Computing and Informatics, Vol. 21, 2002, 399–411

A UNICORE GLOBUS INTEROPERABILITY LAYER

David Snelling, Sven van den Berghe

Fujitsu Laboratories of Europe
e-mail: snelling@fecit.co.uk, s.vandenberghe@fle.fujitsu.com

Gregor von Laszewski*

Argonne National Laboratory
e-mail: gregor@mcs.anl.gov

Philipp Wieder, Dirk Breuer

Forschungszentrum Juelich GmbH
e-mail: {ph.wieder, d.breuer}@fz-juelich.de

Jon MacLaren

University of Manchester
e-mail: jon.maclaren@man.ac.uk

Denis Nicole

University of Southampton
e-mail: dan@ecs.soton.ac.uk

Hans-Christian Hoppe

Pallas GmbH
e-mail: hoppe@pallas.com

Manuscript received 14 November 2002

* The work of Gregor von Laszewski is supported by the Mathematical, Information, and Computational Science Division subprogram of the Office of Advanced Scientific Computing Research, U.S. Department of Energy, under Contract W-31-109-Eng-38. DARPA, DOE, and NSF supported Globus Toolkit research and development.

Abstract. For several years, UNICORE and Globus have co-existed as approaches to exploiting what has become known as the "Grid". Both offer many services beneficial for creating and using production Grids. A cooperative approach, providing interoperability between Globus and UNICORE, would result in an advanced set of Grid services that gain strength from each other. This paper outlines some of these parallels and differences as they relate to the development of an interoperability layer between UNICORE and Globus. Given the increasing ubiquity of Globus, what emerges is the desire for a hybridised facility that utilises the UNICORE work-flow management of complex, multi-site tasks, but that can run on either UNICORE- or Globus-enabled resources. The technical challenge in achieving this, addressed in this paper, consists of mapping resource descriptions from both grid environments to an abstract format appropriate to work-flow preparation, and then the instantiation of work-flow tasks on the target systems. Other issues such as reconciling disparate security models and file transfer support are also addressed.

Keywords: Grid, Unicore, Globus, interoperability

1 INTRODUCTION

Globus has been described as a Toolkit for the Grid and UNICORE as a grid enabled work-flow environment. This distinction probably best describes the differences in approach. Globus provides a collection of tools and an architecture that constitute an infrastructure for Grid enabled application development. UNICORE on the other hand provides a vertical solution for performing work-flow task management across multiple disparate resources. UNICORE is engineered to allow existing applications and their users to become mobile [2]. This mobility requires management of computational and data resources and their coordination. The UNICORE work-flow services (see below) support the management of the complex, multi-site tasks that arise in this mobile environment.

In addition to EUROGRID [3], the European Commission has recently funded the Grid Interoperability Project,[1] to investigate the issues arising from interoperability between Grids. This paper outlines some of the parallels and differences as they relate to the development of an interoperability layer between UNICORE and Globus. Given the established and desirable high-level workflow GUI that UNICORE provides, and the pervasiveness of the Globus Toolkit on high-end compute resources, it is natural to hybridise the two by fitting the high-level UNICORE functionality over the lower-level components of the Globus Toolkit.

[1] GRid Interoperability Project, IST-20001-32257.

2 BACKGROUND

In this section we discuss those aspects of both Globus and UNICORE that are relevant to the development of an interoperability layer between the two.

2.1 UNICORE

The mechanisms and architecture of UNICORE have been presented elsewhere [2, 13]. The reference implementation (Open Source available [13]) is written in Java. Here we highlight some of those features that provide a work-flow environment for application end-users.

The UNICORE resource model, contained in the Incarnation Data Base (IDB), specifies the usual resources of a computer facility such as processing, memory, and disk, but also software resources available at the site. These resources are stored and managed as Java objects and described in a simple text language. In addition to providing specifications of the resources, the IDB includes re-write rules for how abstractions of resources and tasks in the Abstract Job Object (AJO) are to be "incarnated" into their system and site specific forms.

Using the information in the resource set for each site, the user constructs a work-flow style description of the tasks to be performed on a collection of sites. This description is in the form of an AJO. The AJO, as its name implies, is abstract and does not include site or system specific information, e.g. paths to executables or flags for compilers, etc. Once an AJO has been constructed, its contents are signed by the user using her X.509 certificate and *consigned* to a UNICORE site (Usite) for processing. The target Vsite within this Usite is called the *primary* Vsite.[2]

At the primary Vsite, AJO processing includes user mapping, authentication, and authorization; task incarnation (for tasks, including file transfer, to be performed at this Vsite); and consignment of sub-AJOs to other Vsites at this or other Usites. Note that secondary Usites do not need to trust the primary Usite, each sub-AJO was signed by the user prior to consignment to the primary Vsite.

