GWD-I
Category: Informational
Production Grid Management Research Group
http://www.ipg.nasa.gov/ggf-pgm-rg/
draft-ggf-pgm-usecasegrip-1.1.doc

Dietmar Erwin, Forschungszentrum Jülich
Michael Rambadt, Forschungszentrum Jülich
Philipp Wieder, Forschungszentrum Jülich
June 2, 2003

Production Management Use Case: The Grid Interoperability Project

Status of This Memo

This memo provides information to the Grid community on experiences with a heterogeneous, interoperable Grid. It does not define any standards or technical recommendations. Distribution is unlimited.

Copyright Notice

Abstract

This use case document describes the experience gained running the software created in the projects UNICORE Plus[1] [1] and GRIP[2] [2] in a production environment. The software has the acronym UNICORE (**Un**iform **I**nterface to **Co**mputing **Re**sources). It has been originally conceived to provide seamless, secure and intuitive access to distributed resources for the German high performance computing centres and their users. UNICORE was already deployed and used on the production systems of the partners during the project phase (January 2000 till December 2002). The German HPC centres (FZ Jülich, LRZ München, and HLRS Stuttgart) have signed maintenance agreements with Pallas GmbH, the distributor of the UNICORE software, for professional support and enhancements after the end of the project.

GRIP (**Gr**id **I**nteroperability **P**roject) has been started in 2002 to combine the unique strength of UNICORE with those of Globus [3]. By the end of 2002 the project demonstrated that UNICORE users can submit jobs transparently to resources controlled by Globus or UNICORE and retrieve the results through the UNICORE client. The implementation was based on Globus Toolkit 2. GRIP shifted its focus with the advent of the Open Grid Service Architecture (OGSA [4]) and will implement interoperability with Grid Services as they become available during 2003.

The experience with Grid software in production mode results primarily from the UNICORE system. The important interoperability issues have been uncovered through the integration of the production environments with dedicated systems running software under development and by combining different user communities and organizations. This experiences show what can be expected when virtual organizations are to be created and managed.

[1] UNICORE Plus was funded in part through BMBF Grant 01 IR 001
[2] GRIP is funded in part through EC grant IST-2001-32257

Contents

1 Introduction

1.1 Objectives of UNICORE and GRIP

To achieve seamless, secure, and intuitive access to distributed resources, the UNICORE project had the following objectives:

- UNICORE was to hide the seams resulting from different hardware architectures, vendor specific operating systems, incompatible batch systems, different application environments, historically grown computer centre practices, naming conventions, file system structures, and security policies.
- Security was to be built into the design of UNICORE from the start relying on the X.509 standard for certificates to authenticate servers, software, and users and encrypt the communication over the internet.
- UNICORE was to be usable by scientists and engineers without having to study vendor or site-specific documentation. A graphical user interface was to be developed to assist the user in creating and managing complex jobs and to integrate important applications.
- The administrative autonomy of participating sites including the decision who may use the resources has been retained. UNICORE is flexible enough to adapt to existing proven practices at the participating centres.

Project GRIP added the following objectives:

- Make Globus resources available to UNICORE users without changing the architecture of either system.
- Support applications in a combined UNICORE – Globus Grid without modifying the application's source code.

1.2 The UNICORE Architecture

To define the terminology for the remainder of the paper a brief description of UNICORE's three tier architecture is given:

- The user is running the UNICORE Client on a local workstation or PC.
- On the top level, each participating computer centre defines one or several UNICORE Grid site(s) (*Usite*) that Clients can connect to.
- A Usite offers access to computing or data resources. They are organized as one or several virtual sites (or *Vsites*) which can represent the execution and/or storage systems at the computer centres. In the client the user selects the Vsites to which submit UNICORE jobs or on which sub-jobs will execute.

The software architecture of a UNICORE system comprising two UNICORE sites with a total of three Vsites is shown in Figure 1.

