UNICORE – Globus:
Interoperability of Grid Infrastructures

Michael Rambadt   Philipp Wieder

Central Institute for Applied Mathematics (ZAM)
Research Centre Juelich
D–52425 Juelich, Germany
Phone: +49 2461 612057
E-mail: [m.rambadt|ph.wieder]@fz-juelich.de

ABSTRACT:

This paper describes software developed at the Research Centre Juelich to demonstrate the feasibility of Grid interoperability between UNICORE (Uniform Interface to Computer Resources) [1] and Globus [2] without changes to any of the systems. UNICORE user requests, like job submission, status query, and output retrieval had to be mapped to the corresponding Globus mechanisms. Another significant topic was the integration of the Globus security infrastructure into the UNICORE architecture. The functionality of this prototype has been demonstrated successfully on SC2001 in Denver.

KEYWORDS:

Globus, Grid, UNICORE, Interoperability

1   Motivation

There has been a substantial progress in developing Grid technologies in the recent years [3]. At universities and research centers world-wide scientists work on the evolution of Grid computing. Even if the way differs in many cases one of the principal goals of all those projects is the same: to give users access to distributed resources. Different projects focus on different aspects and it is only natural to combine them.

Both Globus and UNICORE provide a Grid infrastructure which gives users access to distributed resources. Globus can be characterized as a toolkit that allows the development of Grid applications using the rich set of Globus services. UNICORE represents a vertically integrated solution focusing on uniform access to distributed computing resources. UNICORE is developed by a consortium of German universities, research laboratories, and software companies. It is funded in part by the German Ministry for Education and Research (BMBF).

The work presented here has been carried out in cooperation between Argonne National Laboratory and Research Centre Juelich. UNICORE acts as a client to access selected Globus resources, allowing job submission, status queries, data staging, and output retrieval. In addition, the objective was to achieve the goal without changes to Globus and UNICORE.

2   UNICORE & Globus architectural concepts

We restrict the description of the two architectures to the core concepts related to our work. Especially components

needed to manage multi-site jobs – like co-allocation used by Globus or job dependencies and data staging in UNI-CORE – are not covered. For a detailed description of the infrastructures of UNICORE and Globus see [4] and [5] respectively.

2.1 UNICORE

UNICORE implements a three tier architecture consisting of user, server and target system tier.

As shown in Figure 1 the user tier consists of the UNICORE client, a graphical user interface which enables users to prepare and manage UNICORE jobs. The client is a Java application that executes on the user's personal workstation. A UNICORE job is created using the *Job Preparation Agent (JPA)*, where the user specifies the actions to be performed, the resources needed and the system on which

the job is to run. From this job description the UNICORE client generates an *Abstract Job Object (AJO)*, which instantiates the class representing UNICORE's abstract job model. The AJO is signed with the user's certificate and sent to the *Gateway*, one of the two components of the server tier.

The Gateway authenticates the user and transfers the AJO to the *Network Job Supervisor (NJS)*. The NJS translates the abstract job represented by the AJO into a target system specific batch job using the *Incarnation Database (IDB)*. The Gateway and NJS execute typically on dedicated secure systems behind a firewall.

UNICORE's communication endpoint is the *Target System Interface (TSI)*, which is a daemon executing on the target system. It's role is to interface with the local operating system and the local batch subsystem.
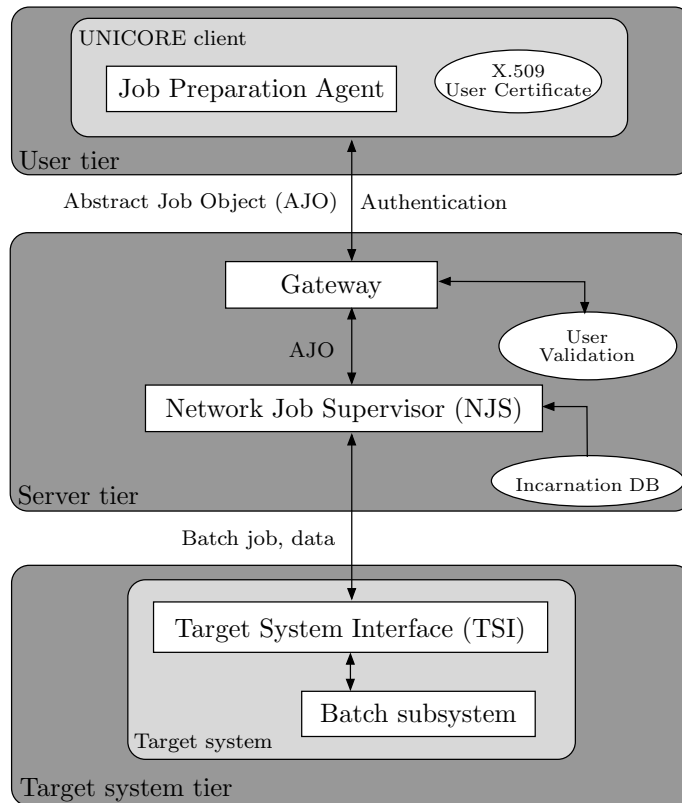

