UNICORE – A Grid Computing Environment

Dietmar W. Erwin

Forschungszentrum Jülich GmbH, Zentralinstitut für Mathematik (ZAM)
D52425 Jülich, Germany

D.Erwin@fz-juelich.de

(revised November 30, 2001)

Summary

This paper gives an overview over the goals, functions, architecture and future development of UNICORE. Its primary goal is to give researchers a seamless access to distributed resources that are available at remote sites. A graphical interface aids users to formulate jobs which are to be performed in a system and site independent fashion. This procedure allows switching between systems without having to change the job. Complex jobs with individual applications running on different systems at different sites may be formulated. UNICORE will perform synchronization and data transfers as required without any user intervention. UNICORE uses X.509 certificates to authenticate users, software, and systems and provide secure communication over the internet.

1. Overview

1.1 Description and Goals

The development of the UNICORE system was conceived in 1997 to enable German supercomputer centers to provide their users with a seamless, secure, and intuitive access to the heterogeneous computing resources at the centers. As a result, project UNICORE[1] was proposed to BMBF, the German Ministry for Education and Research, with the following objectives:

- Foremost, UNICORE – **Un**iform **I**nterface to **Co**mputing **Re**sources – was to hide the seams resulting from different hardware architectures, vendor specific operating systems, incompatible batch systems, different application environments, historically grown computer center practices, naming conventions, file system structures, and security policies - just to name the most obvious.

- Equally, security was to be build into the design of UNICORE from the start relying on the emerging X.509 standard for certificates authenticating servers, software, and users and encrypting the communication over the internet.

- Finally, 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 jobs.

In addition several boundary conditions had to be met: UNICORE had to support operating systems and batch systems of all vendors represented at the partner sites. It had to be non-intrusive insofar as it does not require changes to computing center practices especially for

[1] UNICORE was supported in part by BMBF grant 01 IR 703

user and system administration.In addition to UNICORE's own security model, site-specific security requirements had to be supported. Last but not least, a working prototype had to be demonstrated within the initial funding period of 30 month. This lead to the design and implementation of a vertically integrated solution, combining innovative ideas and proven components, as described below. Like many projects it was started before Grid Computing [1] came the accepted new paradigm for distributed computing. The paper will relate the concepts and developments to the definitions and standards that are emerging from the Global Grid Forum [2].

1.2 Services provided

This section describes the extensive set of services provided by UNICORE. Components of the software are developed either under the funding agreement by the German UNICORE Plus[2] project, the follow-on to the original UNICORE project, or the European EUROGRID[3] project.

Job creation: A graphical interface assists the user in creating complex, interdependent jobs that can be executed on any UNICORE site without changes to the jobs. A UNICORE job, more precisely a job group, may recursively contain other job groups and/or tasks. Figure 1