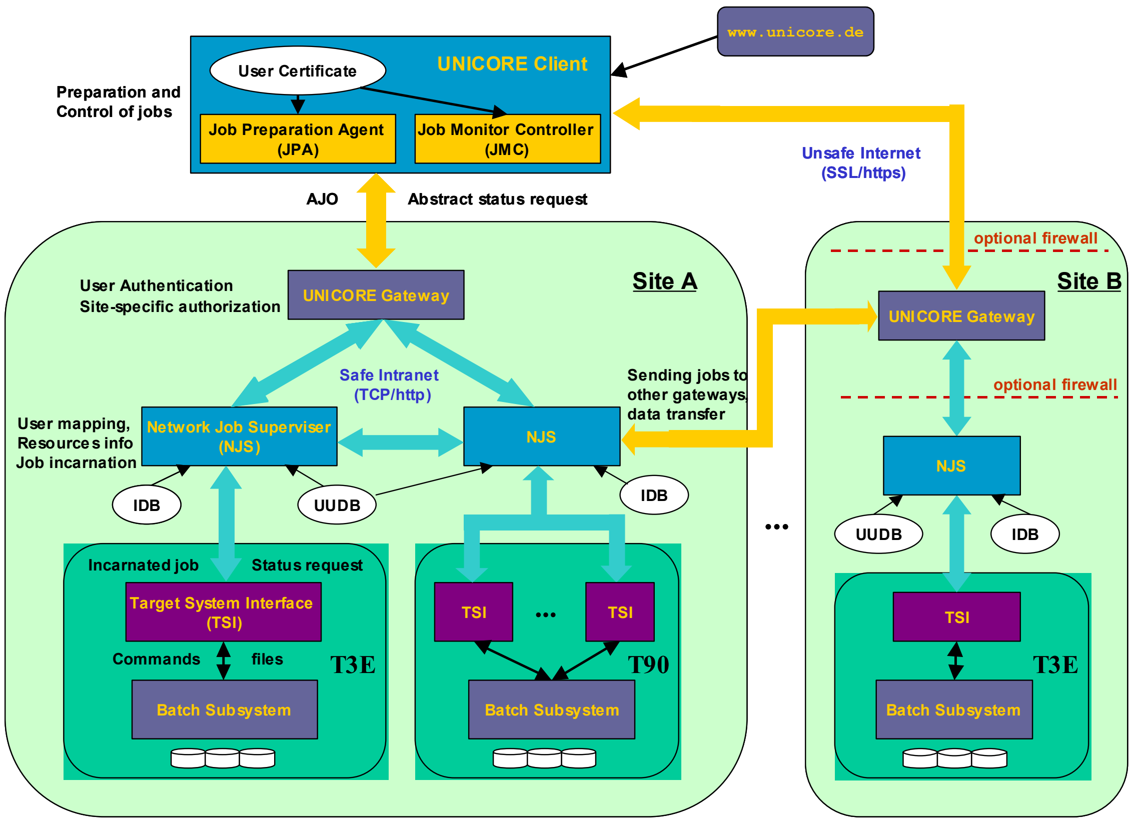

Figure 1: The UNICORE Architecture

The UNICORE components have the following functions:

- The UNICORE *Client* runs on a Java-enabled user workstation or PC, possibly somewhere on the insecure Internet. It is the sole interface to the end-user, who uses it to connect to a UNICORE Gateway. A list of available UNICORE Gateways is maintained as an XML document. The jobs or status requests and the results are formulated in an abstract form using the *Abstract Job Object (AJO)* Java classes.
- The UNICORE *Gateway* is the single entry point for all UNICORE connections into a Usite. It provides an Internet address and a port accessible from the outside for SSL connections. The UNICORE clients will connect to that known port, and use SSL for the UNICORE protocol. The gateway is the point of user authentication, which is the positive identification of a client connection as coming from a UNICORE user. A Gateway can be installed inside or outside of a firewall (also in a demilitarized zone) depending on the site's security requirements.
- A UNICORE Vsite is established by two components: the *Network Job Supervisor (NJS)* and a *Target System Interface (TSI)*. The UNICORE *NJS Server* manages all submitted UNICORE jobs. It performs the user authorization by looking for a mapping of the user certificate to a valid login in the *UNICORE User Data Base (UUDB)*. The NJS incarnates jobs from the abstract AJO definition into the appropriate concrete command sequence for a given target execution system, and hands the incarnated tasks and jobs over to the TSI. The incarnation is based on the specifications in the *Incarnation Data Base (IDB)*.

It is the NJS's task to manage the dependencies between job components and to schedule the components accordingly. The NJS server stores the job status and results, and replies to status and result requests from the client.

In case of sub-jobs which are specified to run on a Vsite at a different Usite, the NJS assumes the role of a Client and submits the sub-job to the remote Gateway (e.g. from site A to site B). Because of the SSL connection this means that a certificate for the NJS itself is also required. The status and results of sub-jobs are gathered in the NJS of the main job.

- Finally, the UNICORE *Target System Interface (TSI)* accepts incarnated job components from the NJS, and passes them to the local batch systems for execution. In addition, file import and export tasks are handled by the TSI, and it also implements low–level status reporting and control of batch jobs.

1.3 Extensions to the UNICORE architecture in project GRIP

The goal of project GRIP is to enable UNICORE users to access Globus resources[3] while preserving the seamlessness of UNICORE. This has been realized without changing the overall architecture of UNICORE nor the security model. The approach chosen to achieve this is described in [5]. As a consequence a Globus controlled resource[4] resides in the UNICORE architecture in Figure 1 at the batch subsystem level. This implies that three main components had to be developed:

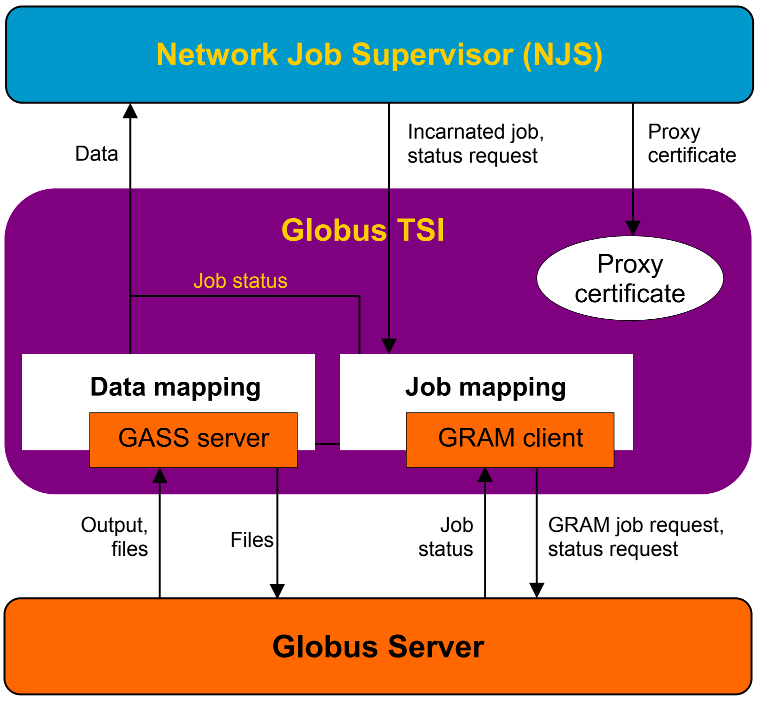

Figure 2: Globus TSI

[3] The Globus architecture is not covered here except for components relevant to the UNICORE – Globus interoperability. The entry point for in-depth information is [3].
[4] In terms of GRIP this is analogous to a Globus Gatekeeper and a MDS providing resource information.

