

# The UNICORE Architecture

## Seamless Access to Distributed Resources

Mathilde Romberg  
Forschungszentrum Jülich GmbH  
Central Institute for Applied Mathematics  
D-52425 Jülich, Germany  
m.romberg@fz-juelich.de

### Abstract

*Seamless access to different systems of different vendors at different sites is an important prerequisite to effective and efficient use of distributed resources. Learning about new systems, new software, and new interfaces is a time-consuming task for users who actually want to run their applications. UNICORE is a project to overcome these difficulties by providing a uniform interface for job preparation and control which gives seamless and secure access to supercomputer resources. It is an ambitious project delivering a production ready prototype within two years. The presentation will focus on the UNICORE architecture, especially the protocol and the underlying security mechanisms.*

### 1. Motivation

Users solving large problems in computational science usually need resources on a variety of systems at different locations. The demand for resources often exceeds those that can be obtained at one site. This causes the users to work in different environments using one system at one site on one day, using a different system at a different site the next day, or using multiple systems at different sites simultaneously. In addition to the shortage of resources at a site users typically need different systems because they have complex pre- and post-processing tasks which run best on another architecture than the main application. As a consequence, the users need to learn the details of the environment at each site like system commands and options and site specific conventions (user identification, security rules, limits, ...). Therefore scientists often continue to work at the site and on the system they know even if their application is better suited for another architecture or a larger system. This results either in sub-optimal use of expensive resources or leaves solvable problems unsolved. This led

the UNICORE<sup>1</sup> project to create a system which provides seamless access to distributed computing resources.

### 2. Solutions and projects

There are several projects dealing with access to remote resources and especially meta-computing, each covering different aspects of the total picture. At the one end of the spectrum developments want to enable the user to easily access computer resources by providing application specific interfaces. These allow users to solve their computational problem using application terms instead of computer hardware and software system terms. The project WebSubmit at NIST is an example (see [9]). It is a Web-based framework providing seamless access to applications (i.e. Gaussian 94) on a collection of heterogeneous computing systems. At the other end are projects dealing with grid computing, with a computational grid being defined as a hardware and software infrastructure that provides dependable, consistent, pervasive, and inexpensive access to high-end computational capabilities (see [5]). One member of this group is the Legion project at the University of Virginia (see [7]). Legion is a meta-system software project using object-oriented technology. It's goal is to provide a single, coherent virtual machine that addresses the issues of scalability, programming ease, fault tolerance, security, and site autonomy. The Globus project (see [4]) establishes a software framework for grid applications by providing a meta-computer toolkit. This includes for example services for resource allocation and process management, communication, data access and security. These basic mechanisms are

---

<sup>1</sup>UNICORE (UNiform Interface to COmputer REsources) is a project funded by the German Ministry of Education and Research (bmb+f). UNICORE is developed by a consortium of people from universities, national research laboratories, software industry, and computer vendors. It is a two years project ending in December 1999. For further information on the project and the project partners refer to <http://www.fz-juelich.de/unicore>.

used to construct various higher-level meta-computing services, such as parallel programming tools and schedulers. Both projects focus on meta-computing at application level and combine a large number of distributed systems to run one huge application which has to be adapted to the framework it wants to use. The projects mentioned above and others in the area of desktop access to remote resources are described in short in [8].

### 3. The UNICORE framework

The idea behind UNICORE is to support the users by hiding the system and site specific idiosyncrasies and by helping to develop distributed applications. Distributed applications within UNICORE are defined as multi-part applications where the different parts may run on different computer systems asynchronously or sequentially synchronized. A UNICORE job contains a multi-part application as described above augmented by the information about the destination systems, the resource requirements, and the dependencies between the different parts. From a structural viewpoint a UNICORE job is a recursive object containing job groups and tasks. Job groups themselves consist of other job groups and tasks. UNICORE jobs and job groups carry the information of the destination system for the included tasks. A task is the unit which boils down to a batch job for the destination system.