Fig. 1: UNICORE architecture

2.2 Globus

Figure 2 shows the components of the Globus architecture relevant to our work, namely security components, job submission, status information and data transfer. Especially

the information services and the multi-site job mechanisms available in the Globus Toolkit (see [5]) are not relevant to the work presented here.

The *Globus Security Infrastructure (GSI)* [5] is based on

2

the Grid Security Architecture introduced in [6]. The GSI is used to manage mutual authentication between the local and the remote machine, and authorization on the remote system. The core part of the security environment from the user's point of view is a temporary *User Proxy Certificate*, which is generated using the user's X.509 certificate and Globus' `grid-proxy-init` service. This temporary certificate is used by all Globus computations where it acts on behalf of the user.

The *Globus Resource Allocation Management (GRAM)* [7] Client submits a GRAM job request to a remote machine, which includes the use of the User Proxy Certificate. The remote entity performing the mutual authentication is the *GRAM Gatekeeper*, which also maps the user represented by the temporary certificate to a user name known locally on the remote system. The interface to the batch subsystem is the *GRAM Job Manager*. It is executed using the

user's local identity on the remote machine and performs tasks like submission of subsystem specific job requests, status monitoring and job control.

The GRAM Client retrieves job status information from the Job Manager via a callback mechanism. The job states defined in the GRAM context are: pending, active, done and failed.

To transfer and access data Globus has implemented the *Global Access to Secondary Storage (GASS)* which is described in [8]. On the client side a GASS Server is started the URL of which is given to the Gatekeeper within the GRAM job request. All file transfer actions directed to the local machine refer to this GASS Server URL. E.g. standard output and error files related to a job computed on a remote machine are transferred from the batch subsystem via the GASS Client (which is attached to the Job Manager) to the GASS Server.
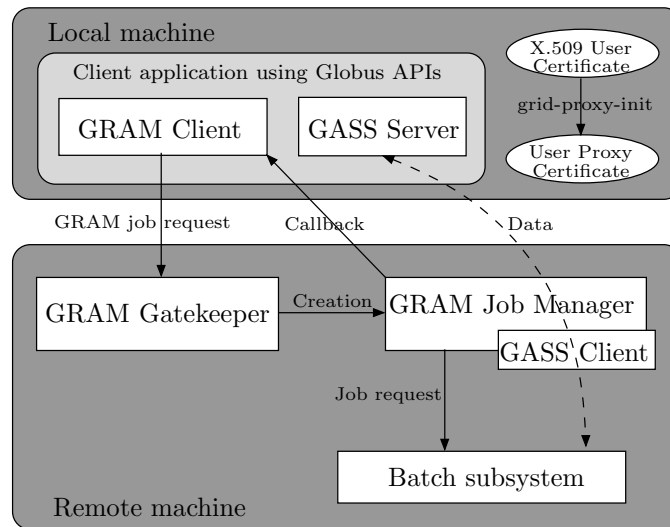

Fig. 2: Globus architecture

3 The UNICORE – Globus interoperation framework

To bridge UNICORE and Globus solutions to the following key aspects had to be developed:

- Translating UNICORE requests for job submission, output retrieval, and status queries to the corresponding Globus constructs.

- Mapping of permanent UNICORE user certificates to temporary Globus proxy certificates.

These functions were to be implemented without changes to the respective architectures. This was a management requirement to prevent interference with ongoing work and delivery schedules of the UNICORE and Globus development.