- An extension to the UNICORE client to create a temporary proxy certificate [6] which is required by the *Grid Security Infrastructure* (*GSI*) used by Globus. Since the UNICORE user's permanent certificate is stored in a keystore in the client, the proxy certificate is created there making use of UNICORE's plugin mechanism (see [7] for information on the UNICORE plugin technique). Section 9 covers certification issues including the *Proxy Certificate Plugin* in detail.
- A specific TSI, named Globus TSI, to access the Globus resource. The architecture of the Globus TSI and its integration into the UNICORE architecture is shown in Figure 2.
- An information provider capable of interpreting Globus *MDS* information (*Monitoring and Discovery Service [8]*) and mapping them to UNICORE specific resource information. An interoperable resource broker providing these capabilities is developed in GRIP. It is work in progress and has not yet been tested in production (see also Section 12).

The usage scenario in GRIP does not differ from the one described in Section 1.2 for a pure UNICORE Grid. This meets the objective to provide access to Globus transparently to the user and resources without changing the architecture of UNICORE[5]:

The user creates a job choosing a Globus site as a target. A proxy certificate is created automatically by the Proxy Certificate Plugin and sent to the UNICORE server via SSL as part of the Abstract Job Object. The server verifies the signed job and unpacks the proxy certificate, passing it to the TSI for interaction with the GSI. This is done before the UNICORE job constructs are mapped to the appropriate Globus *RSL* (*Resource Specification Language [9]*) description and submitted to the Globus Gatekeeper. Pre- and post staging of files including standard output and error files is implemented integrating the Globus *GASS* (*Global Access to Secondary Storage* [10]) server component into the Globus TSI. In addition job monitor requests and results are mapped between the corresponding UNICORE and Globus descriptions.

The first version of the Globus TSI has been implemented in Perl making use of the TSI template which is part of the UNICORE distribution. GRIP also implemented a Java version of the TSI which serves as a basis to integrate OGSI[6] compliant services into future software versions of UNICORE.

[5] Please be aware that extensions to the UNICORE architecture have been developed exploiting APIs, the plugin mechanisms, the IDB and the TSI template. This means customizing, not changing, the architecture.
[6] The Open Grid Services Infrastructure addresses the creation, management and information exchange between services operating in an OGSA environment. [11] is the GGF draft specifying this infrastructure.

The following chapters cover the topics relevant to Grid Production Systems as proposed by the Production Grid Management Research Group of GGF for their Use Cases (see Appendix A for the use case template).

2 Supported Grid Computing Model

In [12] and [13] Foster and Kesselman define the following Grid computing models:

- Distributed Supercomputing
- High Throughput Computing
- On-Demand Computing
- Data-Intensive Computing
- Collaborative Computing
- Desktop Supercomputing

UNICORE and its extensions in project GRIP most closely match the Distributed Supercomputing model in terms of the Grid's purpose and architecture. The computing resources in the UNICORE Grid are of cause not limited to supercomputers; any managed system may be made available to the user of the Grid. All distributed resources, especially supercomputers from most major vendors, are easily accessible via the users desktop. The user creates a job once in the client and may submit it for execution to different systems at different participating sites without having to acquire in-depth knowledge of system configuration, naming conventions, or management policies at the different sites. Complex jobs may be constructed consisting of interdependent tasks which exploit special characteristics (vector processors, MPP system, clusters of SMPs) of the participating systems. Typically, the different systems are located at different, administratively independent sites.

The UNICORE workflow model has been adopted by GRIP and is extended to include Globus resources. The user creates a job which may contain sub-jobs to be executed on different platforms at different sites. By definition all sub-jobs are executed independently. The user may specify dependencies between parts of the job. In this case the NJS ensures that a successor is started only if its predecessor completed successfully. Flow control constructs support conditional execution (it-then-else), loops, or conditional holding of the job flow. A full description of the UNICORE work flow can be found in [1].

- Currently GRIP does not support the management of coupled process involving a co-scheduler. But in connection with the project's scheduling and resource management activities the integration of a co-scheduler will enable this especially in an OGSI-compliant Grid.

To support MPI applications on distributed systems PACX MPI [14], developed at the High Performance Computing Center Stuttgart (HLRS), Germany, has been integrated into UNICORE as a plugin. This prototype is presently not integrated in the UNICORE/GRIP production version because the required co-scheduling functionality is not available at the batch environments at the partner sites with the exception of *CCS*, the *Computing Center Software* developed at PC^2 [15] at the University of Paderborn, Germany. CCS is a topology based resource management for networked high-performance computers with built-in advance reservation features. The feasibility of co-scheduling in UNICORE has been demonstrated at Paderborn using CCS.

3 Grid Data Model

GRIP adopts the UNICORE data model. For each a job a dedicated, temporary user file space (*Uspace*) is created automatically for the duration of the job. The location of Uspace is at the discretion of the site; for performance reasons it typically resides in a file system on the target system. At the end of a job the Uspace is automatically removed.

File systems to which the user has access on the target system are collectively called *Xspace* (for UniX space). Users typically import files from Xspace into Uspace as part of a job and export the results from Uspace to Xspace. This import/export function may be included in the job as an explicit import or export task, it may be an implicit import/export task as part of a compute task. The import/export tasks can also be automatically generated by application plugin that encapsulated the knowledge which files are needed by the application and which results are created and have to be saved.

A file that has been created in one sub-job in an Uspace and that is required in a successor sub-job at a potentially different site (and therefore a different Uspace) must be specified in a transfer task. UNICORE will use an appropriate file transfer mechanism to copy the data between Uspaces as described in Section 11.

The user has the option to transfer data from the client workstation into an Uspace and to save results created during the job to the client for analysis or post-processing. Workstation data is imported into the Uspace at submission time. Thereafter the UNICORE job may execute independently of a connection to the workstation. Results stored in Uspace may be exported to the client as part of the job description. The user has to initiate the transfer once the job is completed. Application plugins initiate the transfer automatically, even while the job is running, for example, to show the progress of a simulation graphically. Consolidated stdout or stderr files, consisting of the individual stdout/stderr files created at each target system may be inspected at the client workstation. Files to be processed after job completion are saved in a special spool directory by NJS. This is erased when the user removes the UNICORE job from the Grid explicitly or when a system administrator initiated clean-up takes place.