Complex work-flow structures are possible in this framework, including the following examples:

1. Data pre-processing at site A, simulation at site B, and post-processing at site C.
2. Iterative re-execution of a simulation until a termination criteria is met. The simulation may be a complex job, such as (1) above.
3. Simple application steering, in which a job "holds" itself at some point and waits for the user to check intermediate results before releasing the job.
4. More complex application steering, in which jobs are monitored and steered during execution. This may include visualization.

[2] A UNICORE site (Usite) has one security gateway and may have several Virtual sites (Vsites) with in it. A Vsite is a collection of (one or more) systems and associated resources sharing a file space and user administration domain.

5. Conditional execution of tasks, or whole sub jobs, depending on the results of other tasks.
6. Ensemble simulation of many cases at different sites and subsequent collation of results centrally.
7. Simple meta-computing jobs.[3]

The next section discusses some features of Globus that are not provided by UNICORE and therefore represent additional motivation for integration.

2.2 Globus

Ian Foster and Carl Kesselman described the Globus project in it's earliest stages as:

> The Globus project is attacking the meta-computing software problem from the bottom up, by developing basic mechanisms that can be used to implement a variety of higher-level services. [4]

This is clearly a contrast to the top down approach taken by the UNICORE system. The resulting Toolkit provides the potential of an almost universal level of functionality, compared to the narrower focus of the UNICORE environment.

The Globus Toolkit provides many important Grid components and services, including a security infrastructure (GSI), Metacomputing Directory Service (MDS), secure third-party file transfer, and a Resource Description Language (RSL). Globus Job-managers can be configured for all popular batch subsystems, including PBS, LSF and Condor. Through MPICH-G2, a communications infrastructure for meta-computing jobs is also available.

In particular, there are two components of the Globus Toolkit that provide real strength, that contrast the UNICORE approach, and therefore form part of the motivation for the integration of Globus and UNICORE. These are the CoG (Commodity Grids) Kits [10], which provide an application level API, and the way in which the MDS services support a dynamically changing grid.

The Globus CoG Kits provide an interface to the application developer that allows extensive use of the Grid infrastructure. These components can be used to simplify the construction of complex jobs in RSL, and can be used to launch jobs (via GRAM and implicitly via DUROC through full delegation) which work in a tightly coupled meta-computing environment. With UNICORE there was a deliberate intension not to support tightly coupled meta-computing or application level development explicitly. The target users had existing applications that did not use Metacomputing environments. These applications were frequently provided by third parties.

The Metacomputing Directory Service (MDS) provided by Globus is based on the assumption that the computational resources on a Grid are dynamic in nature.

[3] UNICORE provides abstractions for resource and coarse grain job synchronization (e.g. hold jobs until a given time).

There is a significant level of intellectual sophistication required in the design of such information services. Until recently, the UNICORE approach has assumed a static grid. Insights gained through the integration of UNICORE with the MDS will provide pointers to adapt UNICORE for use in dynamic environments, including the creation of dynamic Virtual Organizations (VOs).

The overall combination of Globus ubiquity and advanced services with the work-flow support of UNICORE provides a significant motivation for a combined solution. However, when attempting such a merger, careful analysis of the interfaces, where the two approaches overlap or provide redundant or competing solutions and where they differ, is necessary to the completion of such a project. The following section outlines how these interfaces interact to create an architecture for such a combined solution.

3 A WORK-FLOW PORTAL FOR GLOBUS

Both UNICORE and Globus provide the conceptual cycle of resource discovery, job creation with a resource request, job adaptation for site specific details, job execution, and job monitoring and control. The names and mechanisms differ significantly, but abstractly these ideas remain constant.

In the following sections an overall architecture for this hybridisation is presented, along with an analysis of the significant interfaces that must be adapted or extended. In particular, security models, resource mapping, job mapping, and file transfer are considered.

3.1 Architecture

Figure 1 outlines the basic architecture of integration of Globus facilities into UNICORE. The UNICORE Client, Gateway, and Network Job Supervisor (NJS) remain largely unchanged. A new plug-in to the client[4] is needed to perform the grid-proxy-init function (as needed by the Globus architecture) and create the temporary proxy certificate, which can be included in the AJO as a Site Specific Security Object (already part of the standard UNICORE protocol). With this exception, the AJO is the same as would be sent to any UNICORE Usite.

