

UNICORE: A Grid Computing Environment for Distributed and Parallel Computing

Valentina Huber

Central Institute for Applied Mathematics, Research Center Jülich,
Leo-Brandt-Str, D-52428 Jülich,
Germany,
`v.huber@fz-juelich.de`

Abstract. UNICORE (UNiform Interface to COmputer REsources) provides a seamless and secure access to distributed supercomputer resources. This paper will give an overview of the its architecture, security features, user functions, and mechanisms for the integration of existing applications into UNICORE. Car-Parrinello Molecular Dynamics (CPMD) application is used as an example to demonstrate the capabilities of UNICORE.

1 Introduction

The increasing number of applications using parallel and distributed processing, e.g. planetary weather forecast or molecular dynamics research, require the access to remote high performance computing resources through the Internet. Fig. 1 gives the overview upon the geographical distribution of user groups working on the supercomputer complex of the John von Neumann Institute for Computing (NIC) in Jülich.

On the other hand, one of the today's main difficulties is that the interfaces to supercomputing resources tend to be both complicated and vendor specific. To solve these problems, a project UNICORE [1] was funded in 1997 by the German Ministry for Education and Research (BMBF). The goal of two-years project and of the follow-on project UNICORE Plus [2] is to develop a seamless, intuitive and secure infrastructure that make the supercomputer resources transparently available over the network. Project partners are the German Weather Service (DWD), Research Center Jülich (FZJ), Computer Center of the University of Stuttgart (RUS), Pallas GmbH, Leibniz Computer Center, Munich (LRZ), Computer Center of the University Karlsruhe (RUKA), Paderborn Center for Parallel Computing (PC2), Konrad Zuse Center, Berlin (ZIB), and Center for High Performance Computing at TU Dresden (ZHR). The project is structured in eight sub-projects dealing with software development, quality management, public key infrastructure (PKI), resources modeling, application specific support, data management, job control flow, and meta-computing.

The main idea is to allow users to run jobs on the different platforms and locations without the need to know details of the target operating system, data

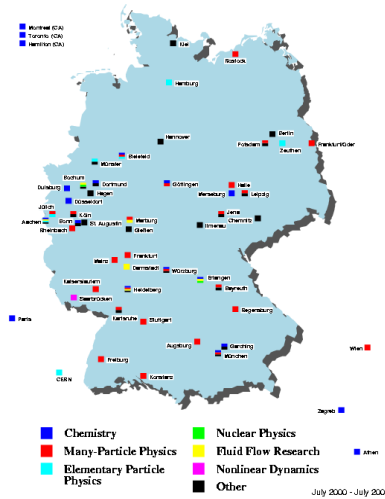


Fig. 1. User Groups of NIC Jülich.

storage techniques, or administrative policies at the supercomputer sites. The graphical interface enables the user to create, submit and control jobs from the local Workstation or PC. UNICORE supports multi-system and multi-site applications for one job. This allows to use the optimal system and resources for the each part of a given problem. In the multi-step jobs the user can specify the dependencies between tasks, e.g. temporal relations or data transfer. Currently, execution of scripts, data transfer directives, and CPMD tasks in the batch mode are supported.

To create a seamless environment, jobs and resources are represented in abstract terms and units. The UNICORE servers translate the *Abstract Job Objects (AJOs)* into platform specific commands and options and schedules the tasks to honor dependencies. The autonomy of sites remains unchanged. The unique UNICORE user identifiers (certificates) will be mapped to local account names (Unix logins).

The developed software is installed at the German HPC centers for the target systems like CRAY T3E, T90, Fujitsu VPP, IBM SP2, Siemens hpcLine.

2 UNICORE System Architecture

UNICORE lets the user prepare or modify structured jobs through a graphical interface, the *UNICORE Client*, a Java-2 application, on a local UNIX Workstation or a Windows PC. The intuitive GUI for batch submission has the same look-and-feel independent of target system and provides the full information about resources to the user. Jobs can be submitted through the *Job Preparation Agent (JPA)* to any platform of a UNICORE Grid, where the user has a local account, and the user can monitor and control the jobs through the *Job Moni-*

tor/Controller (*JMC*). Fig. 2 presents the UNICORE system components and their interaction.