In the UNICORE Plus project access to tertiary storage systems, HPSS and TSM, was evaluated and prototyped. These functions have not been integrated into the production system.

4 Collaboration Structure

UNICORE sites are administratively independent. This was one of the key design decisions. The participating sites reserve the right to define which of theirs resources are made available to a particular Grid and to decide who may use these resources. In other words, UNICORE sites can participate in different virtual organizations. UNICORE must be flexible enough to adapt to existing proven practices at the participating centres. Until new practices are proven superior to existing ones, sites will not change, for example, their accounting systems or naming conventions. In addition, decisions on resource allocations by funding agencies must be respected. Presently, resources to be used in projects amongst the partners are shared by agreement within predefined limits. Production users of the UNICORE HPC Grid in Germany use their existing allocations that they obtained individually at each of the centres. Once the political and administrative hurdles have been overcome, users will have to apply only for Grid resources, not for system resources.

UNICORE, Globus and so GRIP deal with existing trust models. Both UNICORE and Globus use a public key infrastructure (PKI) and X.509V3 certificates.

UNICORE and consequently GRIP use certificates to authenticate users, servers, and software, especially plugins. A UNICORE/GRIP client must present a valid certificate to the Gateway from an accepted CA before it is given information about the resources available in the Grid. A particular resource at a Vsite may only be used if the certificate is found in the UUDB which maps certificates to UNIX login names. This guarantees full control over the site's resources. Other existing restrictions may apply on the target system, independent of UNICORE or GRIP. For example, certain jobs may execute only at certain times, depending on the resource management policy at that specific site.

When the UNICORE project was started the assumption was made that certificates would be ubiquitous and that each user would use them as part of a normal identification process. Therefore a very strict CA policy [16] was selected based on the DFN PCA which verified the identity of the certificate holder and thus could be used to authorize the user. Due to the lack of a global PKI infrastructure this approach will not scale in production with hundreds of locally distributed users. UNICORE is technically capable to handle multiple certificates issued by different CAs. The Germany HPC centres agreed to mutually accept their certificates.

The European Grid project EUROGRID [17], which uses UNICORE as the Grid software, decided not to follow the strict UNICORE CA policy. Instead an EUROGRID CA has been established which issues X.509 certificates for both projects EUROGRID and GRIP.

To use Globus resources, a GRIP user needs a temporary proxy certificate generated from the user certificate within the UNICORE client. By default Globus is configured to accept proxy certificates generated from a user certificate signed by the Globus CA, but it can be configured to accept the EUROGRID CA or any other CA as a trusted CA. (see Section 9 for a in-depth description).

5 Type of Grid

UNICORE and its extension to Globus resources through GRIP support a wide range of UNIX platforms. They include supercomputers from Cray, Fujitsu, IBM, Hitachi, NEC, and Sun. The architectures span vector processors, massively parallel systems, clusters of SMPs, workstation clusters, and individual scalar systems.

The TSI interfaces between UNICORE and the target system's operating system or batch sub-system. Presently supported are the vendor specific UNIX derivatives and various dialects of NQS, LoadLeveler, LSF, PBS, and CCS. To support a new environment requires adapting the IDB and a TSI template (in Perl or Java) to the appropriate operating system or batch system interfaces.

UNICORE assumes that the target systems are properly managed and that the servers running the UNICORE Gateway and NJS are securely configured. A user administration must map the certificates to target system specific user IDs in a controlled fashion.

The GRIP client which is an extension to the UNICORE client by adding a proxy certificate plugin is system independent. It is a Java application and requires a Java runtime environment at an appropriate level (currently 1.4). It has been tested on Windows PCs and UNIX/Linux workstations; it also runs on Apple Macintosh operating systems.

GRIP adds the possibility to access resources controlled by the Globus Toolkit, which is another target system according to the design of the GRIP architecture as shown in Section 1.3. Those resources are potentially administrative separate systems maybe even in a different virtual organisation. Further information on Globus target systems and supported batch systems is provided at [3].

6 Users of the Grid

6.1 Application end users

UNICORE's main focus is to give end users consistent and intuitive access to a wide range of resources in a Grid. GRIP extends the range of accessible systems even further by including Globus resources. The user can construct complex jobs with the aid of the client GUI. The client creates an abstract description of the job and stores it in XML or AJO format. The user may select the system the job or parts of the job should execute on and submits the job. Prior to submission UNICORE automatically checks if the requested resources, for example, number of processors, memory size, disk space, but also software packages are available at the selected system. If not, the user is informed and may select a different system or change the requirements. The server components of UNICORE translate the abstract description into system specific commands[7]. These standard features of UNICORE require only two mouse clicks to make a job run on a different system at a different site. The manual selection of execution systems is consistent with today's way users access HPC resources. The resource broker development in GRIP will support the user in finding systems in a Grid which will run the application according to user selected criteria.

The user has full control over the job. She may check the status, display the results, cancel the execution of the job, or remove the job from the system.

In addition, the UNICORE client supports application specific graphical interfaces which assist the users in preparing jobs in terms of the application. The user specifies problem specific parameters in the GUI, for example initial conditions for a simulation. The plugin constructs the necessary job description, including data imports, exports, or transfers. For complex applications like *CPMD* (*CarParrinello Molecular* Dynamics) a wizard can be provided (see [18]) that guides the user through steps necessary to specify the domain specific values. The wizard can perform syntactic and semantic checks thus allowing the user to focus on the scientific problem. Projects UNICORE, EUROGRID and GRIP created plugins for biomolecular applications like CPMD, Gaussian98, and GAMESS and for the structure mechanic codes MSC.Nastran, Fluent, and STAR-CD. Application specific support, especially for CPMD and Gaussian98 attracted users to use UNICORE even if they use mostly the same target system.