Because Globus already presents a unified view of its sources (including multiple distributed hosts), it is logical to model it as a single Usite, although this is not mandatory. Thus a Globus Usite represents a Virtual Organization [6] and obtains its view of Globus resources through a GIIS. Within the Globus Usite, each major Globus resource (usually a single host) is modelled as a Vsite. At the entry to the Globus Usite, the unchanged UNICORE Gateway will authenticate the connection using a Globus or UNICORE issued certificate. The AJO is then dispatched to the

[4] The flexible plug-in interface, which facilitates the rapid development of application specific GUIs, is a significant UNICORE strength.

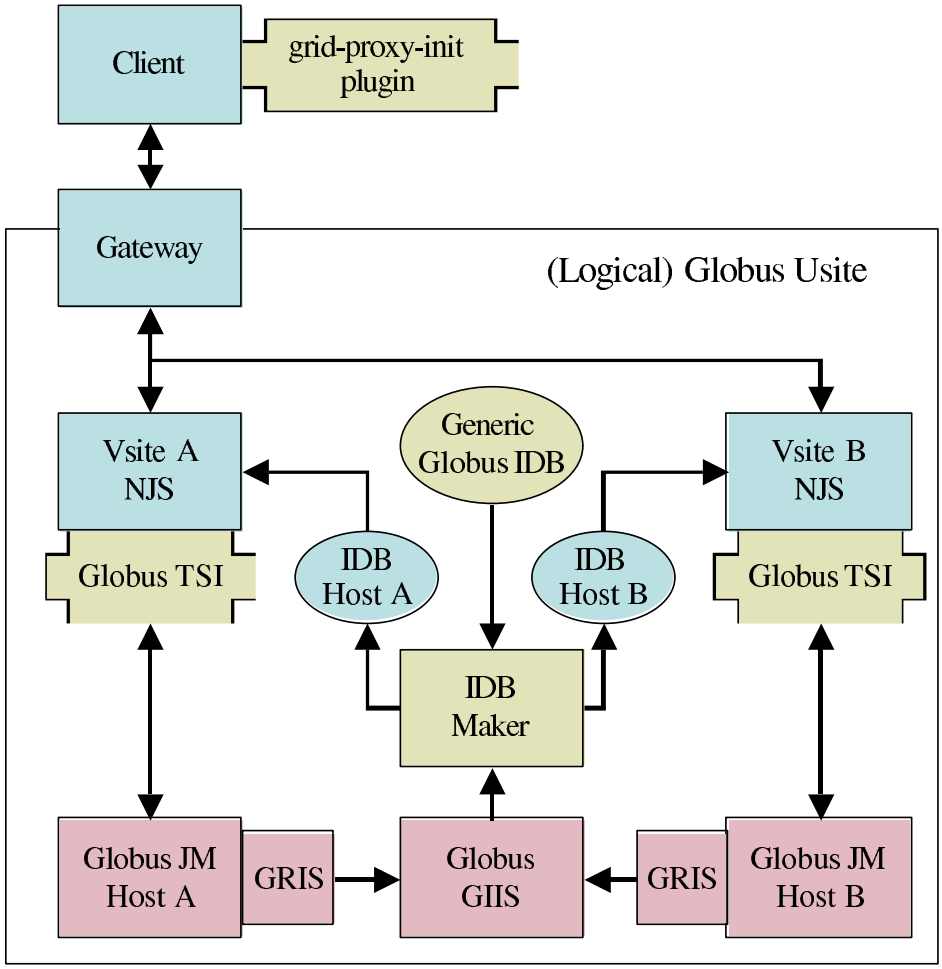


Fig. 1. Overall architecture for Globus and UNICORE interoperation

NJS depending on which Vsite was selected by the user.[5] See later discussion on security.

The NJS also remains largely unchanged. Within the NJS, information from the host IDB is used to incarnate the job into a script. Construction of the script can be achieved with the help of the appropriate packages within the Java CoG Kit. The incarnated job is passed to a largely new component, the Globus Target System Interface (TSI). In existing UNICORE sites the TSI is adapted to the peculiarities

[5] There are plans to incorporate dynamic resource brokering into the UNICORE system.

of batch subsystems on the various computational platforms. Traditionally, the TSI has been written in Perl, however, a Java based implementation is also being considered for this project. In particular, this would allow the TSI to utilise the CoG Kit.