The conclusive design solution chosen to meet the requirements identified above was to enhance the Target System Interface. The resulting *Enhanced Target System Interface (ETSI)* is handled by UNICORE as one of many target systems. To Globus the ETSI acts as a client application using standard Globus APIs. The overall structure of the Enhanced Target System Interface is depicted in Figure 3 and described in the following sections.
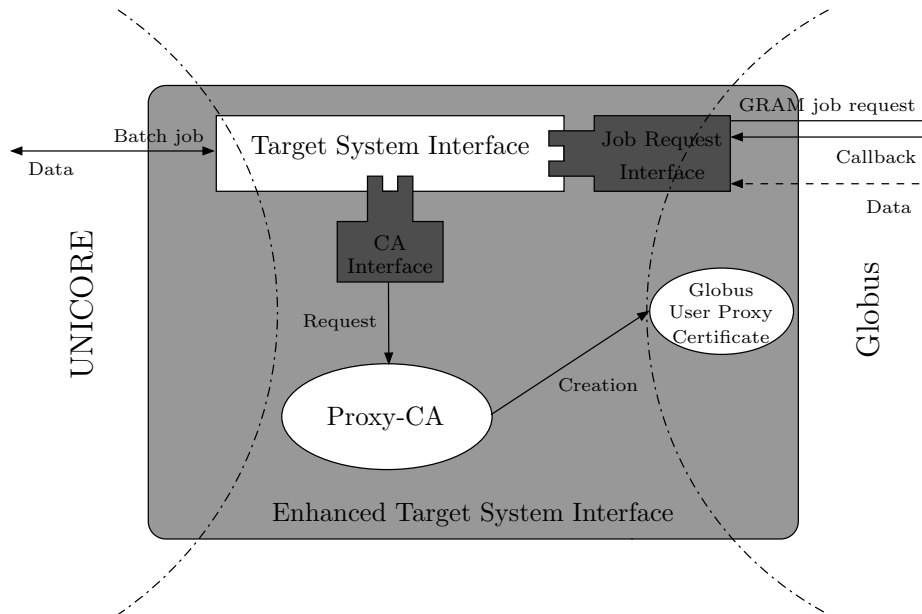
3

Fig. 3: Enhanced Target System Interface

3.1 Design of the UNICORE - Globus job submission

The basics of the UNICORE - Globus job submission are as follows:

A user creates a UNICORE job for the target system "Globus" using the UNICORE client. The UNICORE job is submitted to the NJS which sends a *submit* request to the TSI on the server running the Enhanced Target System Interface.

Instead of communicating directly with a local batch system, the *Job Request Interface (JRI)* translates the UNICORE job request parameters into the GRAM *Resource Specification Language (RSL)* [9] which provides a common interchange language to describe resources. The new Globus job description is submitted to a Globus Gatekeeper which manages the delegation of the job to a Globus target system. The job is executed there and the results of the job are sent back to the JRI which transmits them back to UNICORE again. The user can work with the results in the UNICORE graphical interface.

3.2 UNICORE - Globus certification mapping

A very important topic of UNICORE and Globus is to guarantee secure access to remote resources. The security models of both UNICORE and Globus base on public key

technology using X.509 certificates. But the authentication mechanisms of both systems differ in detail.

UNICORE signs each part of the job with the user's certificate. This guarantees the integrity of jobs and authenticates the submitting entity of a job. Globus uses proxy certificates and delegation. A user proxy certificate is a temporary credential which allows processes being created on behalf of the user to acquire resources, etc., without additional user intervention [6].

A mechanism had to be developed which issues a Globus certificate for a UNICORE user. One possibility to achieve this was to use a Globus tool called *MyProxy* [10]. The MyProxy package provides a secure method for portal users to access resources using a limited proxy working within the Globus Security Infrastructure. As part of the package, a MyProxy server is set up on a trusted host for a site or an application specific portal in order to maintain delegated credentials for users that are valid for a chosen duration.

However, this solution would require architectural modifications in UNICORE and is therefore not relevant for our solution. For this reason a specific Certification Authority, the *Proxy Certification Authority (Proxy-CA)*, has been developed without loosing basic security requirements like authentication, integrity, confidentially and access control [11]. The Proxy-CA is integrated into the Enhanced TSI by the *Certification Authority Interface (CAI)* component (see below).