6.2 Application developers

Two techniques are available to application developers to integrate an application into a UNICORE/GRIP environment. The first is to write a plugin, an optionally wizard to offer an application specific GUI to the end-user. Applications that have no graphical user interface can be made very attractive and easy to use, especially for novices. Existing GUIs can be integrated as in the case of MSC.Nastran and Fluent. This retains the known look-and-feel for the users and adds the seamlessness and security of UNICORE to the application. Writing a plugin in Java can be a simple task of a few days; creating an elaborate wizard may take a few months. Plugins are also capable of monitoring running jobs and displaying the outcome of simulations graphically. In addition to the plugin code, the application developer has to provide an incarnation template to translate the abstract software resource into UNIX commands. This incarnation template may have to be customized at each site to accommodate local conventions.

The second technique has been developed in project GRIP. Wrappers have been created that allow executing applications which can not be modified on the source code level to run both in a UNICORE and in a Globus environment. This is an important interim step towards Grid interoperability. It may become obsolete once the wrapped applications become natively Grid enabled. Examples of this approach can be found in [19].

[7]In case of Globus targets the abstract job description is translated into RSL.

6.3 Grid Software developers

The plug-in technique can also be used to extend the base functions of UNICORE. GRIP uses the technique to create Globus proxy certificates. Interactive access, a major extension to UNICORE, was implemented through plugins in project EUROGRID. Timer driven monitoring of UNICORE jobs is a standard UNICORE plugin.

7 Mechanism to Promote Cooperation and Mutual Technical Support

In addition to regular project meetings between the partners, GRIP uses the following mechanism to promote cooperation and mutual technical support.

In particular all GRIP developers have access to a central server which runs software supporting professional document management, discussion lists, and project administration called *BSCW* (*Basic Support for Cooperative Work*). New software versions, publications, talks, general questions, etc. are archived on this server.

All GRIP partners have access to a problem tracking system called GNATS. This tool is document and submit GRIP software problems to Pallas, the software development company that maintains UNICORE.

Further technical support is given by the UNICORE administrators at partner sites.

8 Deployment of Grid Software

The UNICORE software consists of several components:

- Client,
- Gateway,
- Network Job Supervisor (NJS),
- Target System Interface (TSI),
- basic Plugins, and
- optional Plugins.

The first five components are bundled for a complete install at new sites or as a complete replacement for an existing version. All components can be downloaded individually from the UNICORE Forum website [20].

Individual components can be replaced by new versions without affecting the rest of the system. It is especially important to decouple upgrades of the client software (Client and Plugins) from the server software (Gateway, NJS, TSI).

To enhance UNICORE, especially to introduce new features, the following procedure has proven to work very well during the development phase when frequent updates were the norm.

As a first step, a new version of NJS and/or the Gateway was installed in parallel to the existing operational system. The new software could be accessed from existing clients by specifying a different port number. These tests were typically carried out at one of the partner sites. When the new server components were production ready, they replaced the old ones. All users could continue to work with UNICORE, however, without having access to the new features. The users could subsequently download the client versions at their own pace and exploit the new functions.

This approach allowed also implementing new client functions incrementally in subsequent releases of the client. Of cause bug fixes could be delivered to the clients in a very timely manner. The same holds true for the optional plugins.

TSIs need to be updated only to fix problems during the development phase or to adapt to new versions of the operating system or the batch sub-system. This process is completely decoupled from the UNICORE deployment process.

In the standard distribution the replacement of a UNICORE client is done manually by the user. It involves simply to download an archive file (tar or zip), unpack it, and copy the new files into the appropriate directory on the client workstation. Optionally, the user may use the Java Web Start package [21] from Sun Microsystems to automate this process.

To allow testing of UNICORE without having to install a complete system, the UNICORE Plus project developed a public UNICORE test bed [22] to allow interested users to try out the UNICORE software and functionality. In this testbed several virtual UNICORE sites are simulated on a dedicated system. A simple install process creates an operational client and provides the user with an account and the necessary certificate in the UNICORE test Grid.

The GRIP project introduced the additional complexity to install and maintain an operational Globus system at a well defined level to ensure that the Globus TSI interfaces correctly with the underlying Globus environment.

During the UNICORE and GRIP development period the developers recognized many important topics which are essential for production systems:

It is absolutely essential to retain compatibility between different software versions for the users. A job created and saved in version N of UNICORE has to work in version N+1. During the development phase with only a small number of mostly internal users the project decided twice to break the compatibility to allow a rapid change of the UNICORE protocols and the Abstract Job Object structure. The alternative would have been to invest development resources into compatibility modules without major additional benefits. The compatibility problem is now solved, since UNICORE stores jobs both in the internal AJO format and as XML text. The client is now always able to interpret one of the formats from a previous version.

Incompatibilities between different Java versions can be very painful to the user. It can not always be guaranteed that code written for a newer version of Java will execute properly in each implementation of a Java Run Time Environment. This forces users to upgrade their client to the latest version of Java. Basing the Client on an old version of Java is also no remedy since the user or the installation may decide for other valid reasons to upgrade the Java RTE on the client platform.

9 Certificate Practices

This chapter gives an overview of GRIP security solutions. The GRIP security environment is a combination of models and practices from both the UNICORE and the Globus security. To get a detailed insight into both security solutions refer to [23].

9.1 Certificate Handling

Both UNICORE and Globus use security mechanisms to give users secure access to remote resources. Although both security architectures are based on standard public key technology using X.509 certificates, the authentication mechanisms differ fundamentally. UNICORE signs each part of the job with the user’s private key. NJS can verify the integrity of jobs and can map the certificate to the correct userid at the target system. The same is true for other requests send to a server from the client. It is not possible for a malicious user to fake someone’s identity. The UNICORE user’s private key is encrypted and stored in a keystore locally on the user’s machine. It should never be send to another system over the network; in fact in GRIP it need not leave the user's workstation. The user's public key has to be made available to site administrators to be integrated into the UUDB (see Section 1.2).

