

- we run a set of 10 queries with *base* as scope
- then we waited 20 minutes and sent the same set of queries with *one* as scope
- after 20 minutes we sent the 10 queries with *sub* as scope

We repeated this schema for 24 hours for each GIIS. The GIIS cache expiry period for our GIISes is 5 minutes.

It's expected that for each set of ten queries the first query takes more time then the other nine to get the result, because it is likely that the cache is expired. One example is the following:

```
  base   one     sub
0:13.04 0:16.77 0:13.17
0:00.88 0:00.88 0:01.02
0:00.87 0:00.88 0:00.91
0:00.88 0:00.88 0:00.94
...
```

where the access times (in seconds) are respectively for then base,one and sub scopes.

The graphs in Figures 4.7 to 4.11 show the distributions of access times for the tested GIISes. Note that the last bin includes also the values for data out of range.

As can be seen from the graphs, when the cache is not expired the access time is less than a second.

We have also tried to load the CPU of a GRIS and then send a query to its own GIIS. When the cache is expired, the result is an access time much longer compared to the one we got with no CPU load (98 sec. vs less then 1 sec.).

We have also seen an increase of response time (about 6-7 sec.), even with unexpired cache, when loading the CPU of a GIIS. This may explain the long response time of `pcglob.bo.infn.it`, which is currently used by Condor.

Here's a rule-of-thumb classification of the response time measurements we made on our test machines:

1. values less than a second mean cache not expired and GIIS not busy
2. values around 5-10 seconds mean:
   (a) cache expired and both GIIS and its GRISes not busy, or
   (b) cache not expired and GIIS busy
3. values around or over the minute mean cache expired and GRISes busy

Another thing we can see is that the distribution of latencies over the three scopes is almost the same.

In Figures 4.12 and 4.13 we compare the response times over the three scopes (base, one, sub) for an MDS server (MBASE, MONE, MSUB) and for an OpenLDAP/LDBM (LBASE, LONE, LSUB) server containing the same set of data. The tests were performed on the same machine using the same set of data. The OpenLDAP test used the same LDAP server used by Globus, but with the LDBM backend enabled. The LDBM database was filled with data found in the MDS cache, with no special indexing been enabled. Queries to the MDS server were done to non-expired caches.

The LDAP/LDBM access times are significantly lower than the MDS times. The MDS response times seem to vary very slightly with the scope (the ratio of the average values is $1.00_{(base)}$:$0.96_{(one)}$:$1.00_{(sub)}$), while LDAP times show a dependency on the scope ($1.00_{(base)}$:$3.85_{(one)}$:$5.73_{(sub)}$. Here's a table of the average response times:

MDS - base	**MDS - one**	**MDS - sub**
1.610 s	1.553 s	1.609 s
LDAP - base	**LDAP - one**	**LDAP - sub**
0.0148 s	0.0570 s	0.0848 s

4.1.13 Grid Information System Evolution

The directory based model of the Globus GIS is a workable GIS implementation, that can set a good common standard for access to information on a data grid.

We believe there are still some characteristics of the GIS that need more development (apart from what can be considered just bugs, and that were reported to Globus support):

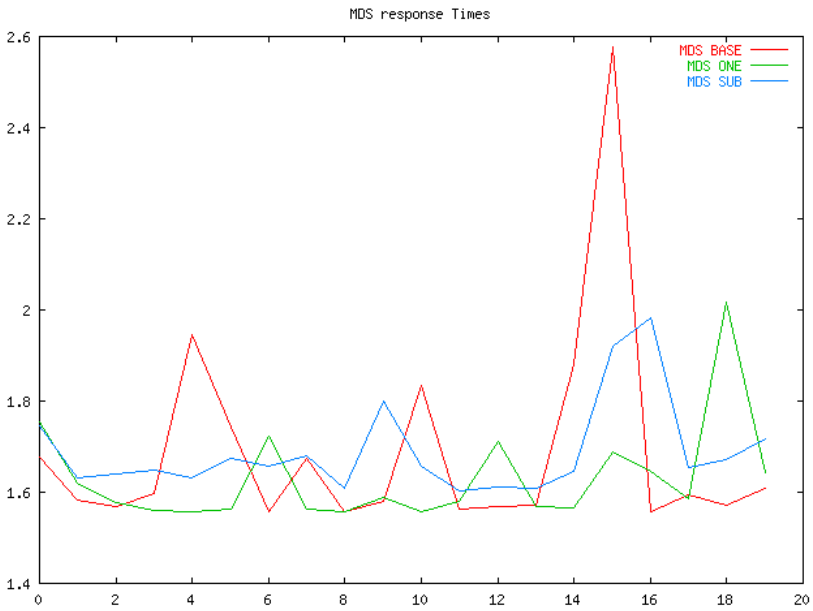


Figure 4.12: 20 measurements of the MDS response times (in seconds) when accessing a certain amount of data.

- The *pull model* (chaining) used for data publishing, in some circumstances is not adequate. A mixed push/pull model is more suitable for real applications.

  In fact there are cases when is needed to make critical information to be available immediately in the GIS. A possible approach is to provide GIS nodes a mechanism to expire data on the path of its ancestors on request.

- The lack of security in the GIS implementation is an important point that must be addressed.

  Registrations are completely anonymous meaning that everyone can register an host on a GIIS. We found that this is a big security problem that can open the path to possible DOS and *man in the middle* attacks.

  Moreover access to data is completely anonymous at all tree levels, this can expose classified data over the net.

- GIS performance, on a geographical scale, is inadequate for a production environment. The main cause for this is to be found in the current shell-based LDAP backend.

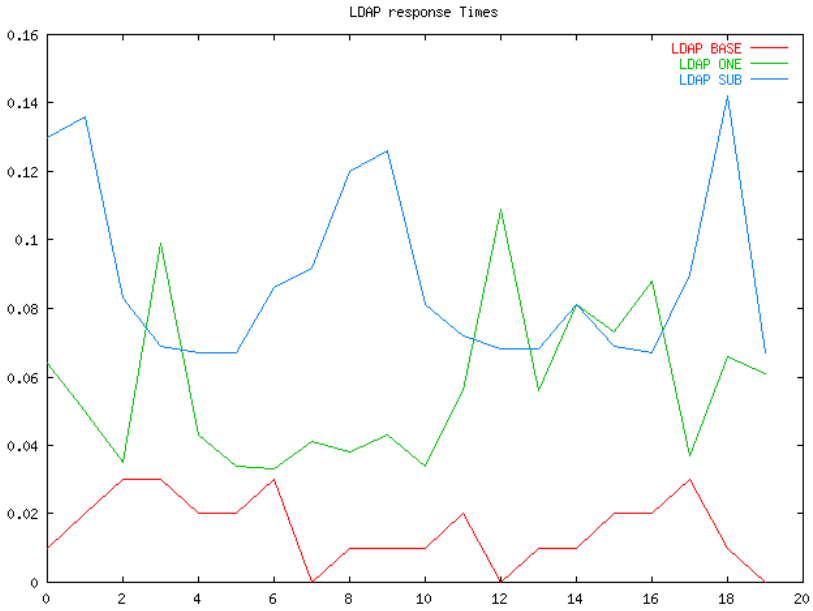


Figure 4.13: Response times (in seconds, 20 measurements) measured from a standard OpenLDAP server with LDBM backend when accessing the same amount of data as in the previous figure.

Keeping in mind that the Grid needs anyway a specialized directory service and that LDAP is a good framework to implement it (in terms of protocols, data structures, data representation and APIs), a possible approach is to implement a custom pluggable LDAP Grid Backend. Both the OpenLDAP and Netscape servers provide APIs to write such backends. These APIs allow to re-implement the inner mechanism of the LDAP server.

Using such APIs it can be possible to implement a Grid backend that implements:

- A standardized and secure registration mechanism.
- A standardized ACL implementation (since now no standard is available for LDAP ACLS, work is in progress in this field [23]), anyway this issue is a more general LDAP server issue.
- Automatic data expiration based, for example on a the *ttl* operational attribute. (ttl is not an operational attribute as now).
- Attribute indexing.
- Distributed indexing.

A relational database capable backend could also be considered a possible solution, since a RDB can limit performance degradation when updating the content of the GIIS servers.

4.2 Conclusions on the GIS

As outlined here, Information Services are a key piece of computational grids since they provide the information about resources needed for discovery and scheduling tasks.

The LDAP Directory based architecture fits the requirements well, as we have seen, but it still needs more work. Lack of fault tolerance, lack of security, lack of data typing, and the unavoidable need for a static server topology are big limitations of the current GIS, at least on a geographical framework.

A lot of design and programming is still needed to have a production ready GIS that suits INFN needs.

Beside the fact the LDAP is good framework to implement GIS's we should consider that going further in deploying directory based systems could take us far off the better approach in the long term since no production test have never been done in large, multi organizational environments, and dealing with such environments requires not only

a very well projected and reliable infrastructure but also a strong cooperation among institutions (in design and access policies implementation, replication and naming schemas planning, etc..) that can be unsustainable for large grid environments.

For these reasons we feel that also non directory based information infrastructures should be evaluated and designed and some man power investment should be done for it.

For example, “web crawler” technology, peer to peer message passing systems, connectionless LDAP and so on can be taken in account to investigate what better fits (or integrates) the grid information system.

Chapter 5

Globus Services for Resource Management

5.1 Globus resource management architecture

The Globus resource management architecture [28][29], illustrated in Figure 5.1, is a layered system, in which high level services are built on top of a set of local services.

The applications express their resource requirements using a high-level RSL (Resource Specification Language) expression. One (or more) broker(s) are then responsible for taking this abstract resource specification, and translating it into more concrete specifications, using information maintained locally, and/or obtained from an Information Service (responsible for providing efficient and pervasive access to information about the current status and capability of resources), and/or contained in the high level specification. The result of this specialization is a request that can be passed to the appropriate GRAM or, in the case of a multirequest (when it is important to ensure that a given set of resources is available simultaneously), to a specific resource co-allocator. Globus doesn't provide any generic, general-purpose broker: it has to be developed for the particular applications that must be considered. The GRAM (Globus Resource Allocation Manager) is the component at the bottom of this architecture: it processes the requests for resources from remote application execution, allocates the required resources, and manages the active jobs. The current Globus implementation focuses on the management of computational resources only.

5.2 GRAM

The Globus Resource Allocation Manager (GRAM) [28] is responsible for a set of resources operating under the same site-specific allocation policy, often implemented by a local resource management system (such as LSF, PBS, Condor). Therefore a specific

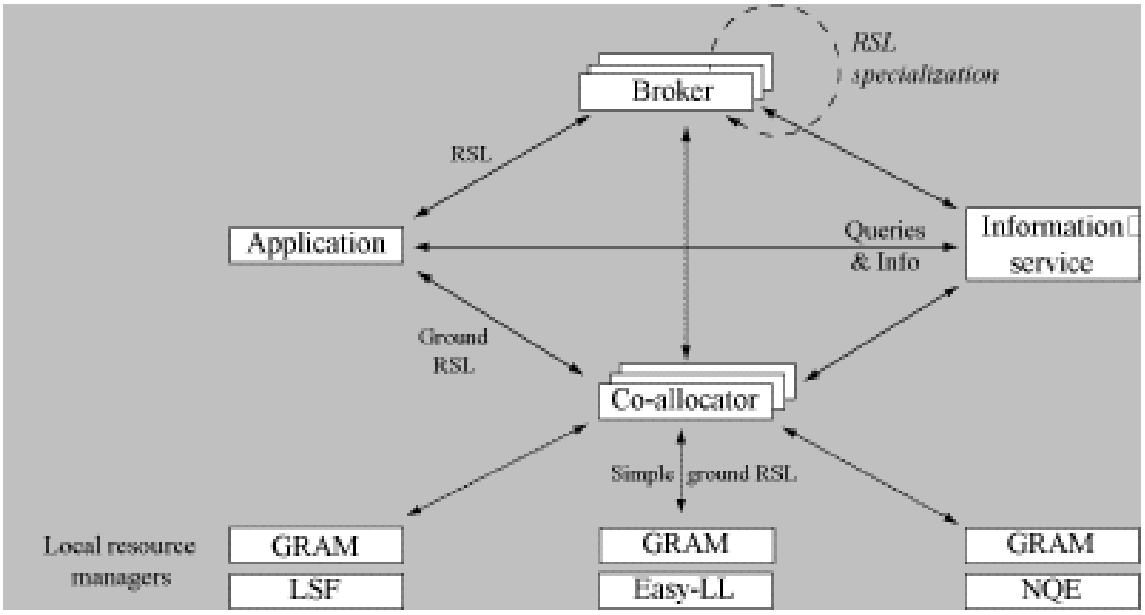


Figure 5.1: The Globus resource management architecture

GRAM doesn't need to correspond to a single host, but rather represents a service, and can provide access for example to the nodes of a parallel computer, to a cluster of PCs, to a Condor pool. In the Globus architecture the GRAM service is therefore the standard interface to "local" resources: grid tools and applications can express resource allocation and management requests in terms of a standard interface, while individual sites are not constrained in their choice of resource management tools.

The GRAM is responsible for:

- Processing RSL specifications representing resource requests, by either creating the process(es) that satisfy the request, or by denying that request;
- Enabling remote job monitoring and management;
- Periodically updating the GIS (MDS) information service with information about the current status and characteristics of the resources that it manages.

The architecture of the GRAM service is shown in Figure 5.2.

The GRAM client library is used by the application: it interacts with the GRAM gatekeeper at a remote site to perform mutual authentication and transfer the request.

The gatekeeper responds to the request by performing mutual authentication of user and resource (using the GSI service), determining the local user name for the remote

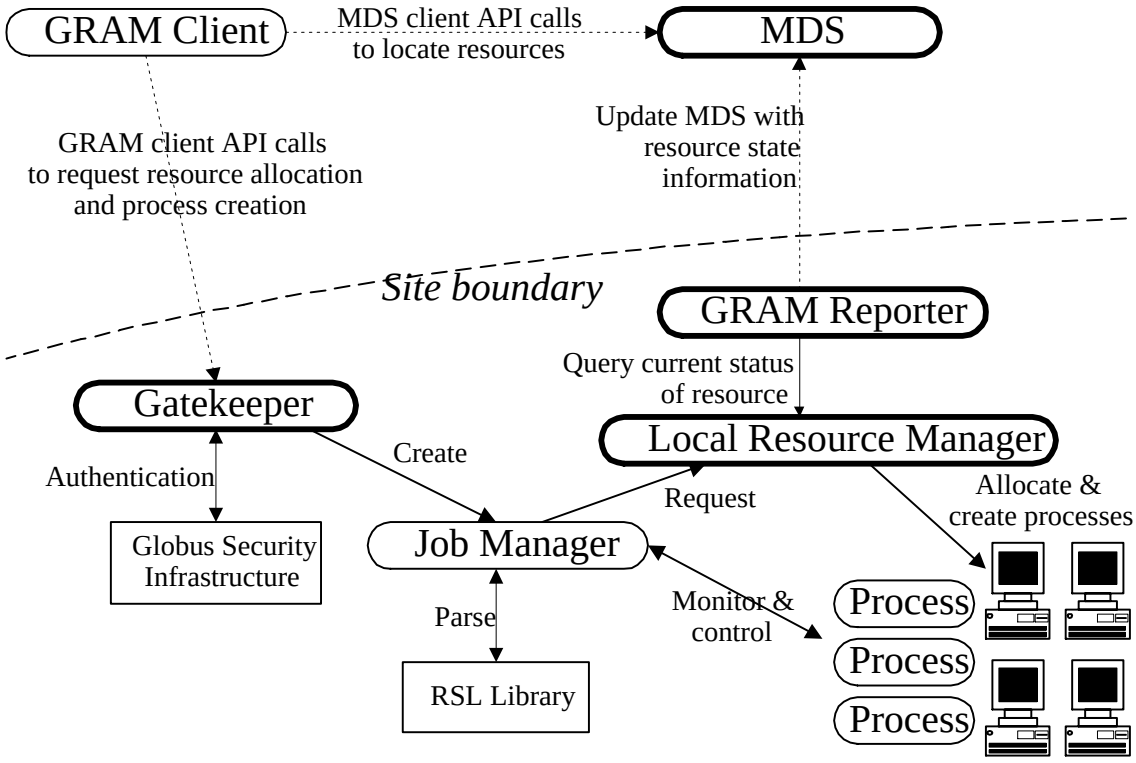


Figure 5.2: GRAM architecture

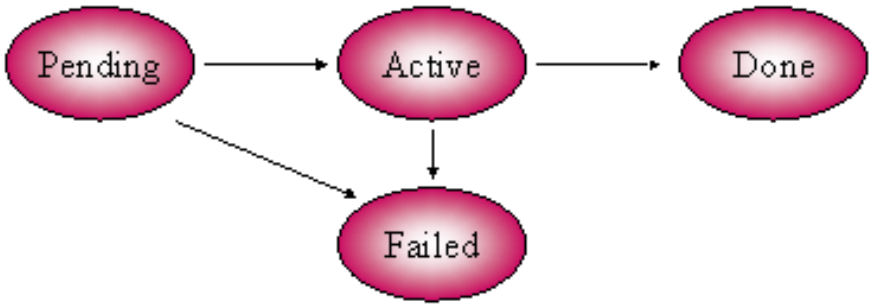


Figure 5.3: State transition diagram for a job

user, and starting a job manager, which is executed as the selected local user and actually handles the request.