The creation of the host IDBs is a major part of this strategy. Based on a Generic Globus IDB, and information obtained from the GIIS (or perhaps directly from the host's GRIS), a Globus Vsite specific IDB will be produced. These IDBs are created automatically by the IDB Maker. The generic Globus IDB lists all those fields whose values must be provided by the GIIS (using GRIP via Java CoG[6]) in order to allow the successful incarnation of UNICORE jobs by the NJS. Where this information cannot be obtained automatically, it may have to be provided manually.[7] Note that because UNICORE's abstraction based approach limits what can be incarnated to those concepts for which abstractions have been created, the scope of this task is not as boundless as it appears at first. This interoperability problem will be investigated in future activities with the goal to achieve a unified standard that is promoted through the Global Grid Forum.

3.2 Security

Both UNICORE and Globus base their user authentication model on X.509 certificates. Therefore, it is possible to use a Globus user certificate to run UNICORE jobs as long as the servers are configured to accept the Certification Authority which issued that certificate. The reverse should also be possible, i.e. use a UNICORE certificate to perform a 'grid-proxy-init'. However, significant differences in the overall security models exit. In the Globus model, a temporary certificate is created by the 'grid-proxy-init' function and used as a proxy for a series of delegated actions within the Globus environment. This proxy is protected by standard Unix mechanisms at the individual sites, which is only a problem if these are compromised at the site. However, for one site to accept a proxy, which has come to it via another site, it must trust that site as well as the end user, i.e. "transitive-trust" is required. The UNICORE model only requires the site to trust the end user but not any Usites en route.

3.3 Resource and Job Mapping

The UNICORE resource package is used for both advertising what resources are available at the Vsite and to describe a user's resource request. Within the UNICORE resource object model, resources fall into three main subclasses:

[6] In [6], changes in the protocols of Globus may be under consideration.

[7] UNICORE makes extensive use of application meta-data which is not always provided by site's MDS services. The Grid Interoperability Project will make use of any emerging standards in this area, e.g. RIB [11].

- **Capability Resources**: These represent services that are provided by the Vsite. Examples include application software, parallel libraries, and job priority levels.
- **Capacity Resources**: As their name implies, these resources all have a concept of capacity associated with them, e.g. memory, disk space, nodes, processors, etc.
- **Information Resources**: These resources describe other aspects of a Vsite that are not essential to the normal function of UNICORE, e.g. contact address for site administration, system architecture, etc.

Although possessing a different structure, the Globus resource model, as described by the MDS, can be mapped to these three categories. It is not necessary that this mapping be isomorphic, as for this project the information flow is unidirectional. Globus resources are converted to UNICORE resources, used to define a user's resource request, and then incarnated to run on a Globus Host. There is no requirement to describe all UNICORE resources in MDS or to use all available MDS information in creating and incarnating a UNICORE job.

There is detail in the MDS resource description (e.g. virtual memory size, current system load[8]) that is not used by UNICORE and can be converted to information resources as a courtesy to users. At incarnation time, there is a system of default values defined for all UNICORE resources that may be used in the absence of specific requests from the user. These defaults allow the incarnation of UNICORE jobs, even where there are resources, required by UNICORE, that are not published in the MDS, e.g. execution priority.

The combination of resource information from the GIIS, the defaults included in the host IDBs, and static mappings from UNICORE abstractions (run script, cancel job) to corresponding the Globus functions, should provide all the infrastructure to complete the Grid Interoperability Protocol intended in this project.

3.4 File Transfer and Management

To implement third party transfer, the UNICORE architecture has been extended by an Alternate File Transfer (AFT) component. In the prototype implementation a GridFTP client and server are installed on the target system together with the TSI. If GridFTP is not supported at a Vsite the file transfer will be performed using the UNICORE protocol automatically. The security for AFT is based on the same security mechanism as for UNICORE jobs.

There are two main functions within the UNICORE file management architecture:

1. Importing/Exporting files between the Uspace (the work space of a UNICORE job) and the user's workstation, home file systems, and storage archives.

[8] As part of EuroGrid [3] a resource broker is being developed to work in the UNICORE environment. This facility will be able to take advantage of dynamic information from MDS.

2. Transferring files between Uspaces.

For example, a file may be imported from a user's archive to Uspace, pre-processed and then transferred to the Uspace at another site.