In contrast, Globus uses proxy certificates to delegate rights. GRIP has to handle both techniques transparently to the user for job submission, job monitoring, and file transfers, for example via GridFTP.

A special plugin has been developed in GRIP as an extension to the UNICORE client that generates a proxy certificate from the GRIP user certificate. The user may use the plugin's graphical interface to adjust characteristics of the proxy certificate, e.g. the expiration time of the proxy. GRIP users do not have to have a certificate signed by the Globus CA to submit jobs from UNICORE to Globus. The prerequisite is that the Globus site trusts the CA which signed the GRIP user certificate.

The GRIP proxy certificate consists of:

- a newly created temporary private key, and
- a temporary X.509 certificate which is signed with the users original private key and which contains the newly created public key, and the user's original public key.

The proxy certificate is transferred via SSL to the NJS as part of the AJO. The Globus TSI acts as a Globus client. The proxy certificate is stored in the user's file space which is only accessible by the user and which is removed after the job has finished. The Globus TSI can access it whenever the proxy certificate is needed for SSL handshake in Globus GSI.

To guarantee secure transfer of the proxy certificate from the NJS to the Globus TSI, GRIP implements this link as an authenticated, private SSL connection. This process is consistent with the handling of proxy certificates in Globus.

In GRIP the UNICORE server components, Gateway, NJS and TSI are provided with a server certificate.

9.2 Experiences with different CAs

The UNICORE and the GRIP projects implemented different policies to issue X.509 certificates. The UNICORE certification policy was based on the very strict policy published by the German Research Network DFN (DFN – PCA). To obtain a UNICORE certificate the user had to appear in person at a Registration Authority (RA) and prove his or her identity and sign a certification request. The RA had to check the correctness of the information in the certificate, like name and organization. A document signed by the user and the RA had to be mailed to the CA by postal mail. A certificate would only be issued upon receipt of this document. The typical turn-around time would be one week. Although certificates issued according to the DFN-PCA policy guarantee the identity of the certificate holder, this certification practice has turned out to be counterproductive because of the long process time. In addition, it requires that Registration Authorities are in place where ever potential customers are. The German HPC centres consequently agreed to relax the CA policy. Each centre may issue certificates and use established process which have been in place and proven adequate to issue user accounts at the centres. This implies typically a fax, signed by the user and the user's department head.

GRIP uses the EUROGRID CA established for a Europe-wide Grid. The EUROGRID project uses a weaker security policy. Users who want to receive a EUROGRID certificate do not have to appear at the RA in person and prove their identity. The local administrators at the participating centre will verify the CA request and confirm this to the CA via email. Since authentication and authorization are decoupled by definition, this process proved to be adequate for project partners to protect their resources.

A production environment requires solutions to the following issues:

1. CA policies that are accepted by all participants of a Grid.
2. A reliable and efficient Public Key Infrastructure (PKI).
3. Software that can handle multiple CAs.

UNICORE and GRIP have proven that the third point can be solved. The first two are non-technical and have to be solved differently.

10 Management of Firewall Issues

UNICORE and its extensions in GRIP have been designed to work with firewalls. Only one port has to be open allowing SSL connections to the Gateway. In production UNICORE Gateways are often running on dedicated machines only opening that single port as an entry point for UNICORE client connections (see also Figure 1).The port number can be selected by the site. The list of open ports is published among the participating centres within a Grid. When the client is started the current list of participating sites is loaded into the client. Alternatively, the user may import the list of gateways and matching ports into the client and use a local copy. The local copy may contain only a subset of all Usites or it may contain additional Gateways for test purposes. The use of multiple site lists is also supported. The access to Globus resources within the firewall from a UNICORE client requires no additional modifications to the local firewall policies.

To verify that UNICORE poses no security hazard to the installation, the security administrators may inspect the Java source of the Gateway code.

At one time, one of the partners in the UNICORE project allowed only network connections to previously defined IP addresses. This firewall policy made Grid computing practically impossible since each new user would have to be added to management list in the firewall.

The Globus TSI allows two different modes of operation:

1. The Globus server installation is local to the Globus TSI, which means that the Globus home directory and the UNICORE Uspace are identical.
2. The Globus server installation is remote to the Globus TSI, potentially with a firewall between the Globus client integrated in the Globus TSI and the Globus server installation.

In the first case, no interoperability problems occur.

In the second case one has to cope with the firewall issues characteristic for Globus as described in [24]. To prevent the UNICORE security model from being weakened, in this case the Globus TSI uses only Globus commands which do not require opening ports for incoming connections.

11 Data Transport

The UNICORE internal file transfer mechanism uses a java zip stream and is based on *UPL* (*UNICORE Protocol Layer*) which is implemented on top of SSL. Depending on the data source and sink, two different data transport paths are possible:

1. From the UNICORE client side to the user's Uspace at a target system and vice versa. In UNICORE terms this is called an import task (to the Uspace) or an export task (from the Uspace).
2. From the Uspace at target system A to the (remote) Uspace at target system B. This is a transfer task.

Section 3 outlines the integration of the data transport mechanisms into GRIP's data model

In case of GRIP there are no enhancements necessary if Uspace and Globus home directory are identical. Assuming that the Globus server is remote to the Globus TSI, files have to be transported to the remote Globus side using the appropriate Globus mechanisms.

As part of the EUROGRID project, the UNICORE architecture has been extended to use alternative file transfer (*AFT*) mechanisms, other than the UNICORE internal one described above. The NJS was extended by an AFT module and a corresponding API to handle any desired alternative file transfer protocol. One implementation of this interface uses GridFTP [25] as an AFT mechanism (see [26]). Compared to the UPL data transport, where data is always transferred via Gateway and NJS to the target system (see also Figure 1), the GridFTP AFT transfers files directly from one target system to another. The GridFTP client residing on the TSI level uses the proxy certificate described in Section 5 and Section 12 to contact the remote GridFTP server which uses the TSI server certificate for GSI enabled authentication and encryption.