The design goals for UNICORE include

- an uniform and easy to use graphical user interface,
- an open architecture based on the concept of an abstract job,
- a consistent security architecture,
- minimal interference with local administrative procedures,
- exploitation of existing and emerging technologies,
- zero-administration user interface through standard Web browser and Java applets, and
- a production ready prototype within two years.

UNICORE is designed to support batch jobs, it does not allow for interactive processes. At the application level asynchronous meta-computing is supported allowing for independent and dependent parts of a UNICORE job to be executed on a set of distributed systems. The user is provided with a unique UNICORE user-id to uniformly get access to all UNICORE sites. An intuitive graphical user interface (GUI) allows job preparation and control. It should be noted that the prototype excludes meta-computing at the application level (synchronous meta-computing), resource brokerage, and interactive applications including application steering.

## 4. The UNICORE architecture

A three tier architecture has been developed for UNICORE consisting of user, UNICORE server, and batch-subsystem level. It is compliant with the definition of a standard three-tier-architecture which Fox and Furmanski gave in [6] for distributed computing architectures. UNICORE emphasizes the user and the middle tier.

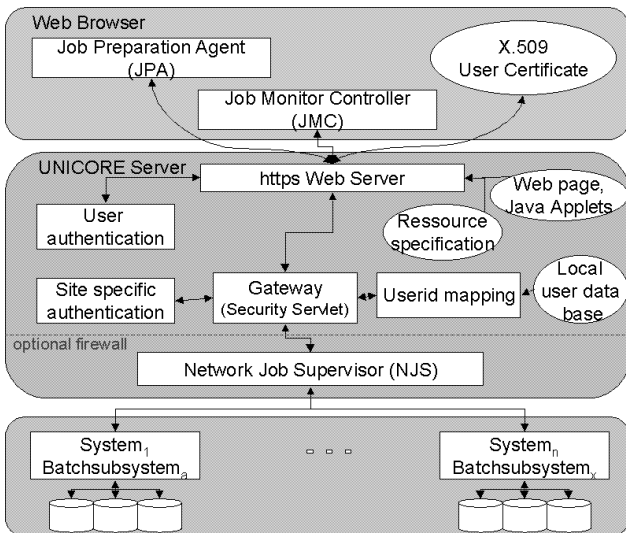
A main component of UNICORE is the underlying security architecture which is based on secure http (https) using X.509 certificates for authentication of users, servers, and UNICORE software. The user's certificate is his/her unique UNICORE user identification which has to be provided by the local Web browser when connecting to the UNICORE server. The unique user identification is translated by the UNICORE server into the user's user-id on the execution host. This mechanism eliminates the need to install uniform UNIX uid/gid pairs for UNICORE users and interferences with the local user administration at the UNICORE sites.

Another key component is the abstract job description language which permits the GUI to seamlessly build a job from the user input and allows the server to translate it to a real job for the destination system. The abstract job is defined as a part of the abstract job object (AJO) which is a recursive Java object specifying the protocol between GUI, server, and system.

In UNICORE a UNICORE site (Usite) is defined as a computer center offering a UNICORE server and execution hosts grouped in so called Vsites. A Vsite (virtual site) consists of systems at one Usite sharing the same data space. The file systems available at the Vsites of a Usite are called Xspace. All data available to a UNICORE job constitute the UNICORE file space (Uspace). Thereby the data model used in UNICORE distinguishes between data inside (Uspace) and outside (Xspace and data from the user's workstation) of UNICORE. All data needed in UNICORE for a job has to be specified by the user and is imported into the Uspace. Analogously data created within UNICORE (in the Uspace) has to be exported to an external file space. UNICORE manages these data transfers transparently for the user. The architecture is shown in figure 1 and explained in more detail below.