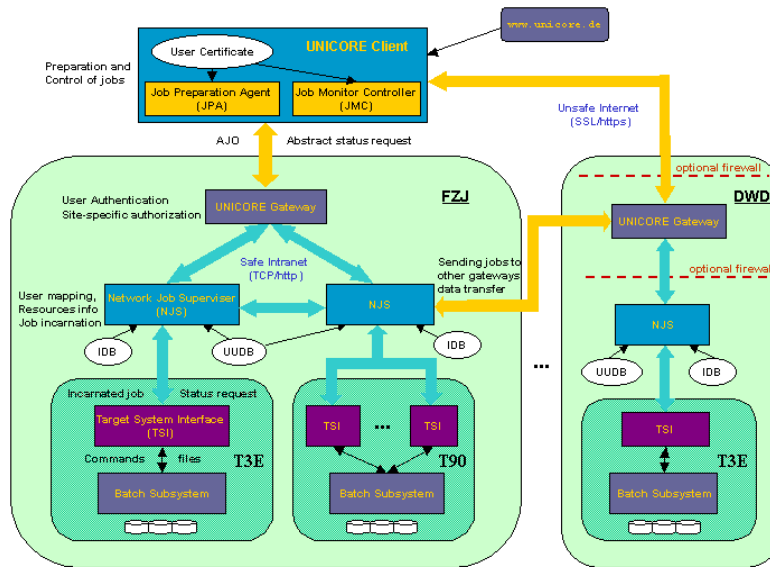


Fig. 2. UNICORE architecture.

The *JPA* constructs a *AJO* with the definition of a job and contacts a *UNICORE Gateway* at a selected site. To support this selection, the *JPA* queries the availability of sites and the addresses of the corresponding gateways from the central UNICORE server (currently at <http://www.unicore.de>).

The Gateway, a small java-application running at the target site, authenticates the user through user's X.509 certificate and provides the user with the information about available resources at the site. It consigns a *AJO* to the appropriated *Network Job Supervisor (NJS)* server.

Each target system or cluster of systems, is controlled by one NJS, also more then one NJS can be installed on a site. The NJS, a Java application, provides the resource information from the *Incarnation Database (IDB)* to the Gateway and checks the authorization of the user to use the requested resources from the *User Database (UUDB)*. It substitutes the site-independent *UNICORE login (Ulogin)*, which is based on a valid user certificate, with the corresponding local *Unix login (Xlogin)* on the destination system. For the target sites, which require additional security, e.g. DCE (Distributed Computing Environment), a *Site-specific Object (SSO)* of the *AJO* will be translated onto the corresponding procedures and commands to provide site-specific additional authentication. The NJS incarnates the abstract tasks destined for a local host into real batch jobs

using the *IDB* and execute them through the *Target System Interface (TSI)* on the batch subsystem. The tasks to be run at a remote site will be passed to a peer Gateway.

The *TSI* is a daemon, a small perl-script, running on the target system, which submit the jobs to the local Batch Subsystem, e.g. NQS, and returns implicit output (stdout, stderr, log-files) from the jobs to the *NJS*, where they are retained for access by the user. Any temporary files, created during running of jobs, are automatically deleted. The Export files (see "Preparation of Jobs"), remain available at the location, specified by the user, on the target system or will be transferred to the local workstation or PC.

A low-level protocol layer between components, called the *UNICORE Protocol Layer (UPL)* provides authentication, SSL communication and transfer of data as byte-streams. The security is based on the Java implementations of SSL and the Java Cryptography Extensions [3] of the Institute for Applied Information Processing and Communications [4] at the Graz University of Technology. A high-level layer (AJO class library) contains classes to define UNICORE jobs, tasks, status and resource requests.

The authentication of users and components (Gateways and NJSes) is based on certificates issued by a UNICORE *Certification Authority (CA)*. It is located at LRZ in Munich and meets the regulations defined by the DFN-PCA (German Research Network - Policy Certification Authority) [5]. The partner centers run a *Registration Authority (RA)*.