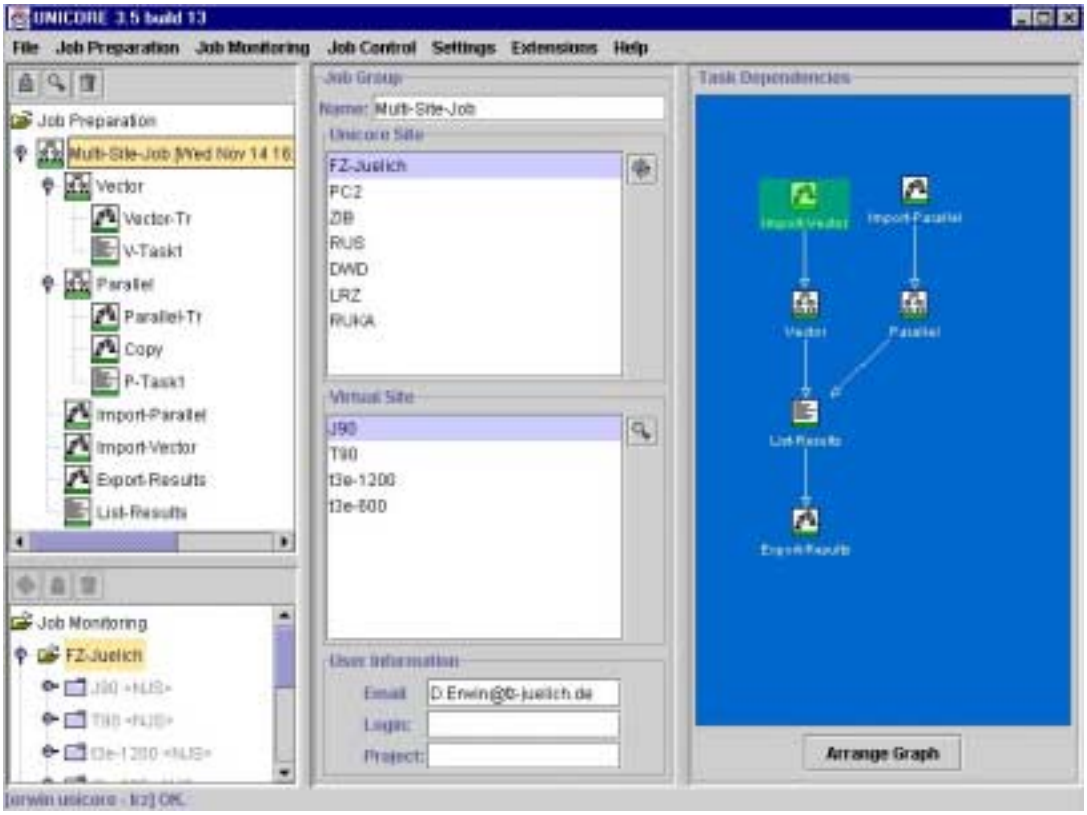

Figure 1: Top level job screen

shows the structure of a job in a typical folder view. Pull down menus give access to additional functions, such as job manipulation, job monitoring, default settings, or online

[2] UNICORE Plus is funded in part by BMBF grant 01 IR 001
[3] EUROGRID is funded in part by the European Commission grant IST-1999-20247

help. Details can be displayed and manipulated by clicking on the appropriate icon. A job group is submitted to a UNICORE site which the user selects prior to submission. For additional details see Page 6. The tasks contained in a job group are *incarnated* into a batch job to be executed on a system at the site or into an action, like a file transfer to or from a storage space. Child jobs groups are transferred to the appropriate site to be incarnated and executed there.

The user may specify temporal dependencies between the entities contained in a job group (see the right hand side of Figure 1). The example shows two tasks, Import-Vector and Import-Parallel running independently. As soon as one of the tasks completes, its successor job group (Vector or Parallel) starts. The task List-Results begins only after both predecessors are completed successfully.

Job management: The user has full control over jobs and data. A color code presented along with the job icon shows the overall status of a job: green, red, yellow, blue, magenta to indicate successful completion, failure, in execution, queued, or waiting for completion of a predecessor, respectively (see Figure 2). The status is available at each level of recursion down to the individual task. It may be refreshed by clicking on the icon or by an automatic timer-driven status update. In addition, detailed log information is available to analyze error conditions. The job output that is written to *stdout* and *stderr* by the execution systems can be reviewed or transferred to the client workstation.

A user may cancel jobs queued for execution anywhere within the UNICORE Grid and terminate executing jobs.

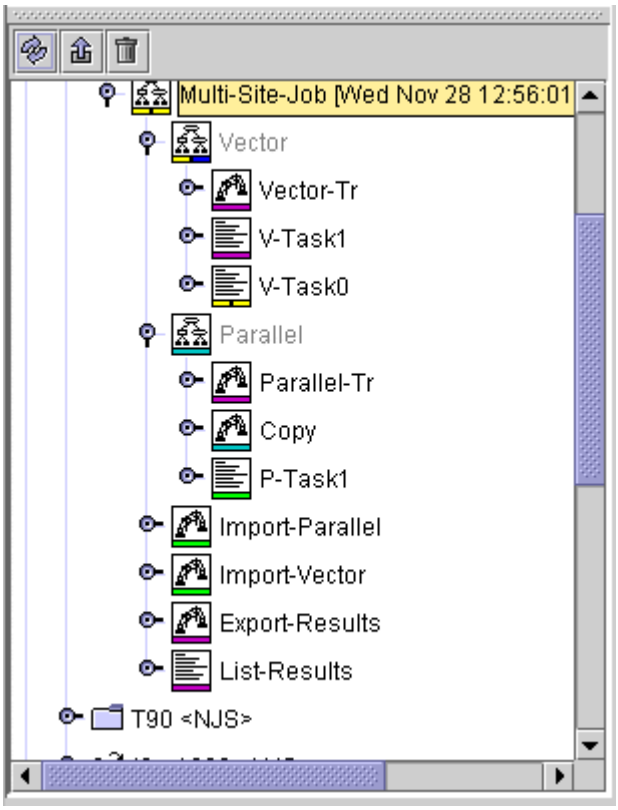

Figure 2: Status information of a UNICORE job

Data management: UNICORE jobs contain task that can be executed at different computing centers. Output created by one task may be used by any of its successors. A temporary UNICORE space, called Uspace for short, is created for each job group. During job creation the user specifies

- which data sets are to be *imported* into the Uspace from the client workstation or any file system or data archive at the UNICORE site to which the user has access,
- which data set are to be *exported* from the Uspace to retain them permanently, and
- which data sets are to be transferred to a different Uspace.

At run time UNICORE performs the necessary data movement without user intervention.

The import, export, and transfer functions are available to the user through the GUI as explicit tasks. Data sets to be imported or exported can also be specified as part of the job. The data management function can also be imbedded into an applications and be performed fully transparent to the user.