### 4.1. User level

The UNICORE user interface takes advantage of existing Web browsers and the https protocol (see [3]). This SSL (secure socket layer) based protocol makes sure that the user has to authenticate him-/herself before getting access to UNICORE. During the SSL handshake between the UNICORE server and the user's Web browser the server first presents its X.509 certificate to the browser in order to be validated. Then the user's certificate is given to the Web



**Figure 1. Detailed architecture**

server for user authentication. The signed applet for the job preparation agent (JPA) or the job monitor controller (JMC) is loaded from the server into the Web browser only in case of successful user authentication. The applet certificate is checked to assure the user that the software has not been tampered with and can be trusted. The use of applets has the advantage that the users always work with the latest version of the software. The GUI provided through the applets assists the user to create uniform jobs or status requests independent from the system it will run on. These uniform jobs are generated by the JPA (or JMC) in the form of an AJO, the transferable unit between the UNICORE components. The hierarchically structured jobs, which may consist of several sub-jobs for different destination systems, and other requests are transferred to the UNICORE server. Messages, standard job output, and status information are sent back to the user and displayed in a consistent way.

#### 4.2. UNICORE server level

The UNICORE server consists of

- the https Web server which provides the UNICORE Web page,
- the signed Java applets,
- resource information about the available execution systems at the Usite, which are provided together with the applet to the user to support him/her in generating jobs suitable for the destination system,
- the user authentication provided by https by checking the user's certificate,
- the Java security servlet (gateway) which maps the user's certificate to the user's id at the target system;

for sites that require the use of smart cards or run DCE (distributed computing environment) it also offers an interface for additional site specific authentication,

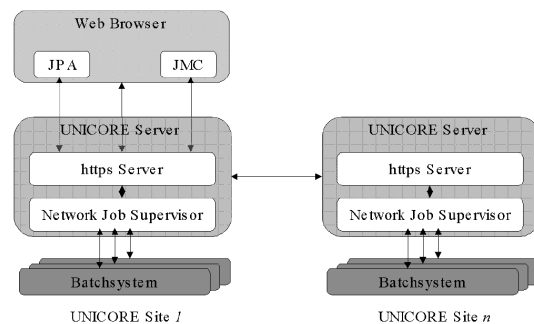
- the network job supervisor (NJS) which does the job management. The NJS translates the AJO into one or more batch jobs for the destination system(s), submits the batch jobs, and controls them. In addition, it transparently transfers data to and from the destination system for the job and makes sure that the dependent parts of the UNICORE job are scheduled in the predefined sequence.

For sites using firewalls the UNICORE server can be separated into the Web server and the NJS part with the firewall in between. In this case the Web server has to sit on the firewall system while NJS runs on a system within the firewall.

#### 4.3. Batch-subsystem level

The third tier contains the destination systems with their batch systems and data storage. One NJS can support multiple destination systems (Vsites) at one UNICORE site. Jobs and requests are submitted to these systems and system messages as well as job output are made available to the UNICORE user.

The architecture described above shows the UNICORE components at one site. The whole UNICORE picture contains multiple UNICORE servers, one at each Usite, providing access to the resources at that site. The different servers are connected so that (parts of) UNICORE jobs, data, and control information can be exchanged to support distributed applications or to allow the user to contact any UNICORE server. Figure 2 gives an overview.



**Figure 2. Architecture overview**

## 5. Implementation of UNICORE

### 5.1. Basic implementation decisions

At the time the project was started some basic implementation decisions were made:

- the use of the World Wide Web as user access mechanism and Java as implementation language,
- strong security mechanisms for user authentication,
- minimal impact on the local administration procedures effective at the UNICORE sites, and
- the use of the resource management system Codine provided by Genias Software GmbH as part of NJS.

The decisions were driven by the idea to use techniques and products which are already widely available and well proven.

### 5.2. Security architecture

These basic decisions influenced the design and implementation of several components. For the security architecture to go with WWW and Java the https protocol has been chosen as a key component. Https together with the X.509 certificates guarantees mutual authentication of the UNICORE 'players' server, user (Web browser), and software (see [2], [10]). The user then knows he/she contacted a valid UNICORE server and that the software has not been tampered with. On the other side the server knows the user is the one he/she claims to be. Different https servers were evaluated but the JigsawSSL server provided by the W3 consortium with the SSL extensions developed at the Institute for Applied Information Processing and Communication (IAIK) at the University of Graz, Austria, (<http://jcewww.iaik.tu-graz.ac.at/Jigsaw/jigsaw1.htm>) has been selected to build a basis for the UNICORE server. It is written in Java and already provides the needed security features while other products demanded additional implementation effort.