3 Application specific GUIs

The general basis for the integration of applications into UNICORE is the usage of the *ExecuteScript Task*. The *ExecuteScript* task contains the definition of a script, the sequence of commands to be executed on the target system, and the list of input and output files for the application. In addition the user can select several execution contexts, e.g. MPI-1, PVM, Debug, Profile, C, Fortran, etc. These contexts are predefined execution environments and are used for example to run parallel programs using MPI. The *UNICORE Client* provides the user with information about the available resources for each task and their limits, e.g. the minimum and the maximum number of processors on the selected machine. The *Transfer Task* is used for the transfer of files from one site to another one.

Furthermore, the users have the possibility to integrate new or already existing application specific GUIs as **plug-ins** into the *UNICORE Client*. The plug-ins are modules that are specifically written to extend the capabilities of the *UNICORE Client*. They use the standard function of the *UNICORE Client* for authentication, security, data transfer, submission and monitoring of jobs, and provide additional support for the applications, e.g. the specification of libraries, the preparation of the configuration files, etc.

Each application plug-in consist of some wrapper classes. The *plugin* class extends the *Job Preparation Menu* of the *UNICORE Client* with the options to add an application task to the job tree and provides the *UNICORE Client* with

the information about other plug-in classes. The *TaskContainer* class constructs the AJO for the particular application task and the *JPAPanel* class presents the application specific GUI. The system plug-ins are located in the *unicore-client/plugin* directory. In addition the user can specify the plug-in directory for own plug-ins in the "User Defaults" dialog. The *UNICORE Client* scans these directories for plug-ins at start-up, loads them, and displays the application specific GUI in the *JPA* by the selecting of the corresponding icon representing an application task.

We selected the widely used Car-Parrinello Molecular Dynamics code [9] as a first application to be integrated in UNICORE. CPMD is an *ab initio* Electronic Structure and Molecular Dynamics program; since 1995 the development is continued at the Max-Planck Institute für Festkörperforschung in Stuttgart [11]. This application uses a large amount of CPU time and disk space and is the ideal candidate for a Grid application. Currently, multi processor versions for IBM Risc and Cray PVP systems and parallel versions for IBM SP2 and Cray T3E are available.

The developed CPMD interface provides the users with an intuitive way to specify the full set of configuration parameters (specification of the input and output data sets, pseudopotentials, etc.) for a CPMD simulation.

4 Preparation of Jobs

Fig. 3 shows the input panel for one CPMD task, in this case a molecular dynamics run. It is divided into four areas: *Properties*, the configuration area for the CPMD calculation, data *Imports* and data *Exports*.

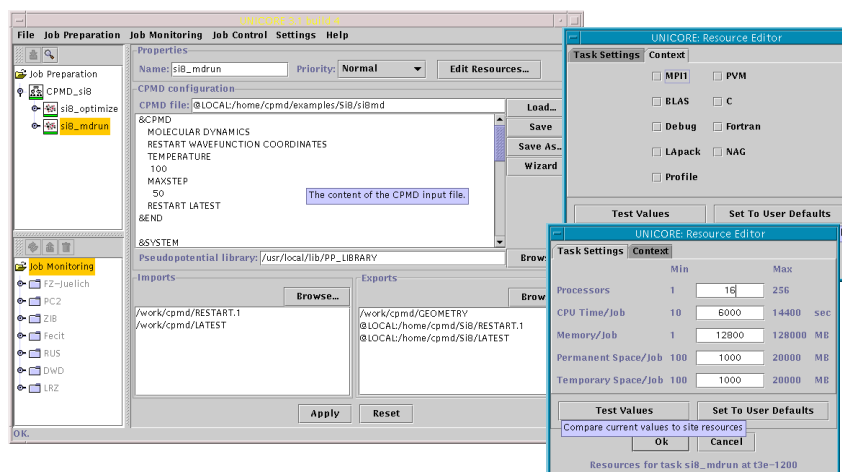


Fig. 3. GUI for the CPMD task.