Application support: The three services described above provide an effective tool to use resources of different computing centers both for capacity or capability. Many scientists and engineers use application packages. UNICORE provides a two-fold support for these users: For applications without a graphical user interface a GUI can be provided, initially through a custom built plug-in. A tool kit is developed to be released with the next version of UNICORE[4] that simplifies the creation of plug-ins. CPMD (Car-Parrinello Molecular Dynamics) is available as a UNICORE demonstrator of an application plug-in (for Details see [18]). Wrappers are developed for applications with existing graphical interfaces, such as Fluent or MSC Nastran, to combine their known look-and-feel with the system independence and security of UNICORE.

Flow control: The above description covers the functions already provided by the first UNICORE prototype. Its job model can be described as a set of one or more directed a-cyclic graphs. Current research is dedicated to including conditional and repetitive execution of job groups or tasks into the job model. It supports of computational experiments which can be repeated a fixed number of times or until a given condition is reached. This development supports applications, which require special action in case errors occur.

Meta-computing: UNICORE is extended to meta-computing, i.e. simultaneous use of two or more systems by one application, typically an MPI application. Even UNICORE's first prototype recognized and incarnated support for MPI libraries, but did not attempt to co-schedule systems for a very pragmatic reason: Except for one batch system (*CCS* developed at the University of Paderborn) none of the available products supported advance reservation, the prerequisite for co-scheduling. Since additional batch systems, like PBS Pro, are available now with an advance reservation API, meta-computing support will be added to UNICORE.

Interactive support: UNICORE set out to assist batch users, a usage mode which still accounts for over three quarters of the cycles consumed at the HPC centers. UNICORE adds support for interactive use, especially to permit application steering.

Single sign-on: UNICORE provides a single sign-on through X.509V3 certificates. The certificate can be mapped to a local account at each UNICORE site. The account name and the UNIX uid/gid may be different at each site, due to existing naming conventions. In addition, the site retains full control over the acceptance of users based on the identity of the

[4] The final version UNICORE 4.0 will be deployed in August of 2002 according to the project schedule.

individual - the distinguished name - or other information that might be contained in the certificate. Each site can restrict and limit accessible resources at each target system, thus retaining the ultimate control. UNICORE can handle multiple user certificates, i.e. it permits a client to be part of multiple, disjoint Grids. The previous version of UNICORE mapped a certificate to exactly one login name at a site. Experience showed that support for project accounts is required allowing the user to select different account for different projects or to assume different roles with different privileges. This feature is available in the current version of UNICORE.

UNICORE uses the asymmetric key in the certificate to sign each job and all sub-jobs contained in it on transit between sites. This protects against tampering while the job is transmitted over the Internet and it allows to verify the identity of the owner at the receiving end, without having to trust the intermediate site which happens to send the job.

Support for legacy jobs: Even the best solution will not be embraced immediately by users, especially if the existing techniques work. Therefore, UNICORE supports traditional batch processing by allowing users to include their old job scripts as part of a UNICORE job. This approach does not guarantee seamlessness but it simplifies the migration. UNICORE immediately helps users in the following scenario: a users submit a job to a supercomputer, periodically check for its completion, transfer results to a different system, using ftp, and submit a successor job on this system. A simple job group can automate these steps without changing the existing job scripts.

Resource management: The initial UNICORE prototype implemented a simple resource model which is still effective in practice. The resources available at each participating site - computing, data, and software resources - are made known to UNICORE by local administrators. The process is completely decentralized. The currently valid resource information is available to all authenticated UNICORE users at the time of job creation or job submission. Presently, the users select the target system, typically at one of the HPC centers where they have a valid account, and specify the required resources. The UNICORE client is in a position to verify the formal correctness of jobs with respect to resources and alert users to correct errors immediately. For example, if the initial resource specification asks for one hour of connect time on a 512 processor T3E and the user redirects the job to a site allowing at most 256 processors, the user is informed that the job can not be executed as specified. The user is asked to adjust the values, i.e. reduce the number of processors and possibly double the connect time, or select a different site. Current research extends the resource model to support resource brokering, cost models, and resource consumption based on application specific parameters, such as number of atoms in a CPMD simulation.

2. Systems, sites, and user served

UNICORE is designed to be system independent. The client software, a Java application, executes on any Windows PC or Unix Workstation that supports a Java run time environment at the current level (presently 1.3). UNICORE servers (Gateway and NJS) run on Unix systems supporting Java 1.3. Sun Solaris, IBM AIX, SGI Irix, and Linux systems have been tested. Interfaces to the target systems exist for batch systems on Cray, Fujitsu, Hitachi, IBM, NEC, SGI, Siemens, and Sun products. The list of supported batch systems includes various flavors of NQS, IBM's Load Leveler, PBS, PBS Pro, and CCS from Paderborn.

Participating sites are the partners of the UNICORE Plus project and the EUROGRID project (see Appendix 1: UNICORE, EUROGRID, and GRIP project partners). UNICORE is available to users of the centers and researchers at universities who are registered users at the centers. EUROGRID establishes test beds for application in bimolecular research,

meteorology, computer aided engineering, and general high performance processing on the European level.