The job manager is responsible for creating the actual process(es) requested. This task typically involves submitting a request to an underlying resource management system (GRAM can operate in conjunction with several resource management systems: Condor, LSF, PBS, NQE, etc...), although if no such system exists, the fork system call may be used. Once the process(es) are created, the job manager is also responsible for monitoring their state, notifying a callback function at any state transition (the possible state transitions for a job are illustrated in Figure 5.3). The job manager terminates once the job(s) for which it is responsible have terminated. The GRAM reporter is responsible for storing into GIS (MDS) various information about the status and the characteristics of resources and jobs.

As we proposed in the work plan [6], the Globus GRAM service has been analyzed

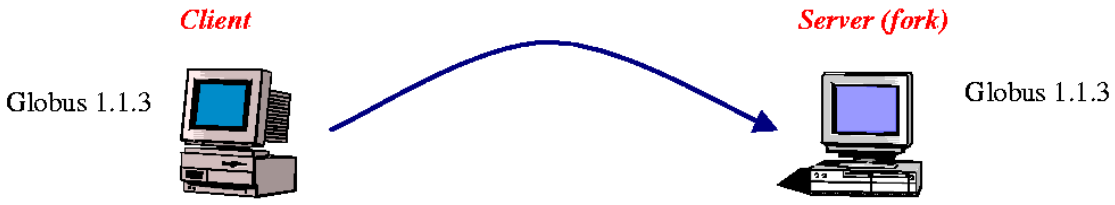


Figure 5.4: Single host as Globus resource

and evaluated, as reported in the following sections. Most of these activities have been performed in collaboration with the Grid Workload Management Work package (WP 1) [2] of the DataGrid project [3].

5.3 Evaluation of the Globus GRAM Service

The tests concerning the evaluation of the GRAM service have been done considering Globus release 1.1.3 (installed using the INFN-GRID distribution, see Chapter 2), using Linux (Red Hat 6.x) Intel PC's.

These activities have been performed in an incremental fashion. As a first step, the Globus functionalities for job submission on remote resources that don't rely upon resource management systems (thus using the fork system call) have been evaluated, considering the configuration represented in Figure 5.4, where a PC has been configured as client (submitting) host, and an other one as server (executing) host [30][31].

A more realistic scenario has then been considered: a PC farm, managed by a local resource management system, has been configured as a Globus resource. Tests have been performed considering LSF [30][31], PBS [32][33] and Condor [30][31] as underlying resource management systems for Globus. As illustrated in Figure 5.5, it has been necessary to install Globus and configure the GRAM service with the considered local resource management system just on a single host of the farm, that plays the role of front-end machine of the cluster.

For what concerns Condor, tests have been performed considering both standard Condor jobs (applications relinked with a specific Condor library, that can exploit the distinguishes features of the Condor system: remote I/O, checkpointing, job migration), and vanilla jobs ("normal" jobs, that don't need to be recompiled and/or relinked). The tests with standard Condor jobs have been done considering as Globus resource the INFN WAN Condor pool, a heterogeneous pool composed by about 200 machines, scattered in

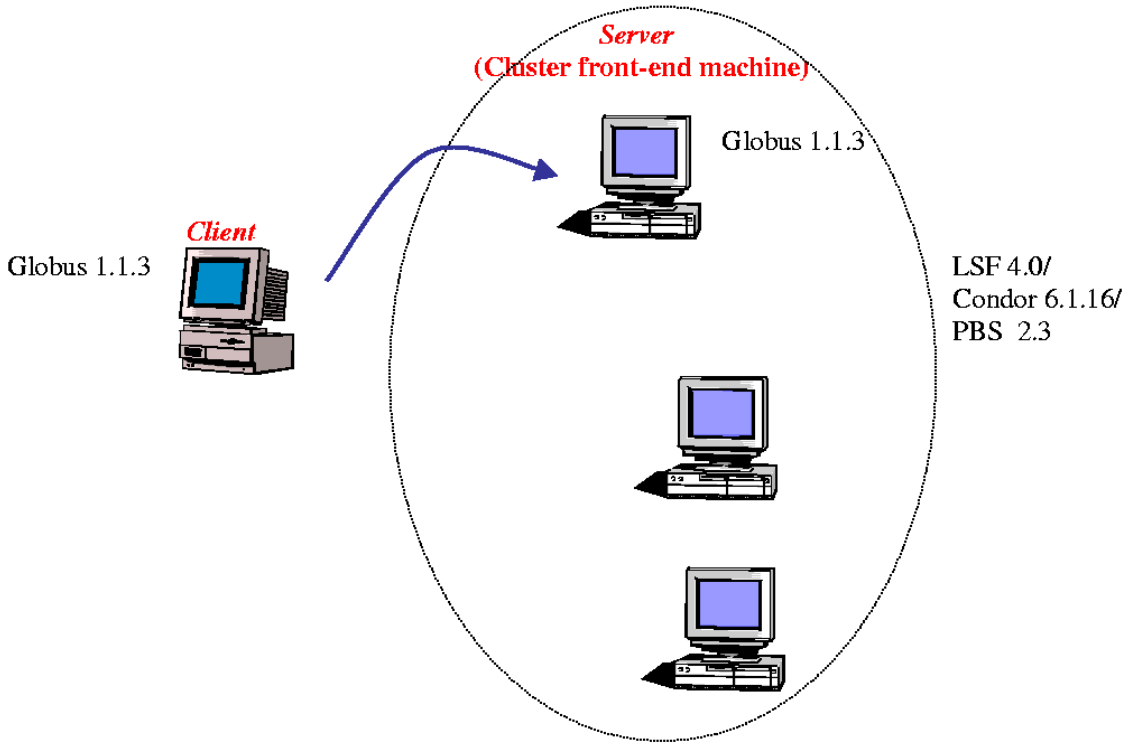


Figure 5.5: A farm as Globus resource

many different INFN sites [34], while for vanilla jobs a layout as the one represented in figure 5.5 has been considered, since in this case a uniform file system and UID domain are required.

For all these tests, the Globus tools (*globus-job-run*, *globus-job-submit*, *globusrun*) [35] have been used to submit jobs on the remote resources, considering different scenarios: submitting batch and interactive jobs, considering the executable and the input file in the file system of the executing machine, staging them from the client to the server host, with the output file created in the file system of the executing/submitting host, etc...

globus-job-run is used to run jobs in the background, while *globus-job-submit* is used for batch jobs. *globusrun* is for submitting jobs, that must run in the foreground or background, specified using the resource specification language. *globus-job-run* and *globus-job-submit* are simply shell scripts that use *globusrun* for job submission. According to the Globus documentation, *globus-job-run* and *globus-job-submit* should present a simpler interface to users than *globusrun*, while in our opinion they don't provide a significant improvement.

The Globus tools for the other job management functionalities (job status monitoring, job removal,...) have been tested as well.

Performing these test, we soon discovered that it is practically impossible to run multiple jobs issuing a single *globusrun* command. For example, it is not possible to submit multiple instances of the same executable (using the `count=x` RSL parameter) specifying different input and output files for the different runs (as it is possible for example in the Condor system appending `$(Process)` on the input/output file name in the submit file). This problem (quite common for example in the physics community, where the same processing must be applied to different data) can be managed only issuing multiple *globusrun* commands, but, as described in Section 5.2, this means that multiple job managers will run in the machine where the Globus gatekeeper has been configured. This could be a problem for the scalability of the system; let's suppose, for example, that hundreds or thousands jobs are submitted to a farm with a host configured as front-end machine: this means hundreds (or thousands) of job managers need to run on this machine. It is possible to submit, with a single *globusrun* command, multiple instances of the same job, only if this application doesn't have any input and output, or if there is just one input file for all the different runs: a quite unrealistic scenario.

We also found that the Globus job manager, responsible to "manage" the created process, as described in section 5.2, is not persistent. This problem can seriously affect the reliability of a Globus-based resource management system. Let's suppose, for example, that a client submits a job to a remote farm. The job will run on one node of this farm, while the corresponding job manager will run on the machine where the Globus

gatekeeper has been configured. If the job manager crashes, for example for a crash of this front-end machine, the job in execution will be left "orphaned", without a job manager taking care of it, and it will not be possible to manage the job from the client side anymore: the client tries to check the status of the job, the remote job manager cannot be contacted because it is not running, so it is assumed that the job has been terminated.

Several other problems were found during our tests [30][31][32][33], showing that the current implementation lacks the reliability that's needed in real production environments. For example:

- Many memory leaks in the job manager, triggered by the job submission via Condor-G (see section 5.8.1);
- Using the *globusrun* command, it is not possible to use the option `-s` (used, for example, to copy the output of the job back to the client machine, via a Gass server activated in the client) for batch jobs;
- A problem related with the monitoring of jobs submitted to LSF: the *globus-job-status* command reported as active even the pending jobs;
- A problem with the standard input/output/error files for vanilla Condor jobs, by default redirected to `/dev/null`;
- For LSF the `count=x` parameter (that should specify that it is necessary to submit `x` instances of the job) is translated to the `-n x` switch of the `bsub` command (used to specify the number of processors required for a parallel job: the command just allocates `x` processors of the LSF cluster, but the job is dispatched just to the first one). The problem could be solved using the LSF job arrays, but, as described above, it is quite unrealistic to submit multiple instances of the same application issuing a single *globusrun* command.

Most of the problems found during our evaluations have been solved, and the correspondent patches have been included in the INFN-GRID distribution. They have also been reported to the Globus team. Hopefully the patches will be included in the next releases, but we haven't been able to see a new release during our evaluation period.

5.4 Resource Specification Language

As illustrated in Figure 5.1, in the Globus resource management architecture the Resource Specification Language (RSL) [28][36] is used for exchanging information about resource requirements between the various components of this architecture.

We think that the RSL syntax model, where the core element is the relation (an <attribute, value> pair), seems suitable to define even complicated resource specification expressions. In the RSL, a common set of *scheduler parameters*, used to communicate information regarding the job (such the name of the executable, the command line arguments for the executable and the environment variables that must be considered), has been defined. These scheduler parameters are interpreted directly by the resource manager. Our tests [30] showed that the Globus job manager only cares about this common set of RSL attributes. Any attributes outside of the set are not passed unchanged to the underlying resource management system (as it might appear reading the Globus documentation): they are simply ignored. In some cases this can be a serious problem, if it is necessary to express a specification that can't be built using just this set of parameters. For example, let's consider a vanilla Condor job (that requires a uniform file system and UID domain) that must be submitted to an heterogeneous Condor pool, composed by machines spread across different administrative domains: it is not possible to force the job to be executed on a uniform subset (machines sharing the same file system and UID domain) of this Condor pool (that is using the *Requirement* ClassAd). The only possible workaround is to modify the script that Globus uses to submit jobs to the underlying resource management system, to allow users to specify other attributes, through the *environment* RSL parameter. For other resource management system (such as LSF and PBS), where the queue concept is used, this limitation might be not so serious, since usually all the policies are defined in the queues, and therefore a queue represent a uniform set of resources: users typically have just to specify the queue where to submit their jobs (and the *queue* parameter is included in the common set of attributes) without additional specific requirements.

The language used to specify the resources is very tied to the broker component of the workload management system. In order to have a flexible matchmaking system, it is necessary to have a flexible, uniform, extensible language. For example, the administrator of a resource should be able to define new attributes describing particular (static or dynamic) characteristics of this resource, and users should be allowed to use these attributes in their resource specification expressions.

5.5 GRAM Reporter

As described in Section 5.2, the GRAM reporter is the entity responsible for providing the GIS (MDS) with information on characteristics and status of the local resources and of the jobs. Shell scripts that invoke the local scheduler tools are typically used to get this information. We analyzed this cooperation between the GRAM and the GIS services, in particular when the GRAM service is used as an interface to a farm managed by a

dn	service=jobmanager-lsf, hn=lxde02.pd.infn.it, dc=pd, dc=infn, dc=it, o=Grid
objectclass	GlobusTop
objectclass	GlobusDaemon
objectclass	GlobusService
objectclass	GlobusServiceJobManager
objectname	service=jobmanager-lsf, hn=lxde02.pd.infn.it, dc=pd, dc=infn, dc=it, o=Grid
service	jobmanager-lsf
hn	lxde02.pd.infn.it
contact	lxde02.pd.infn.it:2119/jobmanager-lsf:/C=IT/O=INFN/OU=gatekeeper/L=PD/CN=lxde02.pd.infn.it/Email=root@lxde02.p
schedulerversion	0.1
schedulertype	lsf
deploydir	/etc/globus
cputype	i686
osversion	2.2.12-20
osname	linux
manufacturer	unknown
machine_type	unknown
gramversion	1.67
gramversiondate	2000/04/17 10:00:45
gramsecurity	ssleay
lastupdate	Thu Nov 09 15:35:49 GMT 2000
ttl	00:30:00

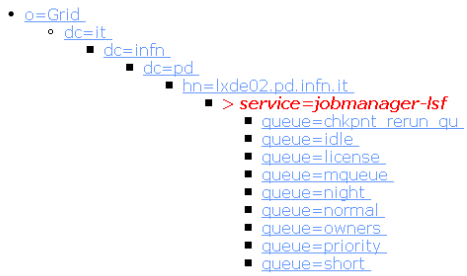


Figure 5.6: Example of a `GlobusServiceJobManager` objectclass

local resource management system. For this evaluation we have already considered farms managed by LSF and Condor [30] [37], while the work considering PBS is in progress.

When a local resource management system is configured to work in conjunction with the GRAM service, by default some information is published in the GIS, according to a specific schema: a `GlobusServiceJobManager` objectclass is considered, as illustrated in the example of Figure 5.6, and under this objectclass, some `GlobusQueue` objectclasses are defined, each one representing a queue[1], as illustrated in the example of Figure 5.7 (the two examples refer to LSF, but there are no significant differences if other resource management systems are considered).

Besides this information related to the resources managed by the resource management system, it is possible to publish in the GIS information concerning the jobs submitted via Globus. These jobs are represented by `GlobusQueueEntry` objectclasses, as in the

[1]For Condor a single `GlobusQueue` objectclass is created, representing all the machines of the pool with the same architecture and operating system of the machine where the Globus GRAM service has been configured

dn	*queue=mqueue, service=jobmanager-lsf, hn=lxde15.pd.infn.it, dc=pd, dc=infn, dc=it, o=Grid*
objectclass	GlobusTop
objectclass	GlobusQueue
queue	mqueue
maxtime	0
maxcputime	0
maxcount	0
maxrunningjobs	3
maxjobsinqueue	0
maxtotalmemory	0
maxsinglememory	0
totalnodes	5
freenodes	1
whenactive	0
status	Open:Active
dispatchtype	batch
priority	30
alloweduserlist	NULL
jobwait	NULL
schedulerspecific	NULL
ttl	00:01:00
lastupdate	Thu Aug 31 10:14:22 GMT 2000

Figure 5.7: Example of a `GlobusQueue` objectclass

dn	*jobid=2481, queue=intel-linux, service=jobmanager-condor, hn=lxde02.pd.infn.it, dc=pd, dc=infn, dc=it, o=Grid*
jobid	2481
objectclass	GlobusTop
objectclass	GlobusQueueEntry
localjobid	2481
specification	&("executable" = "/home/sgaravat/ciao")("stdout" = "/home/sgaravat/ox-cond.out")("stderr" = "/home/sgaravat/ox-cond.err")("count" = "1")("arguments" = "one" "two")
globaljobid	https://lxde02.pd.infn.it:1475/616/973770651/
count	1
status	ACTIVE
start time	0
schedspecific	NULL
ttl	00:01:00
lastupdate	Thu Nov 09 11:51:07 GMT 2000

Figure 5.8: Example of a `GlobusQueueEntry` objectclass

example of Figure 5.8.