The graphical user interface accessible via a UNICORE Web server is implemented as signed applets as stated above. The GUI consists of two parts: The job preparation agent (JPA) to create and submit UNICORE jobs and the job monitor controller (JMC) to monitor the job status, control the jobs, and deal with the output.

The security architecture also had to include a mechanism to provide the local user identifications to be placed into the UNICORE job before it is submitted to the execution system. With the X.509 user certificate being the uniform and unique UNICORE user identification a mapping process has been implemented in the form of a Java servlet which maps the user's distinguished name to the

corresponding user-id. Each UNICORE site administration therefore maintains a user data base for the local mapping. Firewalls are another aspect of the security architecture. It has to be guaranteed that sites using firewalls can run the UNICORE server. Therefore the two parts of the UNICORE server, the Web server and the NJS, can be run on different systems. The Web server has to be installed on the firewall system and the NJS on a system inside the firewall. The communication between the two components is done via IP socket connection to a site selectable port.

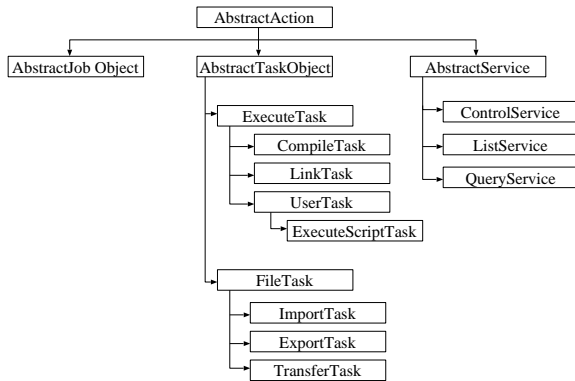
The implementation of the security architecture relies on the existence of a Certificate Authority (CA) to generate the X.509v3 certificates for the server systems, the software developers, and the users. As the X.509 standard is implemented differently by different browsers (i.e. Netscape and Microsoft) a decision for a specific format had to be taken. The Netscape format was adopted because of the availability of this browser on all relevant user platforms, PCs and UNIX workstations. A method for the secure transfer of the user certificates and the distribution of the necessary information to the UNICORE sites for the user-id mapping is defined based on the guidelines developed by the German Research Network Policy Certificate Authority (DFN-PCA; <http://www.pca.dfn.de/eng/dfnpca>).

### 5.3. Protocols

The UNICORE protocols define the form of requests for some action to be performed (high-level protocol) and ensure the security of communication between the UNICORE components (low-level protocol). In the following the high-level protocol is described. It defines a client-server type of communication. JPA/JMC act as client while NJS (resp. the gateway) acts as both client and server depending on the partner. It is server in the JPA/JMC – NJS communication and client when talking to another NJS which it has contacted i.e. to send a job group of a UNICORE job. It is an asynchronous protocol. This design is suitable for batch processing (UNICORE supports batch jobs) and it is more robust than a synchronous protocol. By minimizing the length of time that an interaction takes the asynchronous protocol protects against any unreliability of the underlying communication mechanism.

The UNICORE protocol is implemented as a Java object called the abstract job object (AJO). It specifies all actions to be performed by the NJS which are grouped together in the Java class AbstractAction. Figure 3 shows the class hierarchy of AbstractAction building the AJO: The class AbstractJobObject contains the directed acyclic job graph representing the job components (AbstractTaskObject and AbstractJobObjects) together with their dependencies and information about the destination site (Vsite), the user, site specific security, and the user account group. The recursive

structure of the AJO allows for the AJO to contain sub-AJOs (corresponding to job groups in a UNICORE job) which are intended for other execution systems. The abstract task object (ATO) and the abstract service for job monitoring are the non-recursive parts of the AJO. A Java



**Figure 3. AJO object hierarchy**

class Outcome is defined to contain the status of an abstract action and the results of its execution. Outcome contains a subclass for each subclass of AbstractAction which are associated to give the results of an abstract action.