The *Properties* area contains global settings like the task name, the task's resource requirements and the task's priority. The resource description includes the number of processors, the maximum CPU time, the amount of memory, the required permanent and temporary disk space. The *JPA* knows about the minimum and the maximum values for all resources of the execution system, where the task is to be run, and incorrect values are shown in red.

The configuration area contains the application specific information. It includes the specification of the input file, required for the CPMD program [10]. The button *Generate* brings up the tool *CPMD Wizard*, developed at Research Center Jülich, which generates the CPMD input data automatically. Experienced users may use the data from existing jobs, stored on the local computer. The configuration data may be edited directly or through the Wizard. It is also possible to save data as a text file on the local disk.

For all atomic species, which will be used in the CPMD calculation, the path to the pseudopotential library has to be specified. The local pseudopotential files will be automatically transferred to the target system. Alternatively, the user can specify the remote directory for the pseudopotentials. If this field is empty, then the default library on the destination system will be used.

The *Imports* area describes the set of input files for the CPMD calculation, e.g. a restart file to reuse the simulation results from a previous step. The input files may reside on the local disk or on the target system. Local files marked *@LOCAL* are automatically transferred to the target system and remote files will be imported to the job directory.

The *Exports* area controls the disposition of the result files to be saved after the job completion. In the example some of the output files will be stored on the target system and others, marked *@LOCAL*, will be transferred to the local system and can be visualized there.

Before the CPMD job can be submitted to a particular target system, the interface automatically checks the correctness of the job. Prepared jobs can be stored to be reused in the future.

UNICORE has all the functions to group CPMD tasks and other tasks into jobs. Each task of a job may execute on a different target host of the UNICORE Grid. The job can be resubmitted to a different system by changing the target system. UNICORE controls the execution sequence, honoring dependencies and transfers data between hosts automatically.

Fig. 4 represents an example of a CPMD job consisting of two steps: *si8_optimize* task for the wavefunction optimization of a cluster of 8 Silicon atoms and *si8_md* task for molecular dynamics run. Both tasks will be executed on the same system, T3E in Jülich. The left hand side of the *JPA* represents the hierarchical job structure. The green color of the icons indicates the job as *Ready for submission*. The second task will be run only after the first one is completed. It uses the output files from the *si8_optimize* task to reuse the results of the wavefunction optimization. This dependency is shown on the right hand side and represents a temporal relation between the tasks.

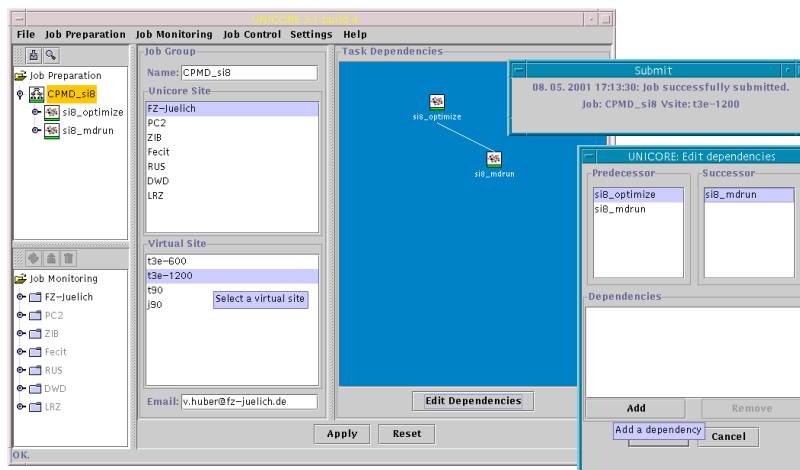


Fig. 4. CPMD job consisting of two tasks and dependency between them.

5 Monitoring of Jobs

The user can monitor and control the submitted jobs using the job monitor part (*JMC*) of the *UNICORE Client*. The *JMC* displays the list of all jobs the user has submitted to a particular system. The job, initially represented by an icon, that can be expanded to show the hierarchical structure. The status of jobs or parts of jobs are given by colors: green - completed successfully, blue - queued, yellow - running, red - completed not successfully, etc. It is possible to delete a job, which has not begun execution, or to terminate running jobs.