We found that many of the attributes published by default in the Grid Information Service are useless (at least for our needs), also because most of them are just defined in the schema, but they are not calculated (they are always defined as 0). Moreover, the Globus shell scripts do not properly calculate some attributes: it is the case, for example, of the attributes *totalnodes* and *freenodes* for LSF (that should represent the number of total and available nodes associated to a specific queue): new scripts able to find the correct values have already been implemented, and are now being tested. Some important information describing the farms and the submitted jobs, necessary for example for a broker that must choose the "best" resources where to submit jobs, is missing in this default schema. We therefore proposed a first draft for a possible modification of the default schema, dropping the attributes that in our opinion are not useful at all, and considering

Attribute	Description
hn	The host name of the “front-end” machine
service	The name of the job manager
contact	The string identifying the contact information
ResourceManagementType	The type of resource management system
ResourceManagementVersion	The version of the resource management system
Gramversion	The GRAM version

Figure 5.9: Proposed GIS attributes for a resource management system

some other information provided by the commands and tools of the underlying resource management system [37]. This first proposal, that is now under discussion, is represented in Tables 5.9, 5.10 and 5.11: the first table represents a local resource management system, the second one a queue (assuming that a queue represents a set of homogeneous resources: when a job is submitted to a specific queue, it doesn't matter in which host associated to this queue the job is dispatched). The attributes in Table 5.11 represent a job.

As a first step we just considered the information that can be provided by tools and commands of the underlying resource management system: other required information could be provided by specific information providers.

5.6 GRAM Client API Evaluation

The GRAM Client API includes eleven functions. Before any of these functions is called, a software module, which is specific of the GRAM client, has to be loaded. Module activation automatically triggers the activation of possibly other modules, which the first one relies on. In the case of the GRAM client also the POLL, IO, GRAM_HTTP modules are loaded together with all the other modules that these need. In particular during the GRAM_HTTP activation the client credentials are acquired, allowing to run in a secure environment.

The full graph of dependencies for the GRAM client module is shown in Figure 5.12.

Once the GRAM client module is not needed any more, it can be deactivated.

A job, in the form of an RSL description, can be submitted through a call to the globus_gram_client_job_submit function. The function returns as output a unique job handle, which can then be used for several other functions, in particular to monitor the status of the job (through the globus_gram_client_job_status function), or to kill it

Attribute	Description
Queue	The name of the queue
Architecture	The architecture of the machines associated to this queue
OpSys	The operating system of the machines associated to this queue
TotalCpus	The number of total CPUs associated to this queue
FreeCpus	The number of free processors available to run jobs submitted to this queue
TotalJobs	The total number of jobs in the queue
RunningJobs	The number of jobs currently running in the queue
IdleJobs	The number of jobs not running in the queue
MaxTotalJobs	The maximum number of jobs allowed for this queue
MaxRunningJobs	The maximum number of running jobs allowed for this queue
Status	The status of the queue
RunWindows	The time windows that define when the queue is active
Priority	The priority of the queue
MaxCpuTime	The maximum CPU time allowed for jobs submitted to this queue
MaxWallTime	The maximum wall clock time allowed for jobs submitted to this queue

Figure 5.10: Proposed GIS attributes for a queue

Attribute	Description
GlobalJobId	The Globus id of the job
LocalJobId	The id of the job in the underlying resource management system
GlobalUser	The id of the Grid user
LocalUser	The username in the local system
Status	The status of the job
PendingReason	The reason for which the job is not running
RSLCommand	The RSL string used to submit the job
SubmitTime	The time at which the job has been submitted
StartTime	The time at which the job first began running
WallClockTime	The wall clock time accumulated for this job

Figure 5.11: Proposed GIS attributes for a job

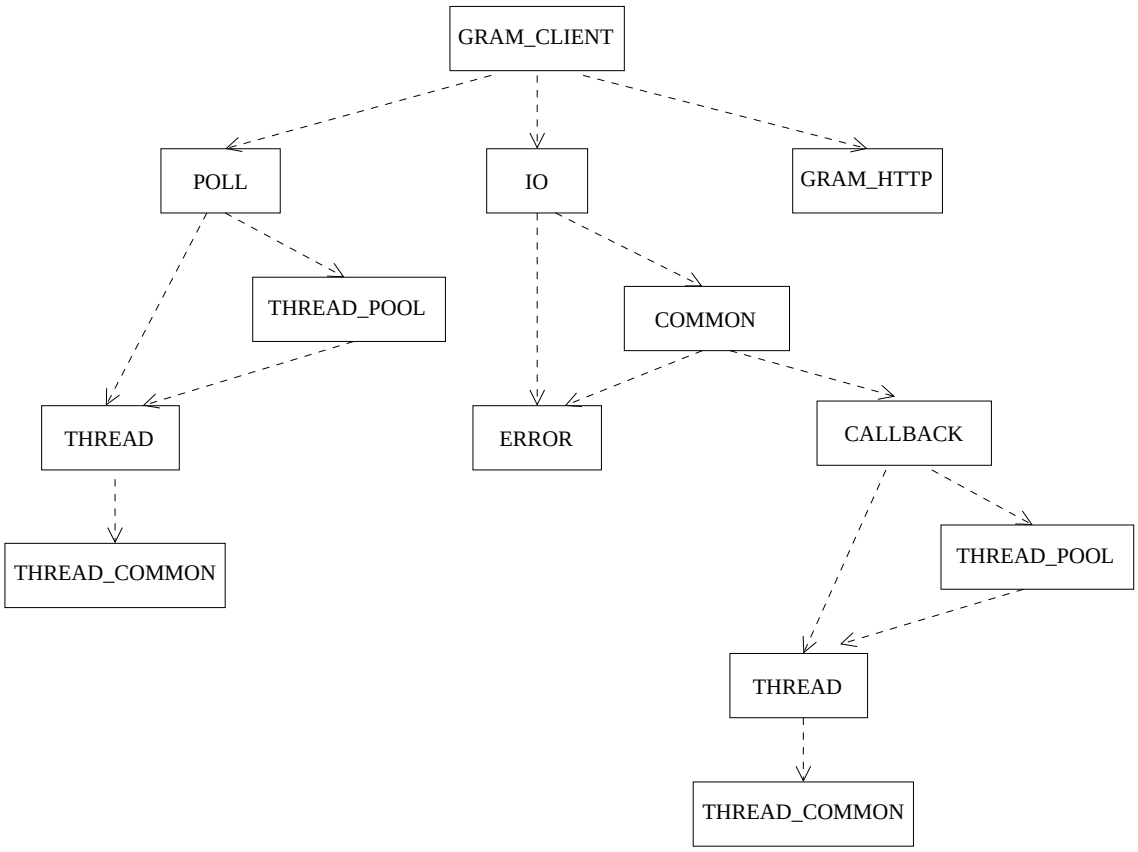


Figure 5.12: Graph of dependencies for the GRAM client module

(through the `globus_gram_client_job_cancel` function).

In addition, the callback mechanism provided by the GRAM client API can be used to allow the job submitter to be asynchronously notified of a job state change.

Two functions of the API (one for jobs already submitted, and another one for not-yet-submitted jobs) allow obtaining an estimate of the time a certain job would start running. Unfortunately these two functions are not implemented yet. They would be extremely useful in the implementation of a grid scheduler, because the scheduler, if needed, could delegate the estimation of a job start time to the resource, which knows its current state better than what would be possible considering the information published in an Information Service.

From the preliminary tests done so far the GRAM client API seems quite complete and correctly implemented. Also the documentation, which happens to be quite poor for other Globus modules, is accurate enough.

5.7 GRAM Performance

We tried to evaluate the overhead introduced by the Globus GRAM service [38]. For this purpose, tests have been done considering a specific, homemade benchmark application, called FBP (Flexible Benchmark Program). In particular, FBP has been used submitting jobs of different durations to a Globus resource, and measuring the wall clock time in the submitting machine and in the executing machine: the difference between these two values represents the overhead introduced by Globus. We found that the overhead is practically constant (about 6 seconds), and doesn't depend on the wall clock time of the job, as expected.

5.8 Submitting Condor jobs to Globus resources

Besides considering the submission of Globus jobs to a Condor pool, as reported in section 5.3, we also tried the submission of Condor jobs to Globus resources [30][31]. There are two ways to run programs on Globus resources, using Condor. The first method is the so-called *Condor-G*, while the second one involves the *GlideIn* mechanisms.

5.8.1 Condor-G

Condor-G allows submitting jobs to Globus resources, profiting from some capabilities, features and mechanisms of the Condor system. In particular, since in Condor the queue of the submitted jobs is saved in a persistent way, using Condor-G it could be possible to implement a reliable, crash-proof, checkpointable job submission service. Also, the

Condor tools for job management (job submission, job removal, job status monitoring) and logging can be exploited.

The current implementation of Condor-G, which relies upon the Condor *schedd* service, runs the Globus *globusrun* command behind the scenes: a *condor_submit* command, used to submit a job to a Globus resource via Condor-G, is simply translated to the submission library equivalent of a *globusrun* command. The Condor-G mechanisms have been tested submitting vanilla jobs to Globus resources represented by single hosts using the fork system call as job manager, and by farms using LSF and Condor as underlying resource management systems. As reported in section 8.3, Condor-G has been also tested considering a Monte Carlo production for the CMS experiment, and it was during these tests that many memory leaks in the Globus job manager were found: we have been able to provide patches for these bugs, that have been included in the INFN-GRID distribution, and that should be included in the next official Globus releases. Apart from this serious problem, we found that the submission of Condor-G jobs, the job status monitoring, the job removal seem to work fine, but the current implementation of Condor-G is just a prototype, and there are still various problems that must be fixed, for example:

- For each Condor-G job submitted to a remote Globus resource, a shadow process runs in the submitting machine; this can be a problem for the scalability of the system;
- It is not possible to submit multiple instances of the same job, specifying different input and output files for the different runs, because of the problems with the standard GRAM submission (see Section 5.3);
- The logging information is not correct, since the name of the submitting machine is reported as executing machine;
- It is very difficult to debug problems; for example if there are some errors in the submit file, the *condor_q* command reports that the job is in execution in the remote resource, while it is not actually running;
- The arguments *input*, *output*, *error* can't be specified in the submit file (the workaround is using the `GlobusRSL` attribute);
- With Condor-G it is not possible to automatically stage the input and/or executable file from the submitting machine to the remote Globus resource, and/or copy the output file back in the submitting machine: this is because, as reported in Section 5.3, the *globusrun* command cannot activate a Gass server for jobs submitted in background.

It must be stressed that Condor-G doesn't provide any brokering/matchmaking functionalities (this means that the Globus resource where the job must run has to be explicitly specified), and there's no plans to provide a way to plug in application specific resource choice policies in Condor-G: the place to implement this resource choice is within a component that sits on top of Condor-G itself.

5.8.2 Condor GlideIn

With GlideIn, what happens is that the Condor daemons (specifically, the *master* and the *startd*) are effectively run on Globus resources (machines that use the fork system call as job manager, or clusters managed by a resource management system). These resources then temporarily (the Condor daemons exit gracefully when no jobs run for a configurable period of time) become part of a given Condor pool, which can then be used to submit any kind (standard or vanilla) of Condor jobs. GlideIn is a particular implementation of the Condor-G mechanism: the Condor master is submitted as a Condor-G job.

The GlideIn procedure operates in two steps, after acquiring a valid user proxy. In the first step, that must be considered only once, the Condor executables and configuration files are downloaded from a server in Wisconsin, while in the second phase the Condor daemons are executed in the remote Globus resource.

Only some preliminary tests, reported in [30][31], have been done using GlideIn, considering as Globus resources single hosts using the fork system call as job manager, and farms managed by LSF and Condor. It is not clear at the moment if and how these mechanisms could be useful for our needs.

5.9 MPICH-G2 performance evaluation on PC clusters

The Message Passing Interface (MPI) [39] is a standard specification for message passing libraries. Among the several implementations of MPI, the most popular ones are LAM and MPICH [40], both available for Linux PC clusters. The MPICH implementation was developed and distributed by the Argonne National Laboratory (ANL) MPICH group. The communication functionality of MPICH is based on a communication device having a common *Abstract Device Interface* (ADI); `ch_p4` is the default device when MPICH is compiled on Linux systems. It supports shared memory through the Unix System V Interprocess Communication (IPC). MPICH-G2 [41], developed at ANL, is the implementation of MPI integrated with the Globus services (e.g., job startup, authentication, security, data conversion, file access, etc.). It uses a new device named `globus2`. Existing parallel programs written for MPICH can be executed over the Globus infrastructure with just a recompilation.

IP name	Host description
janus.pr.infn.it	Dual PII350, 256 MB, Fast Eth 3C905A, Red Hat 6.2, INFN-GRID 1.1.2
janus1.pr.infn.it	Dual PII350, 256 MB, Fast Eth 3C905A, Red Hat 6.2, INFN-GRID 1.1.2
janus2.pr.infn.it	Dual PII350, 256 MB, Fast Eth 3C905A, Red Hat 6.2, INFN-GRID 1.1.2
lxde02.pd.infn.it	PIII450, 256 MB, Red Hat 6.1, INFN-GRID 1.1.2

Figure 5.13: Configuration of the hosts used for the MPI tests

Packages	Compile options
mpich-1.2.1-2.i386.rpm	-with-device=ch_p4
mpich-smp-1.2.1-2.i386.rpm	-with-device=ch_p4 -comm=shared
mpich-G2-1.2.1-2.i386.smp	-with-device=globus2
mpich-G2-smp-1.2.1-2.i386.rpm	-with-device=globus2 -comm=shared

Figure 5.14: RPM packages available from ftp.pr.infn.it.

We performed some tests about the functionalities of MPICH-G2 on a PC cluster with respect to the standard MPICH/ch_p4. Our main goal was to make a performance comparison using different communication mechanisms such as SMP, LAN and WAN, and to verify the interoperability of MPICH-G2 and Globus.

The test cluster, described in Table 5.13, has three local nodes and one node installed in a remote site. The local nodes are interconnected through a 3Com Super Stack Fast Ethernet switch; the remote node is reachable using the WAN with a bandwidth of 2 Mbit/s. Each node is running INFN-GRID 1.1.2, which is the Globus 1.1.4 distribution customized by INFN.

We prepared four binary rpm distributions compiled using the ch_p4 device or the globus2 device with or without the shared memory support option enabled. The RPM packages (listed in Table 5.14) are available on our ftp site [42] and are installed on the MPI submitting machine janus.pr.infn.it. In the rest of this document we will call "MPICH" the distribution compiled with device ch_p4 and "MPICH-G2" the distribution compiled with device globus2.