4

4 Implementation

For the development of the Enhanced TSI we have made the following basic implementation decisions. The software is implemented in C and in Perl. The original UNICORE TSI is written in Perl while the Globus APIs used in the Job Request Interface are implemented in C. The communication between the JRI, the CAI, the TSI, and the Proxy-CA is realized via a socket mechanism. Because changes in the respective architecture were not to be made all functions had to be implemented in the ETSI.

4.1 The Job Request Interface (JRI)

The Job Request Interface manages the communication between the TSI and Globus and performs

- translation of a UNICORE job description into a RSL description,

- job submission to a Globus Gatekeeper,

- job status forwarding, and

- data transfer between UNICORE and Globus.

When the Enhanced TSI is started a socket connection is established between the TSI and NJS. The TSI waits for UNICORE requests like *submit*, *status*, or *retrieve output*. When a submit request is received, the JRI creates an RSL string from the UNICORE internal format. The RSL string contains especially the body of the user's job, the resource requests in Globus syntax, the GASS server URL, and the location where to return the output files.

In addition the JRI initializes a GRAM client and a GASS server using the Globus API. Via the GRAM client the JRI submits the RSL string to the designated Globus Gatekeeper. The JRI also registers for a callback mechanism with Globus. The Gatekeeper returns an identification of the GRAM Job Manager to the JRI. The JRI uses this information to establish the correspondence between the UNICORE job identifier and the Globus identifier. The information is maintained in a table and allows to relate output and status information returned by Globus to a particular UNICORE job.

The job now executes under control of Globus, independent of UNICORE. Whenever the job status changes, Globus returns the new status to the JRI via callback automatically. The JRI translates this status into a UNICORE status description and saves it in a table. If the user requests a new job status, the TSI receives the request, and,

instead of issuing a *qstat* command to a local batch system, the current contents of the table are filtered and the status information from the user's jobs is returned to the client. Upon job completion Globus sends back the results to the GASS Server which saves the data in a location previously designed by the JRI. Output data is now available to the UNICORE user in the normal fashion.

4.2 The Certification Authority Interface (CAI)

The Certification Authority Interface bridges the TSI and the Proxy-CA to obtain a temporary Globus proxy certificate and to sign the Globus job generated in the JRI. The UNICORE job submitted to the target system was signed with a UNICORE user certificate. Globus accepts proxy certificates generated by any trusted CA. The CA which is part of the Extended TSI is trusted by the Globus execution system participating in this project. Implicitly the UNICORE CA is also trusted. The Proxy-CA runs on a dedicated secure server to protect it from unauthorized access.

The proxy certification required by Globus is obtained as follows (see Figure 4): For each submit request, the CAI extracts the username from the job request and transmits it to the Proxy-CA. The Proxy-CA maintains a database containing username, Globus user certificate, and proxy certificate. First, the Proxy-CA checks if the username exists in the database. If it does not exist, a Globus certificate is generated by the CA and stored in the database. If the username exists, and therefore also a Globus certificate, the proxy certificate is checked for validity. If it is expired, a new one with a lifetime of eight hours is generated and signed with the Globus certificate. The valid proxy certificate is returned to the CAI. It is used to sign the job in the JRI and it is also stored in a file accessible to the Globus GSI.

5 Status

The prototype has been implemented meeting the requirements outlined in section 3. It's capabilities have been demonstrated successfully on SC2001 in Denver. Jobs can be submitted to Globus systems from a UNICORE client and the results of the computation are returned to the user. This extends the resources available to UNICORE users and provides Globus users with a graphical job submission interface.
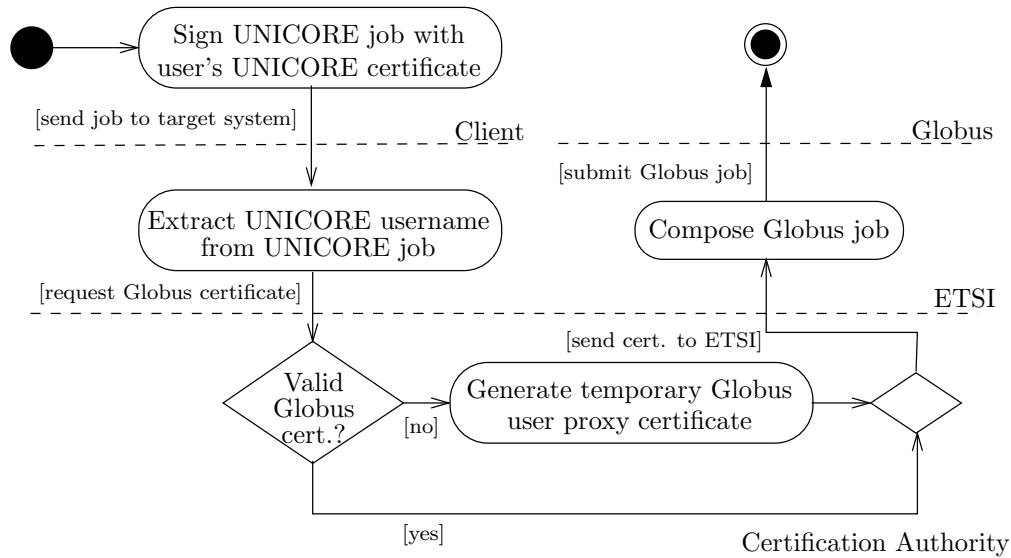
Fig. 4: Certificate mapping