In addition, the Research Center in Jülich, created a UNICORE Test Grid, simulating a Grid Computing Environment on a dedicated machine which creates the opportunity of testing UNICORE's functions and the graphical user interface without becoming a member of one of the projects or having to apply for resources at a participating center. The web based installation and configuration process is described at http://www.fz-juelich.de/unicore-test. The user follows a simple step-by-step process which downloads, installs, and configures the client, generates certificates, maps them to userids on the simulated UNICORE sites and permits testing of all functions. Of course , access to the production systems in Jülich or at the partner sites is not possible from this test Grid.

2.1 Status

The first version of a prototype has been deployed to all partners and has been tested by a group of selected users. Their feedback has been incorporate. into the second version of UNICORE that has been shipped in August 2001. It has been thoroughly tested in Jülich and is now the production version for the UNICORE and EUROGRID project partners. The final version of the software to be developed as part of the funded projects will be shipped to the partners by mid 2002 as defined in the project plan. After that Pallas GmbH (see Appendix 1) is planning to support the software commercially. UNICORE source, interfaces, and a prototype implementation is available under a license agreement following the Sun Community License. The UNICORE Forum has assumed the role and responsibility of technology distributor [10]

2.2 Other Aspects

UNICORE is designed for existing professionally managed production environments. It assumes that all resources that are consumed are to be accounted for and that the centers are accountable to their governing bodies. A seti@home paradigm is not directly applicable. UNICORE provides the technical infrastructure which allows the centers to pool and exchange their resources either in total or in part. This is the method proposed for the German supercomputer centers by the German Science Council [5]. The associated administrative, organizational, and regulatory problems are beyond the scope of a R&D project and have to be solved by the governmental organization involved.

3. The UNICORE Architecture

UNICORE creates a three-tier architecture depicted in Figure 3 at each UNICORE site.

The UNICORE *client* supports the creation, manipulation, and control of complex jobs, which may involve multiple systems at one or more UNICORE sites. The jobs and actions as defined the user are represented as *Abstract Job Objects,* effectively Java classes, which are serialized and signed when transferred between the components of UNICORE.
The *server* level of UNICORE consists of a Gateway, the secure entry point into a UNICORE site, which authenticates requests from UNICORE clients and forwards them to a *Network Job Supervisor (NJS)* for further processing. The NJS maps the abstract request, as represented by the AJO, into concrete jobs or actions which are performed by the target system, if it is part of the local UNICORE site. This process is called *incarnation*. Sub-jobs that have to be run at a different site are transferred to this site's gateway for subsequent processing by the peer NJS (see Figure 4). Additional functions of NJS are: synchronization of jobs to honor the dependencies specified by the user, automatic transfer of data between

UNICORE sites as required for job execution, collection of results from jobs, especially *stdout* and *stderr*, import and export of data between the UNICORE space and target system, and client workstation.

The third tier of the architecture is the target host which executes the incarnated user jobs or system functions. A small daemon, called the *Target System Interface (TSI)* resides on the host to interface with the local batch system on behalf of the user. A stateless protocol is used to communicate between NJS and TSI. Multiple TSIs may be started on a host to increase performance.

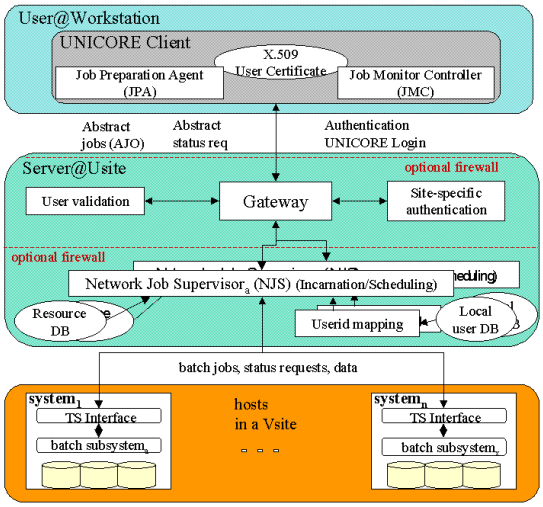

Figure 3: The overall UNICORE architecture at one UNICORE site.

The following technical terms have been defined to describe the entities within a UNICORE based Grid Computing Environment:

- The term *Usite* (short for UNICORE site) is used to identify the participating organization to the Grid. It is the symbolic name that resolves into the URL of the Gateway which control the access. An organization may be part of multiple Grids offering the same or different resources to different communities.
- The term *Vsite* (virtual site) identifies a set of resources at a Usite. Each Vsite is controlled by a Network Job Supervisor (NJS). A Vsite may consist of single supercomputer, e.g. a CRAY T3E, or a cluster of workstations under the control of a distributed batch system, like DQS. The construct was introduced to support flexibility of different system architectures and to give the organization full control over its resources.

The overall UNICORE Grid architecture extends in the following way, as shown in Figure 4.