We measured throughput and latency of each package using the standard tools included in the mpich distribution (example/perftest) [43]:

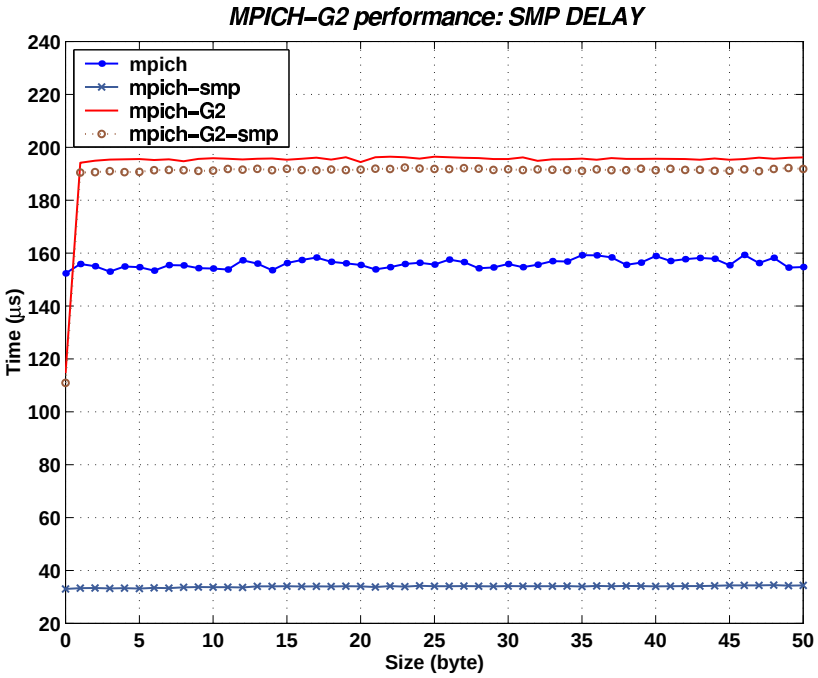


Figure 5.15: MPICH-G2 performance: SMP DELAY

- **mpptest** performs point to point communications, that is basically the classic ping-pong test of messages with different size, repeated several times. (For example `mpptest -size 0 50 1 -reps 4` means repeating 4 times a sequence of roundtrip messages from 0 up to 50 bytes with increment of 1 byte).

- **goptest** performs collective communications such as broadcast (a message from one process is broadcasted to all other processes) and reduction (a function such as sum, max, logical and, etc., is performed on a variable across all the processes). It is possible to specify the number of processes, the size of the variable and the number of repeats. (For example `goptest -np 4 -bcast -sizelist 10,20 -reps 4` means repeating 4 times a broadcast between 4 processes with 2 messages of 10 and 20 bytes).

We also used, as a second functionality test, a custom benchmark named **Rete_MPI** [44]. The program reports the time needed to perform a fixed number of learning epochs of a neural network where the learning patterns are distributed across the processes.

We executed the point-to-point tests using the following commands:

`mpirun -np 2 mpptest -reps 4 -size 0 50000 1000` (to get bandwidth)

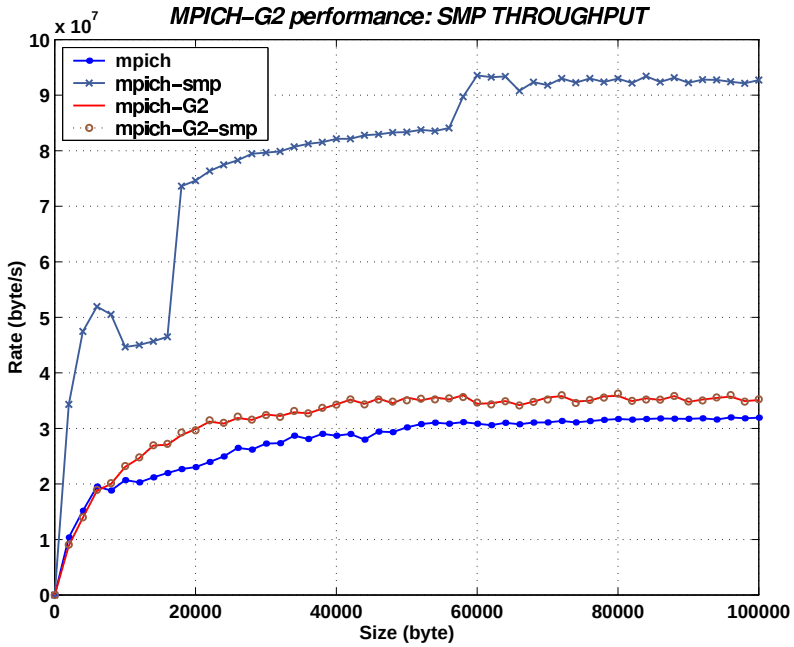


Figure 5.16: MPICH-G2 performance: SMP THROUGHPUT

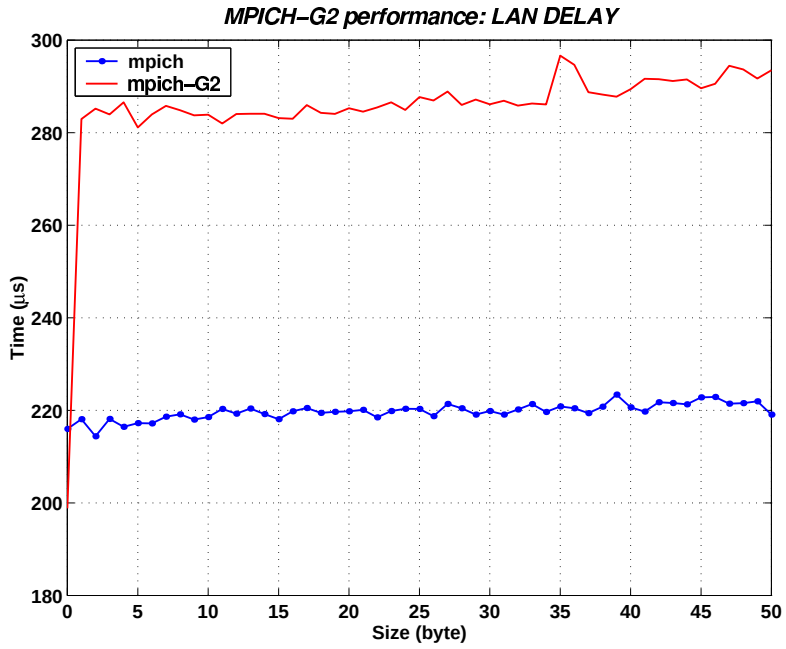


Figure 5.17: MPICH-G2 performance: LAN DELAY

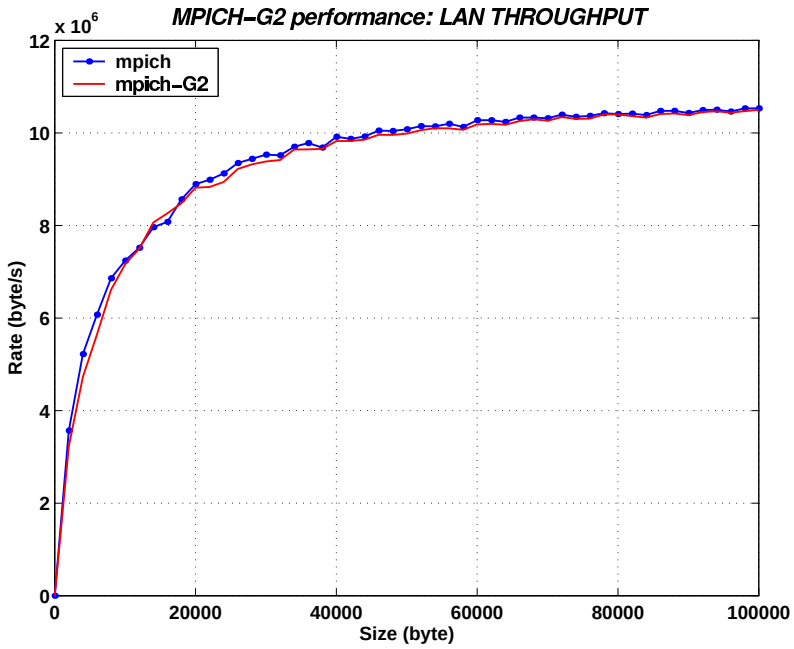


Figure 5.18: MPICH-G2 performance: LAN THROUGHPUT

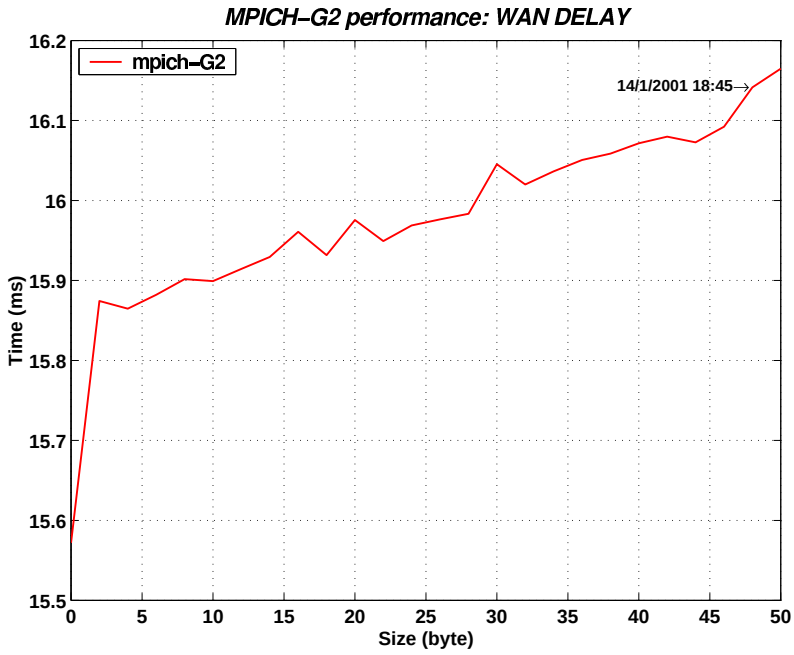


Figure 5.19: MPICH-G2 performance: WAN DELAY

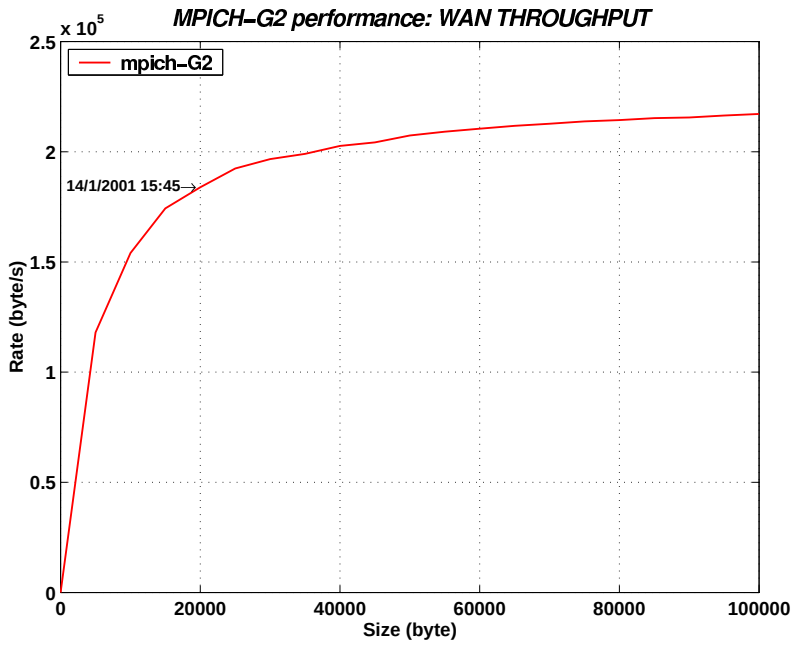


Figure 5.20: MPICH-G2 performance: WAN THROUGHPUT

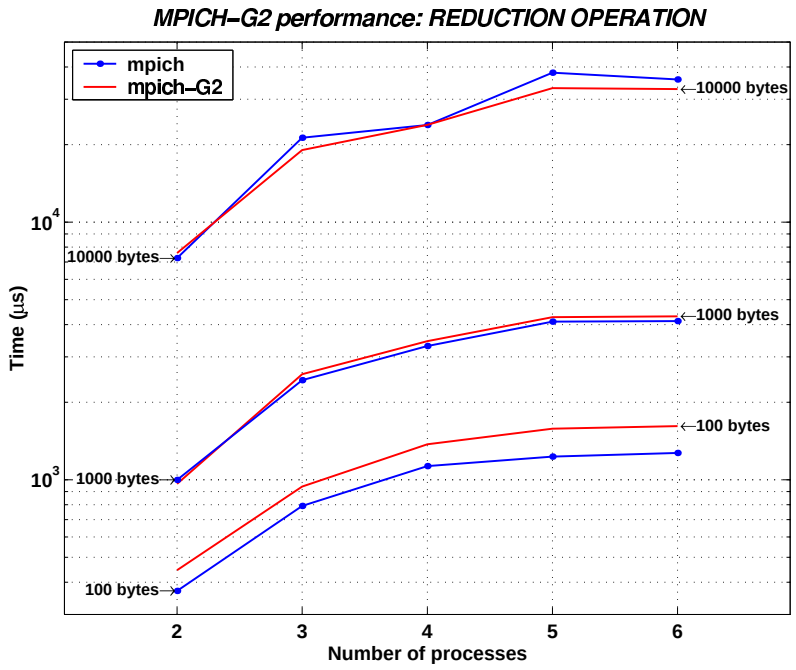


Figure 5.21: MPICH-G2 performance: REDUCTION OPERATION

`mpirun -np 2 mpptest -reps 4 -size 0 50 1` (to get latency)

The four SMP tests were executed on a single biprocessor machine (see Figures 5.15 and 5.16). Support of shared memory in `globus2` is not documented and the tests confirm that shared memory is not yet supported. For the `ch_p4` device the tests confirm that shared memory is supported but there is an unexpected performance 'hole' for message sizes in the 7 to 17 Kbytes range.

LAN tests were performed between two different machines on the same Fast Ethernet LAN without shared memory support. The results (see Figures 5.17 and 5.18) show a higher latency of MPICH-G2 with respect to MPICH.

Global collective communications have been tested locally using MPI reduction operation from 2 up to 6 processors using the command:

`mpirun -np <2-6> goptest -dsum -reps 15 -sizelist 100,1000,10000`

We wanted to compare the behaviour of MPICH and MPICH-G2 with different number of processes and size of messages. The relative figure (Figure 5.21) shows a better performance of MPICH with short messages (100 bytes). MPICH-G2 overcomes MPICH with bigger messages (see 10000 bytes).

In order to evaluate process distribution performance on WAN we generated the proper RSL file for remote execution of mpptest (`mympptest.rsl`) and Rete_MPI (`retempi.rsl`). We started their execution using the commands `mpirun -globusrsl mympptest.rsl` and `mpirun -globusrsl retempi.rsl`.

As an example the list of the rsl file `mympptest.rls` is:

```
+
( &(resourceManagerContact="janus.pr.infn.it")
 (count= 1)
 (label="subjob 0")
 (environment=(GLOBUS_DUROC_SUBJOB_INDEX 0))
 (arguments= "-reps" "10" "-size" "0" "50" "2" )
 (directory="/home/alfieri")
 (executable="/home/alfieri/mpptest")
)
( &(resourceManagerContact="lxde02.pd.infn.it")
 (count= 1)
 (label="subjob 1")
 (environment=(GLOBUS_DUROC_SUBJOB_INDEX 1))
 (arguments= "-reps" "10" "-size" "0" "50" "2")
 (directory="/home/alfieri")
```

```
 (executable="/home/alfieri/mpptest")
)
```

Our WAN tests (see Figures 5.19 and 5.20) included remote execution and submission using the Globus interface. We verified that the remote execution of the command:

`globusrun -r janus.pr.infn.it -f mympptest.rsl`

works from any Globus authenticated account.

To verify remote submission we installed the PBS job scheduler on our MPI submitting machine `janus.pr.infn.it` and we created on janus two PBS script files: `mpich-job` and `mpich-G2-job`. The first one executes the mpptest compiled with MPICH while the second one executes the mpptest compiled with MPICH-G2.

We verified that the command:

`globus-job-submit janus.pr.infn.it/jobmanager-pbs /home/alfieri/mpich-job`

works, while the command:

`globus-job-submit janus.pr.infn.it/jobmanager-pbs/home/alfieri/mpich-G2-job`

fails with the following error message:

```
GSS authentication failure
 GSS status: major:000a0000 minor: 00000000 token: 00000000
 GSS_S_DEFECTIVE_CREDENTIAL - sslv3 handshake
 Function:gss_accept_sec_context
 Reason:Peer is using (limited) proxy
Failure: GSS failed Major:000a0000 Minor:00000000 Token:00000000
GSS_S_DEFECTIVE_CREDENTIAL
 Consistency checks
 performed on the
 credential failed.
```

A ”limited proxy” is a feature of Globus authentication model to enforce security level that, in special situations, unproperly reject the authentication. This problem is well known inside the Globus team and, hopefully, it will be corrected in next Globus release.