Since all data transport actions are treated like normal tasks in the UNICORE, they are fully supported by UNICORE's workflow engine. Currently implemented data transport tasks are: import task, export task and transfer task, described above and in Section 3.

12 Job Information Tools

GRIP uses the UNICORE functions to obtain information about a submitted job. From within the UNICORE client users can request an updated job status. This information is available at three levels: At the overview level, colour coded icons show the status of the jobs, sub-jobs, or individual tasks. The colours indicate, for example, queued within UNICORE, waiting to be executed, executing, completed successfully, terminated abnormally. In addition, detailed information, giving for example the reason for a failure, or the iteration count for a task repeated in a loop, is available on request. At the lowest level all information that the target system typically places in stdout or stderr is at the user's disposal. To receive this information, the user must present her valid certificate.

To prepare a job the user can inspect the resource information for all target systems in the Grid he or she is authorized to use. This includes capacity resources, like number of nodes and processors, available memory size and available disk space, maximum CPU time that may be requested; It also includes capability resources, like installed application including their version.

This information is maintained individually at each participating site in the Grid. No central administration is required. Whenever the hardware or software configuration changes, it is the responsibility of the administrators to update the resource description in the incarnation database (IDB). The updated information is made available to the client during the next connection to the server or upon an explicit refresh request by the user.

This information is also available to the resource broker that is being developed as part of project GRIP. For the time being the translation from the Globus MDS to the UNICORE IDB is not automatic. GRIP is working on an ontology based process.

13 Help Desk Systems Used?

Please refer to Section 7.

14 Tools to Track Operational Problems

The administrators have a range of UNICORE specific command line administration tools at their disposal to check the functionality of the servers and the status of the different TSIs on the target systems. These tools allow obtaining status information about any submitted job, to remove a job on behalf of the user, and to clean up in case of errors.

The UNICORE server components, Gateway, NJS and TSI, produce detailed logging information about all submitted jobs. This information helps the UNICORE administrators to find the reason for problems and assists in solving them.

Further more, the GNATS software which has been described in paragraph 7, is used to request problem resolution from the developers.

15 Failure Recovery without User Interaction

UNICORE has been designed minimize the impact of system problems for the user and to inform the user if components of the Grid are unavailable.

UNICORE components, like the Gateway, NJS, or TSIs are monitored internally. If they fail they are restarted automatically. Also if the systems the run on fail, they are started once the systems comes up. For all these failures of the Grid, no user interaction is required. It might require the intervention of an administrator to resolve the cause of the error.

If the user tries to access a system that is temporarily not available an error message is shown at the client. In practice, a UNICORE/GRIP based Grid is as available as the participating target systems are.

16 Security Considerations

Security is a crucial to the success of a Grid Production environment as the one described in this paper. Therefore the issues concerning the security environment established in GRIP are discussed in detail in the following sections:

- The integration of security solutions into the UNICORE and GRIP architecture is described in the introductory Section 1.
- In context with the description of the collaboration structure the CA and PKI are inspected in Section 4.
- Section 9 illustrates the project’s certificate practices and the experiences made using multiple CAs.
- The firewall requirements are covered in Section 10.

Appendix A: Use case template

The following set of questions has been developed by the GGF PGM-RG to allow comparing different use cases. They have been addressed in this document if applicable.

Which Grid Computing model does your Grid support?

- Loosely coupled processes: types?
- Coupled processes: co-scheduling needed
- Tightly coupled processes: co-scheduling needed, firewall requirements

Which Grid data model do you use?

- Only user data in user's file system
- User's file system + grid scratch file systems
- Occasional access to tertiary storage systems
- Distributed analysis of massive datasets (SRB)
- Large reference datasets (caching)

How does your collaboration structure look like?

- Existing trust model, within an organisation, similar systems
- Administrative diverse systems within one organisation (What is divers?)
- Administrative heterogeneous model (e.g. science labs and industry)

Which Type of Grid?

- Peer to peer
- Administrated separate systems (workstations, clusters, HPC systems)

Type of users in your Grid?

- Application developers
- Applications end users
- Grid software developers (omit them in a production Grid)
- External customers

Which mechanisms are established to promote cooperation and mutual technical support?

- Compare GUS WG

How do you manage the deployment of Grid software?

- Testing procedures (what, codes)
- Testbed implementation (architectures, size, benchmarks)
- Steps to production (roll out procedure, on site testing, system release)

Certificate practices?

- Type of identification verification
- Key protection
- Entities that need certification
- Testing procedures
- Installation

How do you manage co-scheduling?

- Not available
- Type of scheduler, batch system
- Own developments

How do you manage firewall issues?

- Open port control
- Software modifications
- Interoperability problems

How do you handle data transport?

- Separate GridFTP systems
- Global files systems (which one)
- Support by workflow models
- Is up to the user

What information does your user get about his job/account?

Which help desk system do you use?

Which tools do you user to track operational problems?

From which failures can you Grid recover without user interaction?

Author Information

Dietmar Erwin, Michael Rambadt, Philipp Wieder
Research Centre Jülich
Central Institute for Applied Mathematics
Forschungszentrum Jülich GmbH
52425 Jülich
Germany
{d.erwin, m.rambadt, ph.wieder}@fz-juelich.de

Glossary

AFT	Alternative File Transfer
AJO	Abstract Job Object
API	Application Programmer's Interface
BSCW	Basic Support for Cooperative Work
CCS	Computing Center Software
CPMD	CarParrinello Molecular Dynamics
DFN	Deutsches Forschungsnetz (German Research Network)
GRAM	Globus Resource Allocation Manager
GRIP	Grid Interoperability Project
GSI	Grid Security Infrastructure
HPSS	High Performance Storage System
IDB	Incarnation Database
PKI	Public Key Infrastructure
NJS	Network Job Supervisor
LSF	Load Sharing Facility
MDS	Monitoring and Discovery Service
MPP	Massively Parallel Processing
NQS	Network Queuing System
PBS	Portable Batch System
RA	Registration Authority
RSL	Resource Specification Language
SMP	Symetric Multiprocessing
SSL	Secure Socket Layer
TSI	Target System Interface
TSM	Tivoli Storage Manager
UNICORE	Uniform Interface to Computing Resources
UPL	UNICORE Protocol Layer
UUDB	UNICORE User Database

XML	Extensible Markup Language

References

[1] D. Erwin (ed.), “UNICORE Plus Final Report – Uniform Interface to Computing Resources”, *UNICORE Forum e.V.*, ISBN 3-00-011592-7, 2003. `http://www.unicore.org`.