A UNICORE job is submitted to a Usite. If sub-jobs are destined for a Vsite at a different Usite, NJS connects to the designated gateway and transfers the (sub-) AJO. Architecturally NJS assumes the role of a client. However, the AJO that represents the sub-job was signed by the originating client. The gateway at the final destination can always authenticate the user as if they had submitted the original sub-job directly to the Usite.

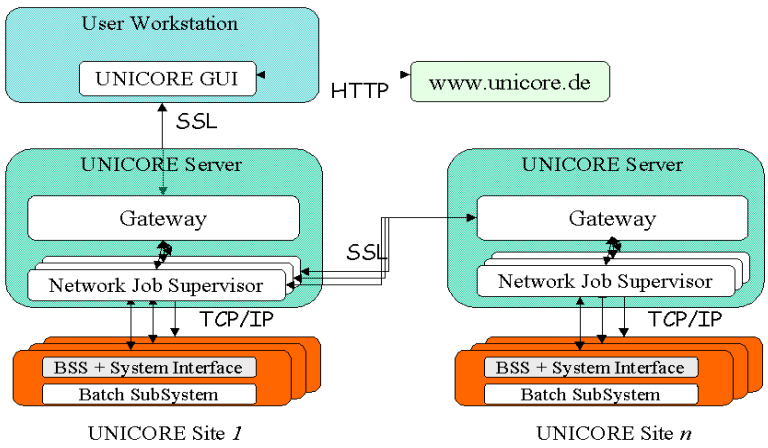

Figure 4: Multi-site UNICORE systems

3.1 Grid Services Used

UNICORE does not depend on any specific implementation of Grid services or Grid software. UNICORE itself implements the complete vertical stack. However, its architecture is modular in the sense that components can be exchanged to make use of different ones. Future projects, for example GRIP (see Project Status and Future Plans), will exploit this feature of UNICORE to implement recommendations for new Grid standards for the Global Grid Forum to interoperate more naturally with Globus.

Security: Certificates according to the X.509V3 standards provide the basis of UNICORE's security architecture. Certificates serve as grid-wide user identifications which are mapped to existing Unix accounts with the option to follow established conventions. Certificates also mutually authenticate peer systems in a Grid. The gateway that decides whether to accept a requests from a client or a peer system and the Network Job Supervisor (NJS) that maps the certificates to a user identification for a target system are trusted software components. They must run on adequately protected systems, normally behind a firewall.

UNICORE's trust model is straight forward. It does not rely on delegation of trust. Each NJS that incarnates an *Abstract Request* for the Vsite it manages also maps the certificate of the originator to the corresponding user identification on the target system. This model is well suited for Grid resources that have to be accounted for in full.

For a smooth operational procedure, all members of a Grid should use the same certificate authority (CA) for both user and server certificates. Handling user certificates from different certificate authorities pose no technical problem: If Grid-A requires certificates from CA-A, and Grid-B only accepts those issued by CA-B, the UNICORE client can handle multiple certificates and the user will have to select the correct one when connection to either Grid-A or Grid-B. This is not different from the present situation where users have different accounts on different systems; now they have a different identity for each Grid.

It is the Usite's choice to accept user certificates from one or more authorities. (The work of the Grid Forum on equivalence of CA policies is invaluable here.) At present, each installation has to make a case by case decision. For example, the research center Jülich operates three separate Grids: A production Grid which is open to users holding certificates issued according to the strict rules of the German research network (DFN) (see [17]). Those users will typically be mapped to existing accounts on the production systems because their identity can be trusted. A second Grid which includes different Vsites is open to partners of the EUROGRID project which operates a temporary CA. As soon as this project has policies which are considered equivalent to those of DFN, they can use the production systems. The third Grid is a UNICORE Test Grid (see [11]) which is open to the word. It uses automatically generated temporary certificates and restricts access to selected systems with no connection to the production systems.

The situation grows more complicated if member installations of a Grid accept only certificates issued by their organization. In that case all partners have to accept two different certificate authorities to allow multi-site jobs involving the special Usite. If more than one site has the same demand, an additional administrative burden is put on all partners. Moreover, multi-site jobs involving these two partner become impossible. Even if technical security issues are resolved, administrative ones are still a major obstacle.

Resources: UNICORE relies on descriptions of resources (hardware and software) which are available at run job creation and submission time to the client. Standards defined by the working groups of the Global Grid Forum on Grid Information Services and Scheduling and Resource Management are valuable for future releases of UNICORE, especially for a resource broker. A replacement of the internal representation of resources will be transparent to the user. Work in this area is carried out in the projects EUROGRID and GRIP by University of Manchester, again taking into account developments in the Global Grid Forum and in contact with other projects, e.g. NIMROD/G [13].