#### 5.4. Resource description

Currently UNICORE has incorporated a simple model for resources. It contains the main resources a user needs for batch job specification and information about available software (compilers, libraries, program packages). An abstract task object (ATO) as the entity to be translated into a real batch job for a destination system contains the information about the required resources for the job. UNICORE supports resource requests for the number of CPUs (or processor elements), the amount of execution time, the amount of memory, and the amount of disk space needed, both permanent and temporary. These values are specified by the user during job preparation in the JPA .

Each UNICORE site provides a so called resource page reflecting resource information about their Vsites. Besides minimum and maximum values for the resources needed for batch submission it contains information about the system architecture, performance, and operating system as well as available application and system software. This information is prepared by a UNICORE site administrator through a resource page editor. It is stored in ASN1 format for the JPA to include it into the GUI supporting the user in creating a job suitable for the selected destination system.

#### 5.5. Network job supervisor

The NJS consists of two main components, a java translation server (JTS) and a system for job control and scheduling which in the current implementation is based on Codine. NJS has a variety of tasks to fulfill:

- transform the abstract job into a Codine internal format,
- split it into the job groups destined for different sites,
- distribute and control the job groups,
- translate the abstract specifications into the local system specific nomenclature using translation tables,
- submit the batch jobs to the execution system,
- create a UNICORE job directory to contain the data for and created during the job run,
- collect the standard output and error files from the batch jobs belonging to one UNICORE job and make them available to the user via the job status interface, and
- initiate all necessary data transfers, imports, and exports. In the JPA the user specifies which data including binaries (executables) have to be transferred to the UNICORE job directory for the job steps (import), which data created during the job steps has to be put to permanent file space somewhere (export), and which data has to be transferred between dependent job steps. The user has to know about the locations of his/her data and binaries and tell UNICORE about it to do the file transfers.

The scheduling done by the NJS is limited to the delivery of the generated batch jobs to the destination systems in the specified sequence. It has no means of influencing the scheduling on the destination systems. Jobs delivered through UNICORE are treated the same way any other batch job is treated on a system. This results from the basic design decision for UNICORE to have minimal impact on the local administration. A negative effect of this decision is that UNICORE can neither estimate the turnaround time for a job nor influence the scheduling of a particular batch job (i.e. to allow for synchronous execution of jobs on different systems). On the other hand, the acceptance for running UNICORE is much higher at the sites if site autonomy is provided.

The UNICORE site administrator together with the Vsite system administrator establishes the environment for running UNICORE. This includes setting up the translation tables for the translation of the abstract job into the real batch job and the connection between UNICORE server and batch system.

## 5.6. File transfer

The data model distinguishes between data inside and outside UNICORE. Therefore data has to be transferred from outside to a Uspace (import) and vice versa (export) as well as between the Uspaces located at different UNICORE sites (transfer). The import of data is implemented for data from the user's workstation and from UNIX file-systems at Vsites while export is done to Xspace at a Vsite. Files from the user's workstation needed in a job are put into the AJO. They are transferred together with the job to a UNICORE server on the https connection. Imports from Xspace to Uspace and exports from Uspace to Xspace are always local operations performed at a Vsite. They are implemented as a copy process available at the Vsite.

The file transfer between Uspaces has to be accomplished through NJS – NJS communication via the gateway (security servlet) for user-id mapping. The https protocol guarantees the secure communication between the sites. As this solution has disadvantages with respect to transfer rates especially for huge data sets UNICORE is working on alternatives. Another open issue is the automatic transfer of result files back to the user's workstation. The current implementation sends data back to the workstation only on user request while the user is working with the JMC.

## 5.7. Status

UNICORE is running at different German sites including the Forschungszentrum Jülich (FZ Jülich), the Computing Centers of the universities of Stuttgart (RUS) and Karlsruhe (RUKA), the Leibniz Computing Center of the Bavarian Academy of Science in Munich (LRZ), the Konrad-Zuse Zentrum für Informationstechnik in Berlin (ZIB), and the Deutscher Wetterdienst in Offenbach (DWD). The systems covered are Cray T3E, Fujitsu VPP/700, IBM SP-2, and NEC SX-4.