Point to point latency and bandwidth results are summarized in the following table:

	Mpich bandwidth	**Mpich-G2 bandwidth**	**Mpich latency**	**Mpich-G2 latency**
SMP	95 MB/s †	37 MB/s	35 μs	190 μs
LAN (100 Mb/s)	11 MB/s	11 MB/s	215 μs	280 μs
WAN (2 Mb/s		220 KB/s		16 ms

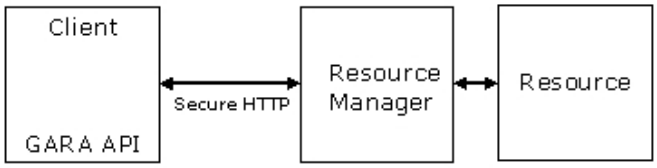


Figure 5.22: GARA basic architecture

†)Shared memory option enabled

These results confirm the absence of shared memory support in MPICH-G2 and a worse latency performance with respect to MPICH/ch_p4. MPICH-G2 seems stable and its performance with respect to MPICH/ch_p4 increase with message size and number of processors.

5.10 GARA

In this section we provide a simple overview on GARA based on [45] and we express some comments on the network reservation part of GARA.

5.10.1 Overview

The goal of the General-purpose Architecture for Reservation and Allocation (GARA) is to provide applications with a method to reserve resources like disk space, CPU cycles and network bandwidth for end-to-end Quality of Service (QoS). GARA provides a single interface for reservation of diverse resources.

GARA has a hierarchical structure ([45], see Figure 5.22). At the lower layer the resource manager performs resource admission control (to make sure that only entitled customers actually get access to the grid resources) and reservation enforcement. Communication with the resource manager is through the Globus Gatekeeper, which authenticates and authorizes the resource requester. At layer 2 the Local Reservation implements an API for reservation request in a single trust domain. Reservation authentication through GSI is supported at layer 3, so that reservations can be requested remotely. The higher level supports mechanisms for end-to-end reservation.

Reservations can be made in advance or immediately when needed by the application itself. Transmission Quality of Service is implemented according to the Differentiated Services architecture, which provides traffic aggregates with differentiation through marking, policing, scheduling and shaping. Packets are marked at the ingress point of

the network with a code called the Differentiated Services CodePoint (DSCP). Packets generated by different application sessions can share the same codepoint. Then, in each congestion point packets are placed in different dedicated queues, so that depending on the priority, they will experience different treatment. Quality of Service can be quantified through several performance metrics like: one-way delay, one-way delay variation, packet loss probability, throughput, etc.

CPU reservation is implemented through a mechanism for process scheduling called Dynamic Soft Real-Time (DSRT), while disk space reservation is based on DPSS.

5.10.2 Network Reservations

Quality of Service configuration in GARA requires three fundamental building blocks: marking, policing and scheduling. Each time a new reservation request is received, the edge router configuration has to be modified. The prototype is designed to work with CISCO routers only and it uses the Cisco Command Line Interface. Scheduling preconfiguration at the egress interface of the router is required. The mechanism requires configuration privileges on the router to proceed with router configuration every time a reservation request is received.

5.10.3 Comments on Network Reservations

We believe that the approach to network reservation adopted by GARA is of great interest, since it addresses the problem of end-to-end Quality of Service, a fundamental requirement for networked applications. However, we think that some aspects may need investigation.

The change in router configuration every time a new reservation is received is a viable solution only if the number of reservations performed locally is not frequent. The alternative approach would be to adopt static configuration, which is possible when the source/destination IP address of GRID hosts or the corresponding subnets is known in advance.

A second issue is related to per-flow policing. The number of policing/marking instances, which have to be enabled on the input interface of the router, is a critical parameter. Performance of small edge routers is greatly dependent on the number of traffic filters (access-lists) enabled at one time for traffic policing. Per-microflow policing offers better traffic isolation at the expense of additional CPU overhead.

A third potential weakness of the architecture depends on the fact that resource reservation does not automatically recognize the ingress interface to which a policer/marker has to be associated, i.e. it does not rely on routing information, but rather requires that for

each host allowed to reserve bandwidth, the corresponding input interface on the router is known. This is specified in a configuration file which has to be manually updated every time the set of local hosts varies. This approach is human-error prone.

Chapter 6

Data Access and Migration

Data management is a crucial issue for both DataGrid and INFN-GRID projects. Its activities could be grouped in two major tasks. The first task is related to the distributed computing environment. In order to run a job, the end user should be able to access remote data and/or to transfer the input data to the target machine and/or copy back the resulting output data sets. The second task is related to a hierarchical distributed data repository system. People in charge for the data replication from the places where they are collected to every site where the experimental collaborations need them should rely on efficient, reliable and fault tolerant mechanisms to perform the task. A more complete description of the data management issues can be found in the proposals of the DataGrid project [3] and the INFN-GRID project [6].

The functional blocks and components of the Globus data-intensive services architecture are described in Figure 6.1.

This figure shows the modularity of the architecture and the role of the GASS service in the I/O management. The service is made of library functions that can be used by other Globus modules and tools. Both libraries and tools can be used by the user programs ("custom servers" and "custom clients" in the picture). In the work plan of the Globus evaluation toolkit Work Package we proposed to test the capabilities, performance and ease of use of the GASS service and the GSIFTP software. While GASS is part of the official Globus toolkit, GSIFTP was essentially standard FTP enhanced to use GSI security mechanisms. A short description of the GASS service and of the functionality tests we have done is reported in Section 6.1. For what concerns GSIFTP, while we were performing simple functionality tests, the Globus Team produced an alpha release of the so-called data grid tools. These include GlobusFTP [47], a wide implementation of the GridFTP protocol [46], and tools for managing replica catalogs. We have been offered by the Globus team to test these tools, so we decided to concentrate our efforts on testing it, despite it was an alpha release and consequently not included in the official Globus

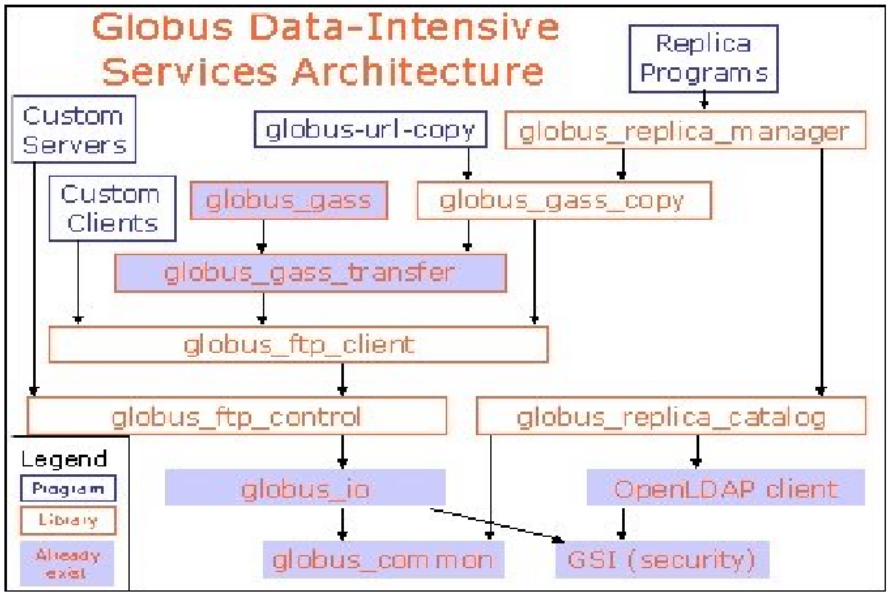


Figure 6.1: Functional blocks and components of the Globus data-intensive services

toolkit. A description of the GlobusFTP software and of the tests we have done is reported in Section 6.2.

6.1 GASS

The Global Access to Secondary Storage (GASS) service [48] is the Globus tool that simplifies the porting and running of applications that use file I/O to the Globus environment. It consists of libraries and utilities that eliminate the need for manually transferring files from/to remote sites and/or share disks by means of distributed file systems. In theory it's quite easy to use the GASS library functions to access remote files: the C `open()` and `close()` functions should simply be substituted by `globus_gass_open()` and `globus_gass_close()` respectively. In practice, however, it could be really difficult. As a matter of fact the code developed for our purposes (High Energy and Nuclear Physics experiments) results from the collaborating work of a huge number of researchers spread over many countries. Since, as for now, the Globus software is not a production-quality code and a Grid infrastructure is not widely implemented, it's very unlikely that the HE(N)P experiments change their production code to take advantage of the library functions described above. In addiction, some HE(N)P experiments make use of commodity software (Objectivity is one of the most used packages) whose source code is not

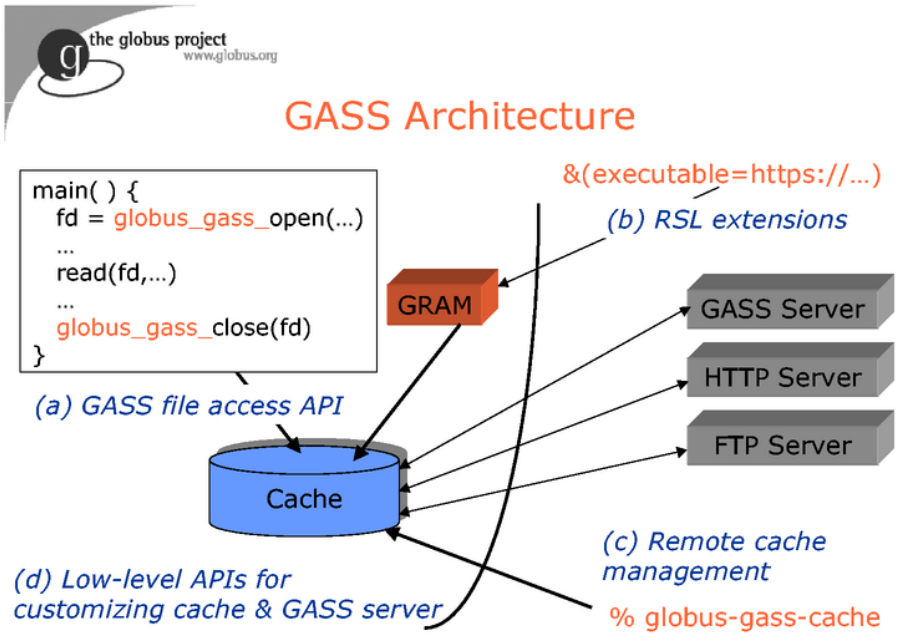


Figure 6.2: GASS Architecture

accessible to the final users. In the current Globus distribution, GASS can rely on some protocols (and the correspondent servers) to transfer data: *x-gass* (whose server is the GASS server) -a protocol specifically developed by the Globus Team-, the standard FTP and the HTTP protocols. They are all accessible by means of a unique URL syntax. The file management is done by means of a cache on disk where the files are staged for the entire duration of the computing phase. The GASS architecture is shown in Figure 6.2.

The GRAM uses the GASS service to copy the input files and the executable from the submitting machine to a disk cache in the executing machine by means of one of the 3 servers shown in the picture. When the job is completed the output and error files are copied back to the submitting machine using the same mechanism. Some APIs are also available to perform the management of the remote cache.

Some tests about GASS have been performed in Pisa [49] using *globus-url-copy*, *globus-rcp* and the GASS API of the INFN-GRID toolkit [8] release 1.1, based on Globus release 1.1.2. The goal was to test the performance of the file access API between two Linux machines: `pcmsfarm04` and `pcmsfarm05`. The test was done when these two machines were idle. The user's `gass_cache` area was a link to the scratch area on another Linux machine (`pcmsfarm01`) which was mounted via NFS. So the result of the test include the LAN performance. Before doing the test a gass server was started on the remote

File size(MB)	File Open (sec)	File Write (sec)	I/O (sec)	File Close (sec)
1	8.8	0.1	0.1	5.5
10	9.3	1.1	1.1	5.8
50	10.2	5	5.3	32
100	40	10	10	64
200	80	20	20	140
400	160	40	40	300
600	180	63	61	435
800	310	87	89	580
File size(MB)	File Open (sec)	File Read (sec)	I/O (sec)	File Close (sec)
1	8.1	0.006	0.037	0.03
10	10.2	0.11	0.33	0.06
50	40	0.8	1.98	0.06
100	80	1.5	2.78	0.12
200	156	3.1	10.4	0.18

Figure 6.3: The result of test on file access API of GASS.

machine using the command *globus_gass_server*. This command isn't available anymore in Globus release 1.1.3. Table 6.3 shows the time and speed of reading/writing a file from (to) a remote machine. The transferred file had a block size of 1024 bytes.

When a remote file is opened by `globus_gass_open()` in write only mode, the following actions are taken:

1. The file is looked up in the local cache.
2. If it does not exist, the file is copied from the remote server into local cache.
3. A tag is added to the cache entry's tag list.
4. The local file is opened for write only access.

When the file is closed:

1. The file is copied from the local cache to the remote server.
2. The tag is removed from the cache entry's tag list.
3. The cache entry is deleted if the tag list is empty.

The I/O time for the file write test is the time to copy the file to the local cache. The file close time is the time to copy the file from the local cache to the remote server and

remove the tag from the cache. When a remote file is opened by `globus_gass_open()` in read only mode, the following actions are taken:

1. The file is looked up in the local cache.
2. If it does not exist, the file is copied from the remote server into local cache.
3. A tag is added to the cache entry's tag list.
4. The local file is opened for read only access.

When the file is closed:

1. The tag is removed from the cache entry's tag list
2. The cache entry is deleted if the tag list is empty.

The I/O time is the time to copy the file from local cache. The file close time is the time to remove the tag from the cache which is in fact very small. If a tag for the file to be transferred in the local cache already exists, the file open time will be around zero. In the file read/write test, the tags in the cache are always null. So the file open time is actually the time to copy the file from remote server into local cache. If a tag for the file to be transferred in the local cache already exists, the file open time would be very small for both cases. The test showed that the advantage of GASS is that the user does not need to manually login on a remote machine to do a file transfer, and the caching mechanism makes the file transfer more efficient. As far as file access is concerned, the GASS APIs are easy to use and easy to be changed from standard Unix or C file access functions. But the user who transfers large files must have a large cache area. During the test we found that for some unknown reason the Gass server becomes unstable after some time of running. Besides, the Gass library `errno` does not coincide with the one of the standard C library.

Further tests have been performed in Torino using the INFN-GRID toolkit release 1.2, that is the INFN customization of Globus v1.1.3 (see Chapter 2). Some evidence about a performance problem has been found during the benchmark tests [38]. A huge decrease in the GASS transfer rate when the input file size is greater than 800 MB has been observed, as can be seen in Figure 6.4.

Each point in the picture corresponds to a different measurement. This behavior could be due to the GASS cache overhead or to some inefficiencies of the FTP protocol. The same problem has also been reported by the Globus Team (see Table 1 of the mentioned document [48]).

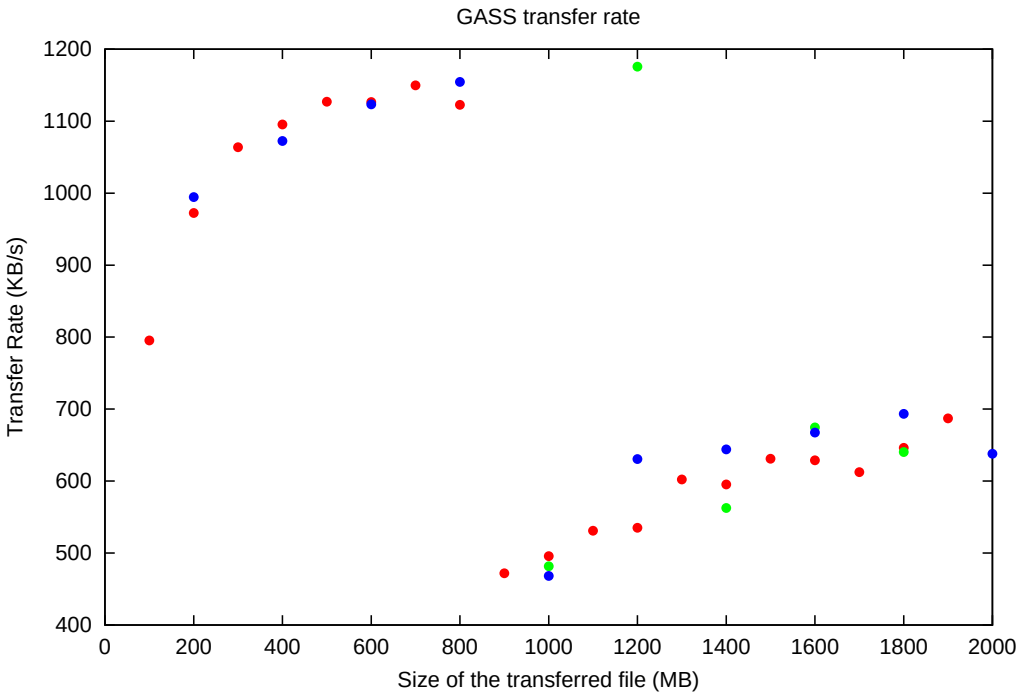


Figure 6.4: Measurement of the average GASS transfer rate as a function of the size of the file being transferred. The transfer rate decreases abruptly when the file size exceeds 800 MB. Each dot represents a different measurement.