Data: The UNICORE data model includes a temporary data space, called Uspace, which functions as a working directory for running UNICORE jobs at a Usite. The placement of the Uspace is fully under the control of the installation. UNICORE implements all functions necessary to move data between the Uspace and the client. This is a synchronous operation controlled by the user. No attempt is presently made top push data to the desktop from the server. Data movement between Uspace and file systems at a Vsite is specified by the user explicitly or by application support in the client exploiting knowledge about data requirements of the application. NJS controls the data movement which may result in Unix copy command or in a symbolic link to the data. Transfer of data between Usites is equally controlled by NJS. Currently, the data is transferred using a byte streaming protocol or within an AJO for small data sets.

3.2 The Abstract Job Object

The Abstract Job Object (AJO) is the basis for the platform and site neutral specification of requests for computational, data, and software resources. The AJO is a conceptual representation of a *Job* as a collection of possibly interdependent operations which can be carried out on various computational and data services at collaborating sites. The object-oriented structure and syntax of the AJO a specification largely independent of hardware architecture, system software interfaces, and site-specific operational rules. Thus, seamless descriptions of user's work can be created without interfering with site autonomy.

Similar to the platform-independence realized in the Java virtual machine, complete uniformity of the AJO may in some cases conflict with performance goals. There may, for instance, be architectural features of HPC platforms which do not allow an abstraction that

maps uniformly to all HPC platforms. When the detailed control of such features is necessary to achieve the maximum performance of the architecture, a performance versus seamlessness trade-off may result. For example the AJO class CompileTask, which includes parameters strongly affecting machine performance, defines uniform concepts such as optimization levels, to which machine-specific parameters can be assigned. Future work will be required in this area. However, the UNICORE partners have agreed that the initial abstract notion of an abstract job is rich enough to describe most of the functions necessary in a batch oriented supercomputing environment. It is not, however, intended to replace the operating system or its batch subsystem, but to inter-operate with them.

The AJO definition will is available under community license for research purposes through the UNICORE Forum [12] together with sources and a reference implementation.

3.3 Other Grid Services

To make Grid Computing successful at least two other aspects have to be considered:

The system must be made user-friendly. This is one of the goals of UNICORE and other projects. Different approaches will help to accelerate the creation of functional and practical solutions. Eventually, users will demand consistency between the graphical interfaces, the look-and-feel, and the functions. The UNICORE project is prepared to contribute to this process.

The second aspect refers to administration and management. In a Grid environment the actions a user initiates go beyond administrative boundaries. It is not sufficient to have knowledgeable administrators and support staff at each installation. Today, identification of the cause of failures in a complex multi-site job requires the involvement of many administrators with system privileges. New methods have to be devised to analyze problems involving other organizations, since it can not be assumed that local administrators will, for example, grant root privileges to outsiders.

4. Implementation

The UNICORE Client is now implemented as a Java application written in Java 2. It has been tested on Unix and MS Windows systems. The initial implementation used a signed Java applet for the following reasons:

- there is no need to install any software on the client other than the browser,
- the user always works with the most current version of the software, which is especially valuable in the early development stages of a new system.

Note that the applet has to be signed to permit access to the desktop's local file system to store jobs and results.

In-depth tests led the UNICORE Plus project to revise the earlier decision and re-implement the client as an application for the following reasons:

- In contrast to customary believe, Java applets are not browser independent. Special code was required to support implementations of Netscape on different platforms. The developers had to invest substantial effort in supporting new versions of Netscape on new versions of the operating system.

- Signed applets were either recognized as such either by Netscape or by Internet Explorer but not by both browsers. The project was forced to drop Internet Explorer support for MS Windows to meet the development dead lines.
- Certain advanced programming techniques, like Swing classes were not supported.
- Signed applets could not be cached. This became a performance concern especially for mobile users on slower connections, because the client had to be downloaded from the server each time UNICORE was started.

Since the resolution of these issues were beyond the control of the project, we decided to implement the client as a Java application. Installing an application on the client carries effectively the same risk as running a signed applet. The user has to trust the developer and the distributor of the code. The installation process is straight forward using the proven techniques for MS Windows and Unix. The issue of automatic updates of the client software will be resolved in a future release of UNICORE. At start time the user will be informed by the server if a more recent version exists and if the version is a prerequisite to use new features of the system. It will be the user's choice to install the new code or continue to execute the installed client.

To facilitate support for applications and services in the UNICORE client, a plug-in technique has been developed. A developer need to implement only three Java classes to incorporate new applications into the client. At start time, the client automatically detects the existence of a plug-in, makes it available in the pull-down menus of the user interface, and creates default resources if none exist for the application. The application can now be used as a task in a UNICORE jobs. All functions available to the client can be applied to this task. The task was initially demonstrated for CPMD, the Car Parrinello Molecular Dynamics Code (see [15]. Now, additional plug-ins are available, for example for Fluent, Gaussian, or MSC.Nastran. A new plug-in can be developed by an experienced Java programmer within days, using existing plug-ins as templates. In addition, application wizards can be developed which guide the user in creating syntactically and within limits even semantically correct input for complex application. A wizard is part of the CMPD plug-in and led to its great success.