[2] “The Grid Interoperability Project.” `http://www.grid-interoperability.org`.

[3] “The Globus Project.” `http://www.globus.org`.

[4] I. Foster, C. Kesselman, J. Nick, and S. Tuecke, "The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration", *Global Grid Forum*, Open Grid Service Infrastructure WG Draft, Version 2.9, 2002.

[5] D. Snelling, S. van den Berghe, G. von Laszewski, Ph. Wieder, D. Breuer, J. MacLaren, D. Nicole, and H.-Ch. Hoppe, "A UNICORE Globus Interoperability Layer", *Computing and Informatics*, Vol. 21, pp. 399 – 411, 2002.

[6] S. Tuecke, D. Engert, I. Foster, V. Welch, M. Thompson, L. Pearlman, and C. Kesselman, "Internet X.509 Public Key Infrastructure Proxy Certificate Profile", *Internet Engineering Task Force*, Public-Key Infrastructure (X.509) WG Draft, `http://www.ietf.org/internet-drafts/draft-ietf-pkix-proxy-06.txt`.

[7] M. Romberg, "The UNICORE Grid infrastructure", *Scientific Programming*, 10(2), pp. 149 – 157, 2002.

[8] K. Czajkowski, S. Fitzgerald, I. Foster, and C. Kesselman, "Grid Information Services for Distributed Resource Sharing", in *Proceedings of the Tenth IEEE International Symposium on High-Performance Distributed Computing (HPDC-10)*, IEEE Press, August 2001.

[9] "Globus Resource Allocation Manager (GRAM)." `http://www-unix.globus.org/api/c-globus-2.2/globus_gram_documentation/html/main.html`.

[10] J. Bester, I. Foster, C. Kesselman, J. Tedesco, and S. Tuecke, "GASS: A Data Movement and Access Service for Wide Area Computing Systems", *Sixth Workshop on I/O in Parallel and Distributed Systems*, 1999. `http://www.cs.dartmouth.edu/iopads/papers.html`.

[11] S. Tuecke, K. Czajkowski, I. Foster, J. Frey, S. Graham, C. Kesselman, D. Snelling, and P. Vanderbilt, "Open Grid Services Infrastructure (OGSI)", *Global Grid Forum*, Open Grid Service Infrastructure WG GWD-R, Version 1.0, 2003.

[12] I. Foster, and C. Kesselman (eds.), "The Grid: Blueprint for a New Computing Infrastructure", Morgan Kaufmann, 1999.

[13] I. Foster, and C. Kesselman, "Globus: A Metacomputing Infrastructure Toolkit", *Intl. J. Supercomputer Applications*, 11(2):115-128, 1997

[14] Edgar Gabriel, Michael M. Resch, Thomas Beisel, and Rainer Keller, "*Distributed* Computing in a Heterogeneous Computing Environment", Vassil Alexandrov, Jack Dongarra (eds.),in *Recent Advances in Parallel Virtual Machine and Message Passing Interface*, pp. 180-188, Springer, 1998.

[15] "CCS: Computing Center Software." `http://www.uni-paderborn.de/pc2/projects/ccs/`

[16] E. Bötsch, "UNICORE-CA (U.CA) – Certification Guidelines for UNICORE". 2000. `http://www.fz-juelich.de/unicoreplus/U-CA_policy_eng.pdf`.

[17] "The EUROGRID project." `http://www.eurogrid.org`.

[18] V. Huber, "UNICORE: A Grid Computing Environment for Distributed and Parallel Computing", in *Parallel Computing Technologies*, Proceedings of the 6th PaCT Conference 2001, pp. 258, Springer-Verlag LNCS 2127, 2001.

[19] J. Pytlinski, L. Skorwider, V. Huber, and P. Bala, “UNICORE - An Uniform Platform for Chemistry on the Grid”, *Journal of Computational Methods in Science and Engineering* **2** (3s-4s), pp. 369-376, 2002.

[20] “The UNICORE Forum e.V.” `http://www.unicore.org`.

[21] “Java Web Start.” `http://java.sun.com/products/javawebstart`.

[22] “The UNICORE Test Grid.” `http://www.fz-juelich.de/unicore-test`.

[23] T. Goss-Walter, R. Letz, T. Kentemich, H.-Ch. Hoppe, and Ph. Wieder, “An Analysis of the UNICORE Security Model”, *Global Grid Forum*, Grid Certificate Policy WG GWD-I, 2002.

[24] V. Welch, “Globus Toolkit Firewall Requirements“, Version 03, 2002. `http://www.globus.org/security/v2.0/Globus%20Firewall%20Requirements-0.3.pdf`.

[25] W. Allcock (ed.), “Protocol Extensions to FTP for the Grid”, *Global Grid Forum*, GridFTP WG GWD-R, Revision 3, 2003. `http://www-isd.fnal.gov/gridftp-wg/draft/GridFTPRev3.pdf`.

[26] D. Breuer, D. Mallmann, D. Snelling, and S. van den Berghe, “GridFTP as an Alternative File Transfer Mechanism within the UNICORE Environment”, *GRIDSTART Technical Bulletin*, Volume 2, 2003. `http://www.gridstart.org/download/TechnicalBulletinFeb2003.pdf`.