6.2 GlobusFTP

As already mentioned, the GlobusFTP software is based on the GridFTP protocol [50]; it has been developed by the Globus Team. GridFTP is an extension of the FTP protocol [46] based on RFC949 [51], RFC2228 [52] and RFC2389 [53].

The GlobusFTP main characteristics are:

- A high throughput, reliable, secure and robust data transfer mechanism.
- GSSAPI security (PKI and Kerberos) support.
- Automatic negotiation of TPC buffer/window sizes.
- Parallel data transfer.
- Third-party control of data transfer.
- Partial file transfer.
- Reliable data transfer.
- Replica catalogs mechanisms.

At present the GlobusFTP implementation consists of a set of production libraries, a set of tools and some patches for third-party software. The production libraries and the tools are included in a special Globus release (alpha version 2) containing also patches that allow the use of GridFTP protocol by GASS and by the *globus_io* functions. For what concerns the third-party software, patches have been produced for the WUFTPD server and for the NCFTP client; they don't support the full GridFTP protocol. A complete description of the features implemented in each tool/software is reported in the mentioned web page [47]. The GlobusFTP software is available via CVS.

The test activity has been concentrating on some new functionalities of the GridFTP protocol, with particular attention to parallel data transfer and reliable data transfer. The replica catalog mechanisms will be probably tested by the people involved in WP2.2 (Data Management) of the INFN-GRID project.

Four INFN sites have been involved in the tests: Cnaf, Napoli, Padova and Torino. As expected, since we evaluated code explicitly labeled as an alpha release, we found various problems installing and configuring the GridFTP software (the software was changing very frequently, various bugs were found, etc...) and therefore a rather large effort has been spent to deploy and use it. We were anyway able to perform some tests using the gsiwuftpd server release 0.4b5, the gsincftp client release 0.3 and the *globus-url-copy*

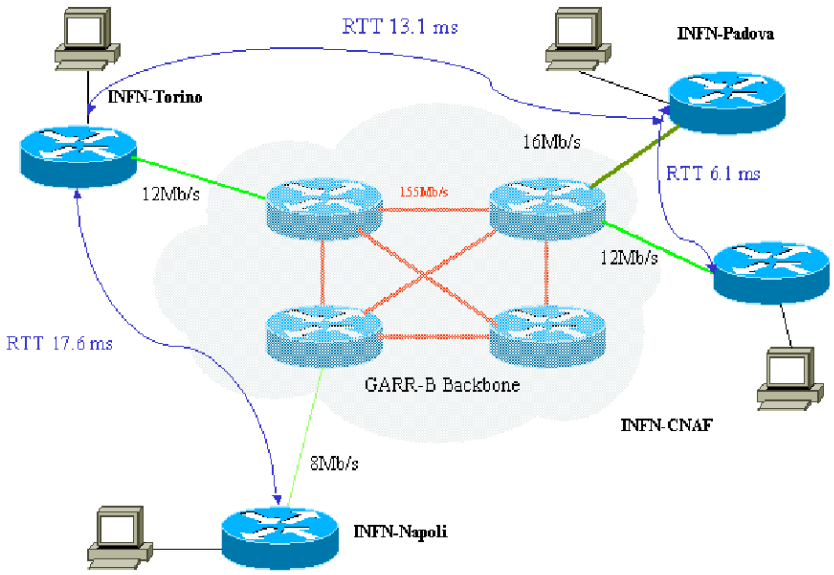


Figure 6.5: Network layout of the hosts and sites used for the GlobusFTP tests

tool. Further systematical tests will be probably performed using the GridFTP APIs in the frame of the Network Work Package (WP5) of INFN-GRID project.

A brief description of the tests and their results is reported in the next paragraphs.

Figure 6.5 shows the network layout used for the tests. The four sites are connected by production links provided by the italian National Research Network (GARR-B).

At each site a PC with the following configuration has been used for the tests:

Operating System	Linux Red Hat 6.1
Globus Toolkit	alpha rev. 2 release (includes *globus-url-copy* v1.1.3b14)
FTP server	gsi-wuftpd-0.4b5
FTP client	gsincftp-0.3

The default values for the socket buffer sizes have been used:

```
/proc/sys/net/core/rmem_default   65535 // default receive window
/proc/sys/net/core/rmem_max       65535 // maximum receive window
/proc/sys/net/core/wmem_default   65535 // default send window
/proc/sys/net/core/wmem_max       65535 // maximum send window
```

At first the capability of resuming an interrupted file transfer and the support for the Globus Security Infrastructure authentication mechanisms have been succesfully tested.

Then some throughput tests using parallel data transfer (up to 64 streams) have been performed. We observed that, given the default TCP socket buffer configuration, the relationship between throughput and the number of parallel streams seems to vary according to the load on the data path - see Figure 6.6(a). The throughput increases for up to 16 streams and then keeps constant or, alternatively, decreases to 1/3 of the performance achieved with just one stream, depending on the traffic direction. One possible interpretation of this phenomenon is that routers on a lightly congested data path are more sensitive to TCP burstiness than router on a non-congested path. When the number of parallel streams increases, the aggregate burstiness produced by ftp increases as well with a consequent increase in packet loss. In case of unloaded data paths (top line in Figure 6.6(a)), performance keeps constant. An increase in throughput is only visible if the $RTT * capacity$ product is less then the configured window. In case of large values of this product the default window size is not enough to keep the line busy; as a consequence the increase of parallel streams improves performance.

Another test was related to the evaluation of the relationship between performance and file size with multiple streams (8 flows). The results show that performance is approximately the same with the only exception of very small file sizes (1 Mbyte). In this case, low performance is caused by the fact that TCP does not get out of the slow start phase at connection set-up due to the limited file transfer time. In case of a lightly loaded data path with very large files, end-to-end performance can slighty decrease (see Figure 6.7).

Next, the relationship between performance and the TCP socket buffer size has been evaluated. We observed that the performance is greatly infuenced by the TCP socket buffer (send and receive) set by the application. The gain in throughput depends on the RTT between the source and the destination. The greater the RTT, the greater is the increase in performance if the TCP socket buffers are properly dimensioned. As shown in Figure 6.8, with a RTT of approximately 15 msec, throughput gets stable with socket buffer size equal to 32 Kbytes.

The last test aimed to evaluate the relationship between throughput and the block size. The block size corresponds to the amount of data copied from the application memory space to the kernel memory space. Normally, with large block sizes performance increases, since it's inversely proportional to the number of interrupts generated by the system call `write()`. Figure 6.9 shows that, given the relatively small amount of available capacity, for data rates of approximately 10 Mbps the CPU of the end-system is not a bottleneck. For this reason, an increase in the block size does not produce a corresponding increase in throughput. The same conclusion can be drawn with both single-stream or multiple-stream file transfers.

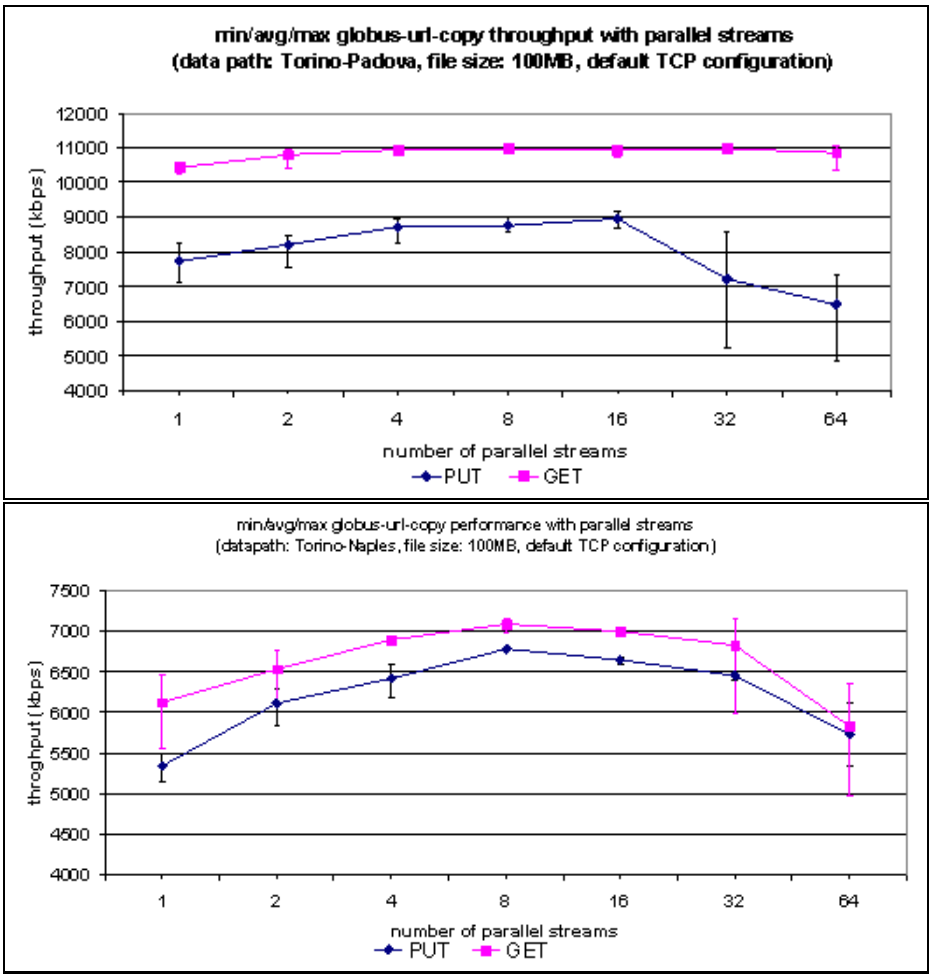


Figure 6.6: FTP throughput with parallel streams on the Torino-Padova datapath (a) and on the Torino-Naples datapath (b).

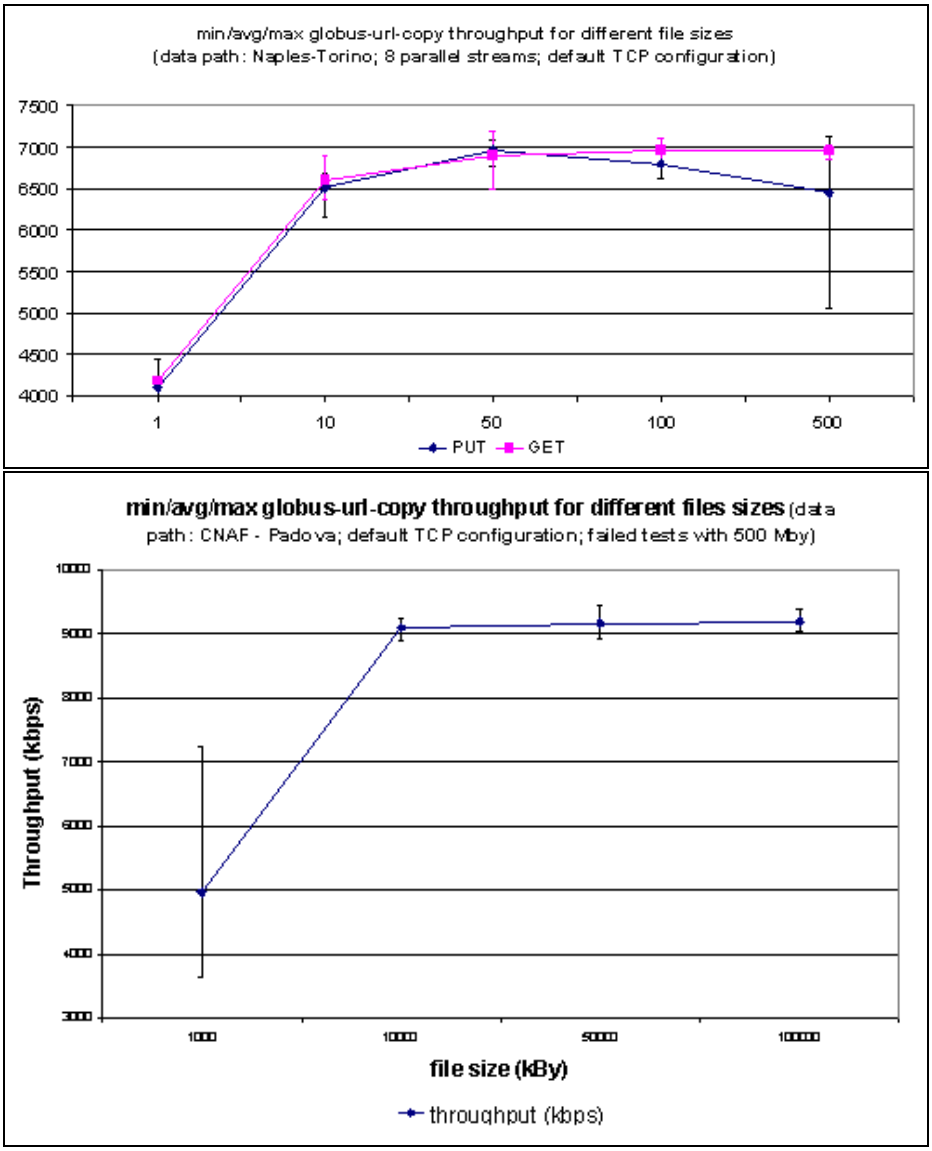


Figure 6.7: Relationship between throughput and file size on two different data paths: Torino-Napoli (a) and CNAF-Padova (b).

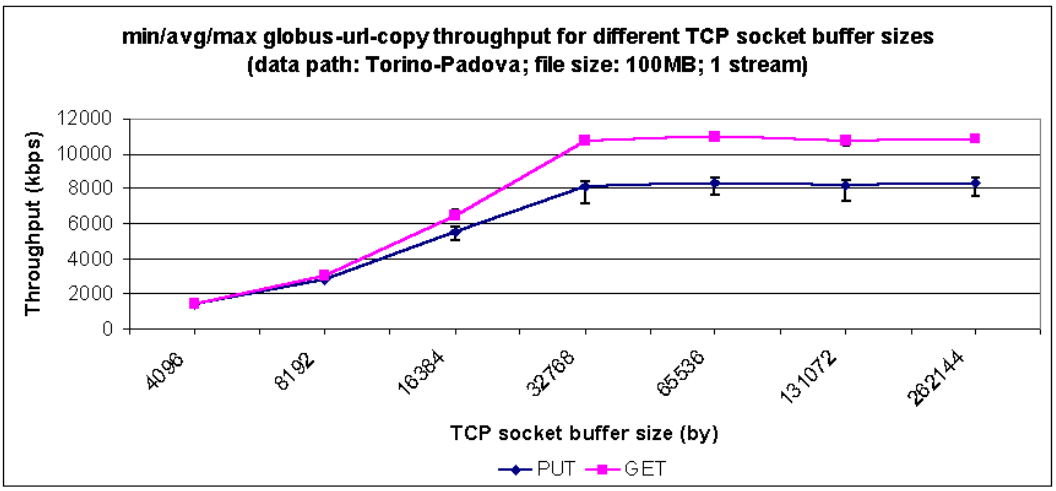


Figure 6.8: FTP throughput vs TCP socket buffer size.