6 Outlook

For a full integration of all tasks required by Grid users (e.g. multi-step and multi-site jobs) additional development is needed. To accomplish this, the project GRIP (Grid Interoperability Project, IST-20001-32257) [12] is funded by the European Commission with a grant period from January 2002 to December 2003. The GRIP project objectives are among others

- A better integration of UNICORE and Globus using the results and insights gained while developing this prototype. This includes also possible changes in the respective architectures. For example the certification mechanism may be integrated into UNICORE.

- Contribution to the Grid standards proposals at the Global Grid Forum [13] and prototype implementations of relevant standards.

- Support for bio-molecular and meteorological applications in the combined UNICORE - Globus environment.

The UNICORE - Globus interoperation has demonstrated the feasibility of Grid interoperability. Future developments will allow both UNICORE and Globus users comfortable, seamless and flexible access to resources distributed in different Grids. The results of the work described here are proof that two independent Grid developments that complement each other like UNICORE and Globus can be combined successfully.

References

[1] "UNICORE Forum." http://www.unicore.org.

[2] "The Globus Project." http://www.globus.org.

[3] I. Foster, C. Kesselmann, and S. Tuecke, "The anatomy of the grid: Enabling scalable virtual organizations," *International Journal of Supercomputer Applications*, vol. 15, no. 3, 2001.

[4] D. W. Erwin and D. F. Snelling, "UNICORE: A grid computing environment," in *Proceedings of Euro-Par 2001*, pp. 825–834, Springer LNCS 2150, August 2001.

[5] "Introduction to grid computing and the Globus Toolkit." Tutorial, October 2001. http://www.globus.org/training/grids-and--globus-toolkit/IntroToGridsAndGlobus-Toolkit.pdf.

[6] I. Foster, C. Kesselman, G. Tsudik, and S. Tuecke, "A security architecture for computational grids," in *ACM Conference on Computers and Security*, pp. 83–91, ACM Press, 1998.

[7] K. Czajkowski, I. Foster, N. Karonis, C. Kesselman, S. Martin, W. Smith, and S. Tuecke, "A resource management architecture for metacomputing systems," in *The 4th Workshop on Job Scheduling Strategies for Parallel Processing*, pp. 62–82, Springer-Verlag LNCS 1459, 1998.

[8] J. Bester, I. Foster, C. Kesselman, J. Tedesco, and S. Tuecke, "GASS: A data movement and access service for wide area computing systems," in *Proc. IOPADS'99*, ACM Press, 1999.

[9] "Globus Toolkit developer tutorial part 5: Ressource management." Tutorial, June 2000. `http://www.globus.org/tutorial/slides/-Dev20html/dev_05_gram_duroc_rsl/`.

[10] J. Novotny, S. Tuecke, and V. Welch, "An online credential repository for the grid: MyProxy," in *Proceedings of the Tenth IEEE International Symposium on High Performance Distributed Computing (HPDC-10)*, IEEE Press, August 2001.

[11] M. Romberg, "The UNICORE grid infrastructure." to appear in: Scientific Programming, Special Issue on Grid Computing.

[12] "GRIP." `http://www.grid-interoperability.org`.

[13] "Global Grid Forum." `http://www.gridforum.org/`.