The UNICORE *gateway* is written as a Java 2 application. It communicates with the client over ssl. Now the Sun JSSE is used, replacing the earlier ssl implementation provided by the University of Graz, which was technically superior when the project was started. The Gateway should be run on a dedicated and securely configured system which allows only services required to execute the UNICORE gateway. It may reside behind a firewall, depending on the site's security requirement. A dedicated system is not mandatory but a system configured for optimal security is highly advised.

The *Network Job Supervisor* (NJS) is written in Java 2. It executes on a highly available, sufficiently powerful Unix system. In a production environment the NJS host should be dedicated, although this is, again, not a requirement. One copy of NJS exists per Vsite. All NJS systems may share the same machine or run on different systems for performance. NJS consists of several components performing the following tasks: incarnation, userid mapping, and job synchronization.

NJS uses a target system specific *Incarnation Data Base (IDB)* to translate the abstract jobs and actions contained in the AJO into concrete batch jobs or commands for the target system. The incarnation is a rule based interpretative process which was previously used in another project. The IDB is customized by each installation. The UNICORE administrator defines global information, like the file system used for UNICORE temporary directories, resource information, like minimum or maximum number of processors allowed on the target system, or the names of libraries for an MPI application. Presently, the IDB is a simple editable file.

Migrating to a different representation of the internally used information, like XML, is under discussion and could be completed without affecting the client or the TSI.

The *UNICORE User Data Base (UUDB)* is a mapping of certificates to login accounts (Unix user names) at the target system. Presently there is no business case to replace this ASCII based representation by an LDAP data base, for example. Again this migration would not affect any other part of UNICORE. LDAP will be seriously considered when the vendors move to LDAP as the preferred and fully supported technique to manage user accounts on their systems. This will allow the moving of user administration out of UNICORE and fully integrate it into the normal user management.

Synchronization of jobs and data transfer is done by NJS using the dependency information contained in the AJO.

NJS configuration is performed using the typical configuration information, describing the location of the UUDB and the IDB. Thus the installation gains the flexibility necessary to share information, through a common UUDB for different NJS systems to avoid the double update problem, or to create a separate IDB to test new features.

The *Target System Interface* (TSI) is a daemon running on each host that is part of the UNCIORE Grid. It interacts with the Operating System and the Batch Subsystem. The current versions are implemented as Perl scripts. A C version is used in a project to demonstrate submission of UNICORE jobs to Globus controlled resources. Presently, the TSI runs single threaded. An atomic action, for example the transfer of a job script and the execution of the appropriate qsub command, must be completed before the next action will be accepted from the NJS. Multiple TSI daemons can be started at the target system - up to the number of parallel threads in the NJS - to improve response.

The TSI must be run in privileged mode since it performs actions on behalf of the user and must have access to the user's data with the correct privileges. The switch to the user's identity is performed by using the Unix setuid mechanism for small and well defined sections of the code. A TSI may also run in non-privileged mode under any userid. This is useful when it comes to testing new TSI code where the developer functions as the owner of the designated account. In production mode, this would result in all UNICORE users of a Vsite sharing the same file space, namely that owned by the account under which the TSI is started.

5. Project Status and Future Plans

The first production ready release of the UNICORE software is deployed to all partners of the UNICORE Plus and EUROGRID projects for field tests and as the basis for development of additional functions by the partners. Presently, funding is secured till December 2002 for the UNICORE Plus project and for the EUROGRID project till October 2003. Pallas GmbH, Brühl, is committed to enhance and support UNICORE commercially. The UNICORE specifications, source, and a reference implementation are available to research under Community License. The material is downloadable from the web site of the UNICORE Forum [12], an organization registered in Germany. Its members are project partners, vendors, and other organizations interested in UNICORE.

In addition to becoming the production environment for the German supercomputer centers and their users at German universities, negotiations with other interested parties are under way whether to use UNICORE within intranets of commercial companies.

A UNICORE Test Grid is available (see [11]) to anyone. Any user who wishes to test the function and the graphical interface may download and install the client, create a certificate,

import a certificate into the client by following a simple step-by-step procedure. This process will automatically map the certificate to an account at the test Grid which allows the user to explore UNICORE for a period of 30 days. A full function Grid consisting of three Usites (Tera, Mars, and Saturn) and five Vsites (names after moons of the planets) will be available. They are all simulated on a workstation, thus permitting no production runs and no access to the production systems in Jülich.

Two additional projects are in progress: A feasibly study together with Argonne National Laboratories, to demonstrate that UNICORE can be used to submit jobs to resources controlled by Globus has been completed. A special UNICORE TSI plays the role of a Globus client an interfaces with a Globus Gatekeeper. The work has been presented as part of the SC Global workshop organized by FZ Jülich [19] and demonstrated during SC2001 in Denver.