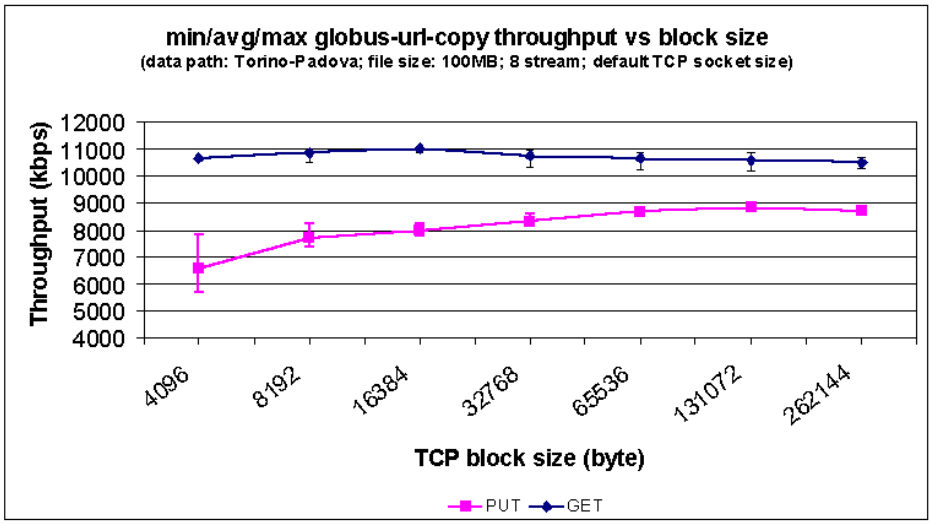


Figure 6.9: FTP throughput vs. block size.

During the tests described above some problems (bugs) have been found: the system crashes when using *globus-url-copy* with 64 streams and 500 Mbytes of data; the network interface goes down after parallel transfers of large files (500 Mbytes) which frequently freeze. Besides, some problems with 16 streams (between CNAF and Padova), with 32 or 64 streams between Turin and other sites have been observed. They were reported to the Globus Team.

Since the tests are not exhaustive and some problems have been found, some more test activity will be done in the frame of the Network Work Package (WP5) of the INFN-GRID project. The availability of dedicated circuits (VPN) for the tests will also be investigated.

Chapter 7

Other services

In the work plan [6] we proposed to evaluate other two Globus services: the Globus Executable Management (GEM) service, and the Heartbeat Monitor (HBM) service.

7.1 Globus Executable Management

According to the Globus documentation [29][54], the Globus Executable Management (GEM) service should provide mechanisms to implement different distributed code management strategies, providing services for the identification, location and instantiation of executables and run time libraries, for the creation of executables in heterogeneous environments, etc.

Actually, we found that GEM doesn't exist as a package, and Globus can only provide some functionalities, to do just executable staging, that is transfer the application (i.e. the executable file) to a remote machine immediately prior to execution. This is possible if the executable file is accessible via HTTP or HTTPS, or present on the machine on which the *globusrun* command is issued.

This executable staging does not do anything with regard to moving shared libraries with the executable and setting the `LD_LIBRARY_PATH` environment variable, so if shared libraries are in non-standard places on the target machine, or if the application uses non-standard shared libraries, then this application will probably fail.

Nothing exists in the Globus toolkit about the packaging and portability issues that would allow new executables to be automatically built for a new architecture from some portable source packages.

7.2 Heartbeat Monitor

The Heartbeat Monitor (HBM) service [29][55] should provide mechanisms for monitoring the status of a distributed set of processes. Through a client interface, a process should be allowed to register itself with the HBM service, and sending regular heartbeats to it. Moreover, a data collector API should allow a process to obtain information related to the status of other processes registered with the HBM service, thus allowing to implement, for example, fault recovery mechanisms. Unfortunately this service is not seeing active development: an HBM package, implementing some very preliminary and incomplete functionalities, has been included in the early Globus releases, but now it is not supported anymore, and has been dropped from the distribution.

Chapter 8

HEP Application Experiences

This chapter describes the preliminary activities performed by some experiments within INFN, in order to evaluate the Globus services for a possible use in their activities.

8.1 Alice

In the framework of the work being done for the preparation of the Physics Performance Report (PPR), the ALICE Collaboration has put in place since November 2000 a task force with the aim to evaluate Globus services for a possible use during the distributed data production expected in 2001. The first part of the PPR will be the simulation of about 10000 Pb-Pb central collisions at the LHC energy, each of them producing about 80000 primary particles. AliRoot, the ALICE simulation program making use of ROOT as a framework, will be the software tool for the layout simulation. At the moment, the output size is about 1.5 GB/event for the ”hits” (track impact points on the detectors), and about 0.15 GB/event for the ”digits”, containing the simulation of the detector response. Therefore, we expect an overall output size of about 16.5 TB. Many institutions will participate to the PPR production. Among them, four INFN sites: Cagliari, Catania, Padova and Torino. The Alice Collaboration has decided to set up a short-term testbed to verify whether Globus could be used for a central control of the whole data production. Cagliari, Catania, Lyon, Ohio State and Torino configured PC farms with Globus and different local job managers (PBS, LSF). Within the INFN sites, Globus has been deployed using the INFN-GRID distribution toolkit. At the same time, a standard installation kit for the ALICE software (ROOT + AliRoot) is being prepared, as long as a set of standard, site independent scripts for the job submission. INFN, Globus, and IN2P3 CA certificates have been managed without any problems although for the first tests a ”manual” configuration of the gatekeeper nodes (certificates and grid-mapfiles) has been adopted. In order to simulate a real production, all certificates have been mapped to a unique local account.

With a minimum amount of assumptions, using a few common environment variables, the local installation of the software has been made transparent for the remote user, and full ALICE simulated events, taking more than 24 hours to be generated, were successfully remotely ran, with a local staging of the standard output and error on the submitting machine. On the other hand, the big ROOT output file will be stored remotely, with a copy (or possibly FTP) to the final remote destination. All of this was based on the *globusrun*, *globus-job-run*, *globus-job-submit* commands. During the tests, jobs have been submitted "on purpose" by the user on a given remote machine without using any GIS information or broker action. Anyhow, a monitoring of both the local farms and the test-bed has been put in place. Examples can be found at [56] and [57]. The outcome of the tests, in view of the forthcoming production, can therefore be considered positive, since we reached our goal. At the same time, we however realized that some features could be easily managed only because of the fact that this production involves a simple and standard input for the application to be run and a limited number of users involved. For the second part of the production, we expect Globus/DataGrid to implement at least the features we list here:

- A tool for the automatic upgrade of the grid-mapfiles, getting the information from a central (LDAP?) server. On this purpose, we have already created an 'Alice' branch in the INFN LDAP server installed at `bond.cnaf.infn.it` (see Section 3.4) with a series of sub-trees which could be mapped to different Alice sub-detectors in order to sort users and establish different running priorities on the local farms. We are currently waiting for the INFN CA manager to populate the LDAP directory. At the same time we are discussing within the Collaboration the possibility to have similar LDAP servers working in other sites in a short amount of time;

- A standard definition of the Certification Authorities to be allowed, possibly with an automatic cross check when updating the central server and/or the grid-mapfile;

- A tool to access the distributed input, eventually with an automatic interface to the Database managing the bookkeeping;

- A (even rough) workload manager.

8.2 ATLAS

Atlas will be involved in a big production to finalize the study of the Barrel Muon Trigger. In particular, for system optimization, $O(10^7)$ single muon events, $O(10^4)$ background events and $O(10^5)$ physics events will be generated and analyzed by May 2001,

while performance studies for the HLT Technical Design Report (TDR), due in 2002, will require productions one order of magnitude larger.

The application that has been chosen to test the GRID environment is a part of the software chain and consists in the full event simulation in the Atlas detector. This task requires approximately 30 SpecInt95*sec/event for single muons, 10^5 SpecInt95*sec/event for background and 10^4 SpecInt95*sec/event for relevant physics channels.

The test has been performed between two sites: Milano with one PC and Cern with one of the five PC's of the IT farm dedicated to GRID testbed. The installation of the Grid tools on both sites has been done with the INFN Installation Toolkit version 1.2 (Globus version 1.1.3). X509 certificates signed by the INFN CA were used throughout.

The initial step has consisted in making the application independent from the common and uniform filesystem provided by AFS, putting in one single tar file all the files needed to run the job: the datacards (`dice.datacards`), the executable (`dice.exe`) and all the data files needed have been provided by the Rome group which is directly involved in the muon production.

A first test has been performed between Milan and Cern and consisted in the following steps:

1. send to remote host (Cern) the tar file (*dice.tar*), and decompress it
2. submit from local host (Milan) a simulation job on a remote host using the fork service
3. monitor locally the execution of the remote application
4. receive back from remote host the output files: zebra file(`ZEBRA.O`) and the output file with histograms (`atlas.his`).

To execute the actions explained above, as first attempt, *globusrun* with RSL (Resource Specification Language) scripts have been used but a first problem has been encountered when trying to transfer the tar file from Milan host to Cern host using GASS (Globus Access to Secondary Storage) protocol that should permit to access and transfer remote data.

Unfortunately the tar file ($\sim$ 50 MB) is transferred only partially without any error messages: we tried to transfer the same file between other italian hosts with the same result, so we decided to submit to the Globus support mailing list the problem, but we are still waiting for an answer [1].

[1] An answer from the support group was received on Feb 22, 2001, too late for it to be tested in time for this report.

To overcome this problem we found a useful tool to handle the file transfer and I/O redirection in Bypass [58], an interposition agent building software developed by the University of Wisconsin - Madison Condor team. With the help of INFN-GRID people, we wrote a Bypass agent to transfer the tar file, submit remotely the simulation program, monitor the remote job execution (thanks to the log-like standard output file which is updated locally while the job is running) and receive back the output. This solution (Bypass-instrumented jobs submitted via Globus *globusrun*) has proved to be workable: 10.000 single muon events have been full simulated remotely (on the Cern host) and the output files: `ZEBRA.O` $\sim$ 150 MB and `atlas.his` have been received successfully on the Milan host.

This work is still in progress, actually at the moment we are still at the level of identification and evaluation of a minimal set of GRID tools to permit basic functionalities like e.g. file transfer, access to remote files and resources. We found Bypass a useful and working tool and we suggest that this tool could be added to one of the future releases of the INFN Installation Toolkit.

The next steps we intend to do consist in going on testing the functionalities of GRID tools and identifying which tools satisfy at best our application requirements, then we plan a wider test of the same application on the Rome Linux farm, presently made of 38 Pentium III 800 MHz processors, with $\sim$1 TB disk space.

8.3 CMS

CMS is going through a big effort in order to produce the data samples needed for the High Level Trigger studies. The production schema is summarized in Figure 8.1. We can identify 5 steps. For each step the numbers refer to a job with a typical size (500 events):

1. **Simulation of the physical process.** This step is performed by PYTHIA (FORTRAN program). Uses as input file a set of datacards (a few bytes) and the output is an ntuple (RZ file) tipically 40 MB and a text file (a few bytes). The CPU needed varies between 15 and 2000 KSpecInt95*sec depending on the physics channel.

2. **Simulation of tracking in the detector.** This step is performed by CMSIM (FORTRAN program, uses GEANT 3). Uses as input the ntuples produced by PYTHIA and a text file with datacards (a few bytes). The output is an FZ file, tipically 500 MB large, containing the so called hits. The CPU needed is about 1000 KSpecInt95*sec.

3. **Hit Formatting.** This step is performed by ORCA (C++ program). Moves the content of the FZ files to an Objectivity/DB federation. The input are the FZ files

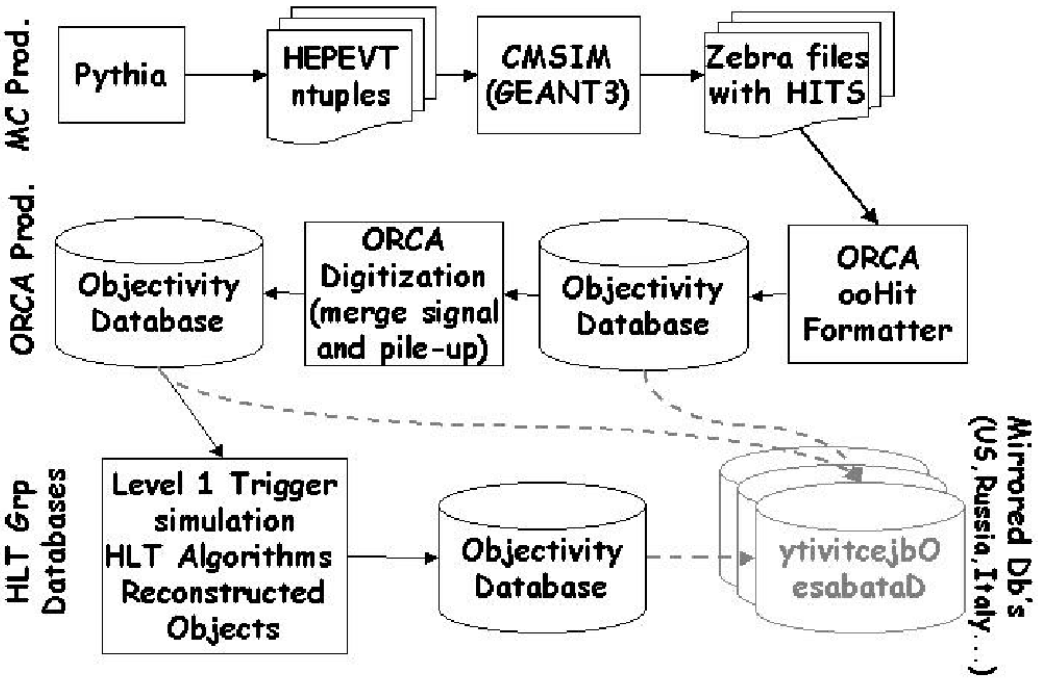


Figure 8.1: CMS Production chain

produced by CMSIM and a text file (a few bytes). The output has almost the same size of the input. A very small amount of CPU is needed.

4. **Digitization (simulation of detector response).** Is performed by ORCA. Reads the hits from the database, combines them with the hits of the pile-up events (about 200 pile-up events are mixed with each signal event) and write the so called digis i.e. the raw data. The total amount of data read in is about 100 GB and the data written to the database are about 500 MB (1 GB if the tracker is also treated). The CPU required is about 1000 KSpecInt95*sec (2000 KSpecInt95*sec if the tracker is also treated)

5. **Reconstruction.** This step is performed by ORCA. This is the chaotic phase of the chain, since the kind of access is not predictable.

Starting in spring 2000 the first part of the production chain (PYTHIA + CMSIM) has been implemented at several italian (INFN) sites. Dedicated computing farms have been set up in a consistent way using a linux installation toolkit that took care also of installing CMS specific software and all of the needed packages (e.g. CERNLIB, Objectivity/DB, etc). The configuration included a monitoring system which allowed to check

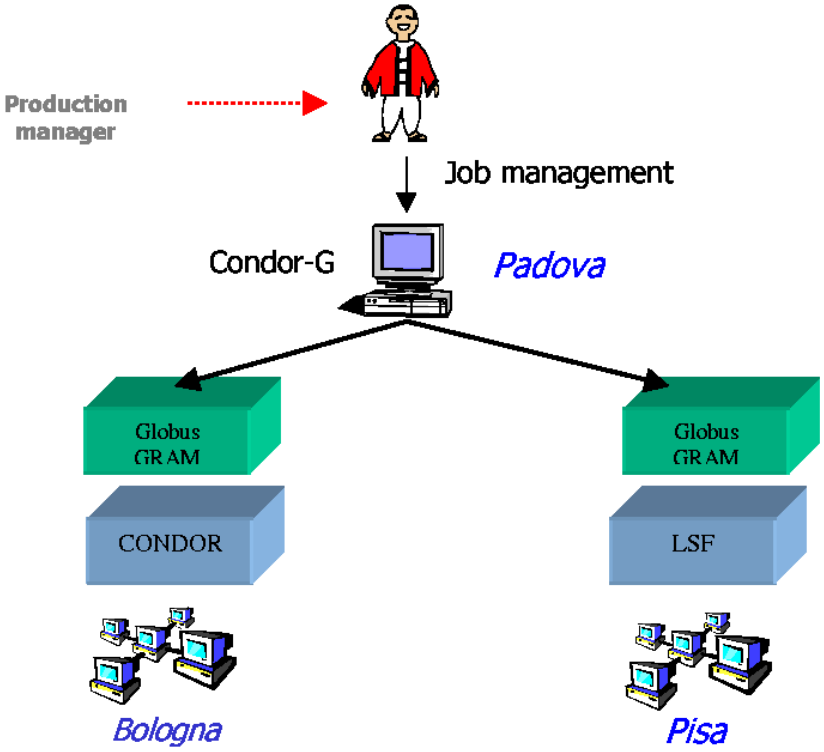


Figure 8.2: Testbed for the Grid CMS production

the status of the productions in the farms as well as the status of the farm nodes from the web. On the front end machines of Bologna and Pisa farms Globus has been installed using the INFN installation toolkit, using INFN setup (see Chapter 2) and certificates provided by the INFN CA. On the Bologna gatekeeper the GRAM has been interfaced with a local Condor scheduler, on the Pisa one with an LSF scheduler. The job management (the submission of jobs to these farms, the job status monitoring, etc.) has been done from a single location (Padova) by a single user: the production manager; the subject of his certificate (signed by the INFN CA) has been mapped in the grid-mapfiles of the Bologna and Pisa farms to a specific local production account, with the necessary environment properly defined. As job submission service Condor-G has been used, and therefore the job management has been done using the Condor commands and tools (*condor_submit*, *condor_q*, the Condor logviewer tool,).