After a job or a part of a job is finished, the user can retrieve its output. A completed job retains in the list of jobs until the user removes it.

Fig. 5 presents the status of the jobs submitted to the T3E system in Jülich. The right hand side displays the summary standard output and standard error from two steps *si8_optimize* and *si8_mdrun* of *CPMD_si8* job.

6 Outlook

The technique of allowing independent task to execute simultaneously on different machines and independent child AJOs to execute simultaneously at different sites, supported by UNICORE, provides an alternative to the asynchronous parallelism. In addition, one of the sub-projects aims to extend the capability of UNICORE to allow the metacomputing in the usual sense, which typically requires support of synchronous message passing.

The technique used for the CPMD integration is extensible to numerous other applications. We plan to develop the interfaces for MSC-NASTRAN, FLUENT and STAR-CD applications. These interfaces are going to be integrated into the

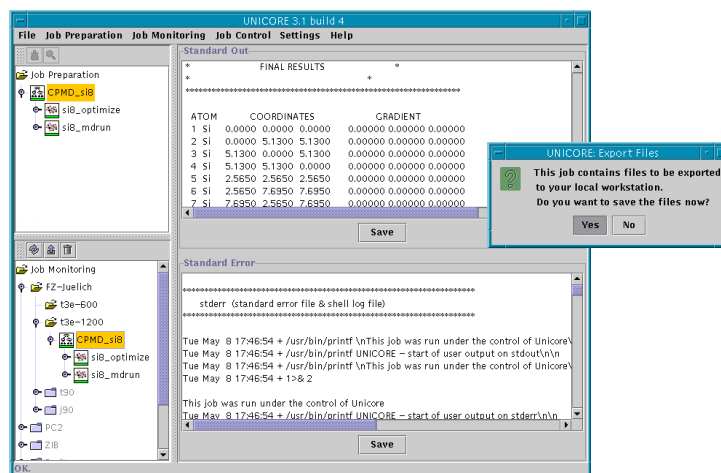


Fig. 5. The Job monitor displays the status of the jobs submitted to a particular system.

UNICORE Client for seamless submission and control of jobs. In the future it is planned to build a generic interface to allow easier integration of applications.

The first production-ready version of the UNICORE system has been deployed for operational use at the German HPC centers.

References

1. The Project UNICORE Web Page.
<http://www.kfa-juelich.de/zam/RD/coop/unicore>
2. The Project UNICORE Plus Web Page.
<http://www.kfa-juelich.de/zam/RD/coop/unicoreplus>
3. Java(TM) Cryptography Extension (JCE).
<http://java.sun.com/jce>
4. Institute for applied Information Processing and Communications.
<http://jcewww.iaik.tu-graz.ac.at>
5. Policy Certification Authority (PCA).
<http://www.cert.dfn.de/eng/dfnpca>
6. J. Almond, D.Snelling: UNICORE: uniform access to supercomputing as an element of electronic commerce. *FGCS* **15** (1999) 539-548
7. J. Almond, D.Snelling: UNICORE: Secure and Uniform access to distributed Resources via World Wide Web. A White Paper.
<http://www.kfa-juelich.de/zam/RD/coop/unicore/whitepaper.ps>
8. Romberg, M.: The UNICORE Grid Infrastructure.
Proceedings of the 42nd Cray User Group Conference (2000)
9. Marx, D., Hutter, J.: Ab *Initio* Molecular Dynamics: Theory and Implementation Modern Methods and Algorithms of Quantum Chemistry. Proceedings of Winter-school, 21-25 February (2000) 329-478

10. Hutter, J.: Car-Parrinello Molecular Dynamics - An Electronic Structure and Molecular Dynamics Program. CPMD Manual (2000)
11. Research Group of Michele Parrinello.
<http://www.mpi-stuttgart.mpg.de/parrinello>