A project proposal called GRIP[5] (Grid Interoperability Project) has been accepted by the European Commission to be funded for a period of two years starting in January of 2002. The goal is to develop an interoperability layer between UNICORE and Globus to make resources controlled by Globus available to UNICORE clients and those managed by UNICORE accessible to Globus applications. GRIP is intended to exploit the strength of each system and to advance and implement Grid standards within the framework of the Global Grid Forum.

Appendix 1: UNICORE, EUROGRID, and GRIP project partners

The following is a list of the partners in the UNICORE and EUROGRID projects, including their web addresses. These addresses can also be found at the respective project web pages [9] and [10]. The project the partner participate in is indicated by (U), (E), and (G) respectively following their name.

Forschungszentrum Jülich GmbH, Jülich, Germany (U, E, G)
http://www.fz-juelich.de

Pallas GmbH, Brühl, Germany (U, E, G)
http://www.pallas.com

Deutscher Wetterdienst (DWD), Offenbach, Germany (U, E, G)
http://www.dwd.de

Rechenzentrum Universität Stuttgart (RUS), Stuttgart, Germany (U)
http://www.uni-stuttgart.de/Rus

Leibniz Rechenzentrum der Bayerischen Akademie der Wissenschaften (LRZ), Munich, Germany (U)
http://www.lrz.de

Rechenzentrum der Universität Karlsruhe, Karlsruhe, Germany (U)
http://www.uni-karlsruhe.de

Paderborn Center for Parallel Computing (PC²), Paderborn, Germany (U)
http://www.uni-paderborn.de/pc2

Konrad-Zuse-Zentrum für Informationstechnik Berlin (ZIB), Berlin, Germany (U)
http://www.zib.de

[5] GRIP is funded in part by EU grant IST-2001-32257

Zentrum für Höchstleistungsrechnen Universität Dresden (ZHR), Dresden, Germany (U)
http://www.tu-dresden.de/zhr

Fujitsu European Centre for Information Technology (fecit)
Hayes Park Central, UK (U, E, G)
http://www.fecit.co.uk

CNRS IDRIS, Paris, France (E)
http://www.idris.fr/eng/index_eng.htm

CSCS, Manno, Switzerland (E)
http://www.cscs.ch

University of Manchester, Manchester, UK (E, G)
http://www.man.ac.uk

Parallab, Bergen, Norway (E)
http://www.paralab.no

ICM, Warsaw, Poland (E, G)
http://www.icm.edu.pl

EADS CCR, Toulouse, France (E)
http://www.eads-nv.com

T-Systems debis Systemhaus (debis), Stuttgart, Germany (E)
http://debis.de

Argonne National Laboratory, Argonne, USA (G)
http://www.anl.gov

References

[1] I. Foster and C. Kesselmann, ed.
The Grid: Blueprint for a New Computing Infrastructure. Morgan Kaufman Publishers, 1998.

[2] http://www.globalgridforum.org.

[3] http://dast.nlanr.net/Clearinghouse.

[4] http://www.w3c.org/Jigsaw/Doc.

[5] Wissenschaftsrat: Empfehlung zur Versorgung von Wissenschaft und Forschung mit Höchstleistungsrechenkapazität, Wissenschaftsrat, Drs. 2104/95, 7.7.1995.

[6] Wissenschaftsrat: Empfehlung zur künftigen Nutzung von Höchstleistungsrechnern, Drs. 4558/00
http://www.wissenschaftsrat.de/texte/4558-00.pdf.

[7] Hoßfeld, F.; Nagel, W. E.
Verbund der Supercomputer-Zentren in Deutschland – eine Machbarkeitsanalyse
Jülich, 1997, BMBF-Förderkennzeichen O1 IR 602/9
http://www.fz-juelich.de/zam/pt.s/mannheim/vesuz_1999.ps.

[8] Erwin, D., Ed. UNICORE – Uniformes Interface für Computing Ressourcen
(Final report – in German) http://www.unicore.org, 2000.

[9] http://www.fz-juelich.de/unicoreplus.

[10] http://www.eurogrid.org.

[11] http://www.fz-juelich.de/unicore-test.

[12] UNICORE Forum e.V., http://www.unicore.org.

[13] http://www.csse.monash.edu.au/~jon/nimrod.

[14] Romberg, M.
The UNICORE Architecture–Seamless Access to Distributed Ressources
Proceedings of the Eight IEEE International Symposium on High Performance Computing , Redondo Beach, CA, USA, August 1999, Pages 287-293.

[15] http://www.itl.nist.gov/div895/sasg/websubmit/websubmit.html.

[16] http://java.sun.com/products/jlf.

[17] http://www.cert.dfn.de/dfnpca.

[18] Huber, V. Supporting Car-Parrinello Molecular Dynamics Application with UNICORE
Proceedings ICCS 2001, San Francisco, CA, pp 580-567.

[19] Rambadt, M; Wieder, Ph., SC Global, November 14, 2001,
http://www.fz-juelich.de/scglobal/presentations/pres_rambadt.pdf.