For these jobs the executable and the input files were already stored in the executing machine, while the log files produced by Condor-G were created in the submitting machine. The output files (the FZ and the text output files) were created in the file system

of the executing machine, albeit we felt that it could be useful to have the standard output of the jobs on the submitting machines since these log-like files are useful to determine if the applications are running correctly. For this purpose we found that the Bypass software [58], in particular the Grid Console implementation, could be a viable solution for this problem: we are now testing the new release of this software, resilient to different types of failures. Instead these files were available through the web interface of the farm monitoring. No schedulers/brokers were used, and therefore it was up to the production manager to decide in which Globus resource (farm) the jobs had to be submitted. About 20 jobs per farm were submitted. This setup worked, but we found that the submission via Condor-G triggered some (many) memory leaks in the Globus job managers, running in the front-end machine, and therefore the production crashed for lack of memory on the gate-keeper after about 2 hours (having one job manager for each job, as explained in section 5.3, made this situation worse). It's worth to say that the problem was easily identified thanks to the monitoring of the memory usage on the farm nodes, performed using the standard monitoring system of the farm. We have then been able to provide fixes for these memory leaks, that have been included in the INFN- GRID distribution (and that will be hopefully included in the next official Globus release): therefore it is now possible to repeat these tests again.

At the end of 2000 in Bologna were also performed the following parts of the production chain, running ORCA to produce Objectivity/DB files containing hits and digis for the events produced in the other italian farms. About 100 GB of data were produced. To move the data to CERN another Globus-based tool has been used. GDMP is a tool developed at CERN for Objectivity/DB database transfer through the grid. Pisa was chosen as the italian front-end site to CERN for GDMP data transfers. Since at CERN only certificates signed by the Globus CA were accepted at the time of the test, the Pisa front end machine was configured in such a way that it was able to work with Globus certificates but was also able to accept INFN ones. In Bologna GDMP was installed using the INFN installation toolkit. In Bologna the certificates of the persons in charge of production at Pisa were mapped to the local production account and vice versa. After that, transfers from Bologna to Pisa could be carried on using the GDMP tools in a couple of days. The main problem encountered was related to the expiration of the grid proxy after 12 hours. This caused the interruption of the file transfers. After the proxy was renewed (for an appropriate amount of time) the transfer could be resumed at the point it was suspended.

The features expected for the (next) future include:

- An even rough broker, able to automatically choose the best resources where to submit the jobs;

- A new version of GDMP based on the new Globus Replica Manager.

Chapter 9

Conclusions

The general breakdown and analysis of the Grid computing problems that underlies the design of the Globus toolkit is sound and appealing, so the main purpose of the 6-month technical evaluation process we describe in this document was to assess the organization and capabilities of the Globus toolkit at the implementation (or "production") level. In particular, we proposed [5] "to find which services can be useful for our needs, what is missing, what is necessary to integrate/modify".

We found that many of the Globus services that are actually seeing active development, especially in the Security (see Chapter 3), and Resource Management (see Chapter 5) areas, fit very well with our requirements. In the area of the Information Services we may require some design modifications (see the specific conclusions in Section 4.2). As for the Globus services for data management, we think that these functionalities are of great interest, but further tests and investigations are needed.

We were also interested in evaluating the process of feeding fixes and modifications back into the Globus support and development teams, and to see how these interaction would reflect in the Globus toolkit release process.

During the evaluation and test phase of the current Globus release, a number of show-stopping bugs appeared. These bugs were submitted to the `support@globus.org` mailing list, but the rather long average response time (an "official", funded Globus support service is not in place), along with our activity deadlines, prompted us to spend a good amount of resources in actual bug tracking and bug reporting: we submitted to the support mailing lists on the order of 20 bug fix patches. In many cases we had a fruitful collaboration with members of the support team, and received assurance that the appropriate bug fixes had been committed to the Globus code repository. Unfortunately, no new releases of Globus appeared over our 6-month evaluation period and up to the appearance of this document, so we cannot be sure that the fixes to some of the problems we reported

actually made their way down to the official codebase[1].

As we reported in some of the meetings we had with Globus developers early on during this evaluation phase, part of the problems may just lie in the "closed" development scheme adopted for Globus (no "public" CVS site to keep in sync with what's actually happening), so that the contacts between us and the development team were confined to the "software support" model, with the Globus team basically acting as a service provider. A genuine collaboration channel definitely needs to be established for larger-scale projects such as DataGrid.

We also have some concerns about the long-timescale maintainability of the current Globus code base, for the lack of "code" (with respect to "design" or "functionality") documentation.

We would like to thank the Globus design and development team for the collaborative interaction: in particular Steve Tuecke, Lee Liming, Steven Fitzgerald, Ann Chervenak, Doug Engert, Stuart Martin.
We also wish to thank Roberto Cecchini, the manager of the INFN CA, for the very fruitful collaboration.

[1] At the March 2001 GGF1 event we were informed that we may now have access to α-release code of most (if not all) of the Globus packages.

Appendix A

Standard LDAP objectclasses

Here is a detailed listing of some of the standard LDAP objectclasses that can be useful to describe GRID users [15] (see Section 3.4).

Person (Child of top)	
Required Attributes	**Description**
objectClass	Defines the object classes for the entry
cn (commonName)	The person's common name
sn (surName)	The person's surname, or last name
Allowed Attributes	**Description**
description	Text description of the person
seeAlso	URL to information relevant to the person
telephoneNumber	The person's telephone number
userPassword	Password with which the person can bind to the directory

organizationalPerson (Child of person)	
Required Attributes	**Description**
objectClass	Defines the object classes for the entry
Allowed Attributes	**Description**
destinationIndicator	The country and city to provide Telegrams
fax (facsimileTelephoneNumber)	Fax number
internationalIsdnNumber	ISDN number
l (localityName)	Location at which the person resides
ou (organizationUnitName)	Organizational unit
physicalDeliveryOfficeName	Location where physical deliveries can be made
postalAddress	The person’s mailing address
postalCode	The person’s postal code
postOfficeBox	The person’s post office box
preferredDeliveryMethod	The preferred method of contact or delivery
registeredAddress	Postal address
st	State or province in which the person resides
street	Street address at which the person is located
teletexTerminalIdentifier	Identifier for the teletex
telexNumber	Telex number of the organization
title	The person’s job title
x121Address	X.121 address of the organization

inetOrgPerson (child of person)	
Required Attributes	**Description**
objectClass	Defines the object classes for the entry
Allowed Attributes	**Description**
audio	Contains a sound file in binary format
businessCategory	Business in which the person is involved
carLicense	The license plate number of the person's vehicle
departmentNumber	Department for which the person works
employeeNumber	The person's employee number
employeeType	The person's type of employment (for example, full time)
givenName	The person's given, or first, name
homePhone	The person's home phone number
homePostalAddress	The person's home mailing address
initials	The person's initials
jpegPhoto	An image in JPEG format
labeledUri	Universal resource locator that is relevant to the person
mail	The person's electronic mailing address
manager	Distinguished name representing the person's manager
mobile	The person's mobile phone number
pager	The person's pager number
photo	Contains a photo, in binary form
preferredLanguage	Defines a person's preferred written or spoken language
roomNumber	the room number in which the person is located
secretary	The person's secretary or administrator
uid	Identifies the entry's userid (usually the logon ID)
userCertificate	Contains a user's certificate in cleartext (not used)
userCertificate;binary	Contains a user's certificate in binary form
userSMIMECertificate;binary	Contains a user's certificate in binary form (S/MIME)
x500UniqueIdentifier	Undefined

groupOfNames (child of top)	
Required Attributes	**Description**
objectClass	Defines the object classes for the entry
cn (commonName)	The group's common name
Allowed Attributes	**Description**
businessCategory	Type of business in which the group is engaged.
description	Text description of the group's purpose.
o (organizationName)	Organization to which the group belongs.
ou (organizationUnitName)	Organizational unit to which the group belongs.
owner	The group's owner.
seeAlso	URL to information relevant to the group.
Member	A group member in distinguished name format.

certificationAuthority (child of top)	
Required Attributes	**Description**
objectClass	Defines the object classes for the entry
cACertificate	Certificate, in binary form, from a certification authority
Allowed Attributes	**Description**
authorityRevocationList	Revoked CA certificates
certificateRevocationList	Revoked user certificates
crossCertificatePair	CA cross trusting

Bibliography

[1] Home page for the INFN-GRID project.
http://www.infn.it/grid

[2] Home Page for the Workload management workpackage of the DataGrid Project.
http://www.infn.it/workload-grid

[3] Home Page for the DataGrid Project.
http://www.eu-datagrid.org

[4] Home page for the Globus project.
http://www.globus.org

[5] Home page for the Globus evaluation activity of the INFN-GRID project (described in this document).
http://www.infn.it/globus

[6] *A computational and data challenge for future INFN experiments: a GRID approach*.
http://www.infn.it/grid/doc

[7] http://www.pi.infn.it/grid/dist

[8] http://www.infn.it/grid/dist

[9] http://www-unix.mcs.anl.gov/∼smartin/repackage.htm

[10] http://www.pi.infn.it/GRID/dist/GRID_INST_1.2.html#Sec0301

[11] *Models of Networked Analysis at Regional Centres for LHC Experiments (MONARC): Phase 2 Report*
http://monarc.web.cern.ch/MONARC/docs/phase2report/Phase2Report.pdf.

[12] Home page for the OpenSSL project.
http://www.openssl.org/

[13] http://security.fi.infn.it/CA/

[14] http://security.fi.infn.it/CA/policy.html

[15] M. Wahl, *A Summary of the X.500(96) User Schema for use with LDAPv3*, RFC2256

[16] B. Didier, K.Schuchardt, G. von Laszewski, S. Fitzgerald *Representing People for the Grid Information Services*.
http://www-unix.mcs.anl.gov/gridforum/gis/reports/people/people.pdf

[17] Home page for the OpenAFS project.
http://www.openafs.org

[18] Home page of the Global Grid Forum Information Services Working Group.
http://www-unix.mcs.anl.gov/gridforum/gis

[19] Notes on extending the GRIS information schema.
http://www.globus.org/mds/extending-gris.html

[20] http://perl-ldap.sourceforge.net/

[21] Olivier Dubuisson, *ASN.1 - Communication between heterogeneous systems*.

[22] A. Gulbrandsen, P. Vixie, L. Esibov, *A DNS RR for specifying the location of services (DNS SRV)*, RFC2782

[23] E. Stokes, B. Blakley, D. Rinkevich, R. Byrne, Internet-Draft LDAP Extensions WG , *Access Control Model for LDAPv3 / draft-ietf-ldapext-acl-model-06.txt*, 14 July 2000

[24] J. Allen, M. Mealling *The Architecture of the Common Indexing Protocol (CIP)*, RFC2651

[25] Giuseppe Lo Biondo *GIIS configuration for INFN sites* November 2000.
http://www.mi.infn.it/globus

[26] *The SLAPD and SLURPD Administrators Guide* University of Michigan Release 3.3 April 30, 1996
http://www.umich.edu/~dirsvcs/ldap/doc/guides/slapd/

[27] Home page for the MRTG project.
http://ee-staff.ethz.ch/~oetiker/webtools/mrtg/

[28] K. Czajkowski, I. Foster, N. Karonis, C. Kesselman, S. Martin, W. Smith, S. Tuecke, *A Resource Management Architecture for Metacomputing Systems*. Proc. IPPS/SPDP '98 Workshop on Job Scheduling Strategies for Parallel Processing,1998
ftp://ftp.globus.org/pub/globus/papers/gram97.pdf

[29] I. Foster, C. Kesselman, *The Globus Project: A Status report, Proc. IPPS/SPDP '98 Heterogeneous Computing Workshop*, pg. 4-18, 1998.
ftp://ftp.globus.org/pub/globus/papers/globus-hcw98.pdf

[30] Massimo Sgaravatto, *First Evaluation of the Globus GRAM Service*.
http://www.infn.it/globus/Docs/gram-report.pdf

[31] Massimo Sgaravatto, *Report from visits to Condor (Madison) and Globus (ANL) teams*.
http://www.infn.it/workload-grid/docs/20000802-report-usa.pdf

[32] A. Forte, A. Guarise, *Valutazione preliminare dello scheduler PBS*.
http://www.to.infn.it/grid/globus/pbs+globus.ps

[33] F. Giacomini, *Early use of PBS as a jobmanager for Globus*.
http://www.infn.it/globus/Docs/pbs-globus.txt

[34] Home Page for the INFN "Condor on WAN" Project.
http://www.infn.it/condor

[35] Globus Quick Start Guide.
http://www.globus.org/toolkit/documentation/QuickStart.pdf

[36] *The Globus Resource Specification Language RSL v1.0*.
http://www.globus.org/gram/rsl_spec1.html

[37] M. Biasotto, M. Sgaravatto, *Providing the Grid Information Service with information of local farms*.
http://www.infn.it/globus/Docs/gis-farm.pdf

[38] A. Guarise, *Analisi preliminare sulle performance del toolkit Globus*.
http://www.to.infn.it/grid/globus/globus-benchmarks.pdf

[39] http://www-unix.mcs.anl.gov/mpi/

[40] http://www-unix.mcs.anl.gov/mpi/mpich/

[41] http://www.hpclab.niu.edu/mpi/

[42] ftp://ftp.pr.infn.it/pub/linux/rpm/contrib/

[43] http://www-unix.mcs.anl.gov/mpi/mpptest/

[44] ftp://ftp.pr.infn.it/pub/bench/

[45] Administrator Guide to GARA, March 2000.
http://www-fp.mcs.anl.gov/qos/papers/gara_admin_guide.pdf

[46] GridFTP extensions to the FTP protocol.
http://www.globus.org/datagrid/deliverables/globus_ftp_control/extensions.html

[47] http://www.globus.org/gsiftp-alpha2

[48] J. Bester, I. Foster, C. Kesselman, J. Tedesco, S. Tuecke, *GASS: A Data Movement and Access Service for Wide Area Computing Systems*.
ftp://ftp.globus.org/pub/globus/papers/gass.pdf

[49] http://www.pi.infn.it/GRID/wp2/gass_test.ps

[50] White Paper, *GridFTP Universal Data Transfer for the Grid*.
http://www.globus.org/datagrid/deliverables/C2WPdraft3.pdf

[51] M. Padlipsky, *FTP Unique-Named Store Command*, RFC949

[52] M. Horowitz, S. Lunt, *FTP Security Extensions*, RFC2228

[53] P. Hethmon, R. Elz, *Feature negotiation mechanism for the File Transfer Protocol*, RFC2389

[54] http://www.globus.org/hbm/heartbeat_spec.html

[55] P. Stelling, I. Foster, C. Kesselman, C.Lee, G. von Laszewski, *A Fault Detection Service for Wide Area Distributed Computations*, Proc. 7th IEEE Symp. on High Performance Distributed Computing, 268-278, 1998.
ftp://ftp.globus.org/pub/globus/papers/hbm.pdf

[56] http://alipc1.ct.infn.it/mrtg/monitoring.html

[57] http://alipc1.ct.infn.it/mrtg/netmon.html

[58] http://www.cs.wisc.edu/condor/bypass/