The functions offered to the users by the JPA include creation of a new UNICORE job, loading of an old UNICORE job for resubmission, and loading and modification of an old UNICORE job. The interface allows to specify a sequential dependency between job groups and/or tasks at the same level of the job tree. In addition each dependency can be augmented by the names of the files to be transferred from one to the other. UNICORE then guarantees that the specified data sets created by the predecessor are available to the successor. In the current state of implementation the interface offers support for the creation of jobs containing script tasks (to include existing batch applications) and compile-link-execute tasks (for new applications). At this point in time the compile is implemented for F90.

The JMC shows the job status of the user's UNICORE jobs in a display similar to the one of the JPA. The icons are col-

ored to reflect the job status in a seamless way. Depending on the chosen level of detail the status is displayed for job groups and/or tasks. The standard output and error files can be listed and/or saved for tasks.

## 6. Summary and outlook

UNICORE offers an easy to use and seamless user interface to supercomputer resources hiding the system and site specific nomenclature and thereby allows the user to focus on the application. Users then can use different systems at different sites for their computations without modifying the application for the different environments; this is all done by UNICORE. Access to the systems is provided through the user's UNICORE X.509 certificate which together with the usage of https and signed applets guarantees a high level of security both for the user and the sites. A main advantage of UNICORE is that supercomputer resources which tend to be concentrated at only a few sites can be used more efficiently for the large applications because the effort to learn how to use them is minimal. Users will of course appreciate enhancements to UNICORE, for example:

- Application specific interfaces for standard packages like Ansys or Pamcrash will make life easier especially for users from industry.
- Application steering is needed to control the progress of a computation.
- A resource broker which supports the users in a way that they can specify the needed resources on a more abstract level and the broker finds the appropriate execution server for it. Together with accounting functions and load information the resource broker can find the best system for an application with given time constraints.
- For the big grand challenge problems the integration of meta-computing is a topic. This extends the usage of distributed systems in one UNICORE job to the synchronous use for a single application (UNICORE task).

UNICORE has laid a solid basis for seamless computing and is well accepted by non-expert users. Future developments will integrate features to support more users with more sophisticated applications.

## References

- [1] J. Almond and D. Snelling. Unicore: Uniform access to supercomputing as an element of electronic commerce. *Future Generation Computer Systems*, 613:1–10, 1999.
- [2] D. Durbin, R. McGregor, J. Owlett, and A. Yeomans. *Java Network Security*. Prentice Hall, 1998.

- [3] J. Feghhi, J. Feghhi, and P. Williams. *Digital Certificates - Applied Internet Security*. Addison-Wesley, 1998.
- [4] I. Foster and C. Kesselman. Globus: A metacomputing infrastructure toolkit. *International Journal of Supercomputer Applications*, 11(2):115–128, 1997.
- [5] I. Foster and C. Kesselman. Computational grids. In I. Foster and C. Kesselman, editors, *The Grid: Blueprint for a New Computing Infrastructure*. Morgan Kaufman Publishers, 1998.
- [6] G. C. Fox and W. Furmanski. High performance commodity computing. In I. Foster and C. Kesselman, editors, *The Grid: Blueprint for a New Computing Infrastructure*. Morgan Kaufman Publishers, 1998.
- [7] A. S. Grimshaw, W. A. Wulf, and the Legion team. The legion vision of a worldwide virtual computer. *Communications of the ACM*, 40(1):39–45, January 1997.
- [8] Java Grande Working Group on Concurrency/Applications and Argonne National Laboratory, MCS Division. Notes of the 1st international workshop on "desktop access to remote resources". <http://www-fp.mcs.anl.gov/~gregor/datorr/report/datorr-report.doc>, October 1998.
- [9] R. McCormack, J. Koontz, and J. Devaney. Websubmit: Web-based applications with tcl. Technical report, National Institute of Standards and Technology, June 1998.
- [10] Netscape DevEdge Online Documentation. Netscape object signing - establishing trust for downloaded software. <http://developer.netscape.com/docs/manuals/sign-dobj/trust7index.htm>, July 1997.