The transfer of files in UNICORE between Uspaces is based on the UNICORE protocol layer (UPL). An AJO is passed between the two Vsites followed by a byte stream. The same communication and data path is used for the transfer of files as for the transfer of sub-jobs between different Vsites. In practice this means that the data will be send from Vsite1 via the remote Gateway to Vsite2. This ensures both a secure transfer and the correct mapping of the user's certificates to the correct account at the target system.

A prototype implementation of an alternate file transfer mechanism in UNICORE with the integration of GridFTP [8] which is part of the Globus Toolkit has been done. This allows to exploit the functions of GridFTP, like parallel streams and restart of interrupted transfers, without additional development effort. In a pure UNICORE environment, files can be transferred directly from the Uspace at Vsite1 to a Uspace at Vsite2 as depicted in Figure 2. The communication path used for an alternate file transfer is established between AFT and remote Gateway and the data path is directed from a target system to the remote target system. Note that one of the Vsites can also be a Globus site.

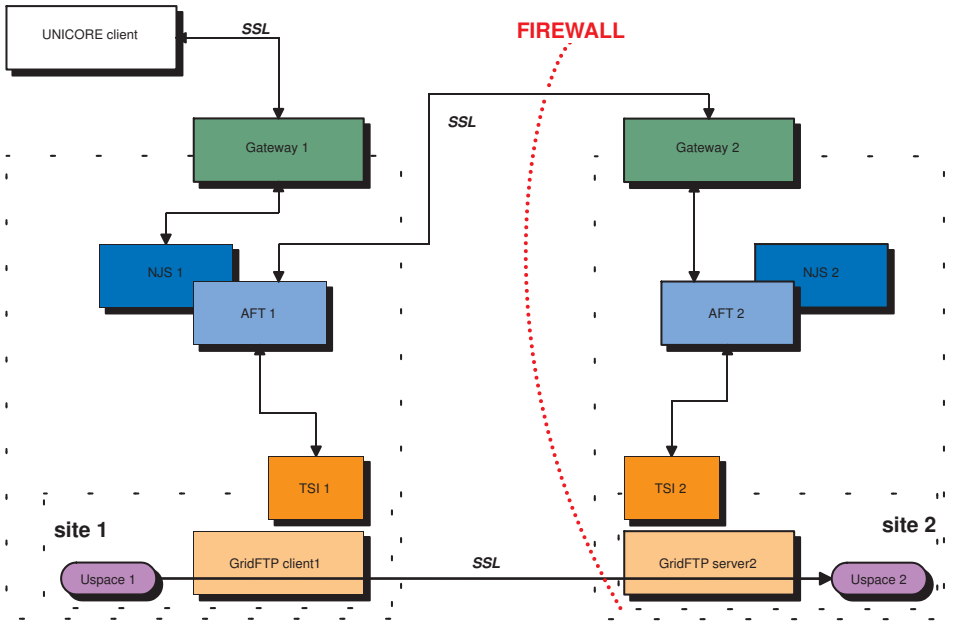


Fig. 2. GridFTP based file transfer from Uspace1 to a remote Uspace2 in a UNICORE environment

4 CONCLUSION AND FUTURE DIRECTIONS

The success of this project has formed the basis for further work in the area of interoperability between UNICORE and Globus, in particular in the area of security. This project demonstrated that an interoperation between these two security mechanisms was possible. Recent developments outside the scope of this paper have demonstrated that UNICORE can function with Globus certificates and that a UNICORE issued certificate can be used to create Globus proxy certificates. Therefore, the differences between these to Grids with respect to security resolve to issues of policy alone.

On a wider scale, both projects are working on the Open Grid Services Architecture [6] within the GGF to develop a much more extensive framework for interoperability. Like this project, this continuing work is based on interface design backed up by working implementations. Thus, the expectation is that Grid interoperability between UNICORE and Globus will be the norm rather than the exception.

REFERENCES

[1] Czajkowski, S.—Frtzgerald, I. F.—Kesselman, C.: Grid Information Services for Resource Sharing. In HPDC-10, IEEE Press, 2001.

[2] Erwin, D.—Snelling, D.: UNICORE: A Grid Computing Environment. LNCS 2150, Euro-Par 2001, Springer, 2001.

[3] EuroGrid project. IST-1999-20247, www.euorgrid.org.

[4] Foster I.—Kesselman, C.: Globus: A Metacomputing Infrastructure Toolkit. Intl J. Supercomputer Applications, Vol. 11, 1997, No. 2.

[5] Foster I.—Kesselman, C. ed.: The Grid: Blueprint for a New Computing Infrastructure. Morgan Kaufman Publishers, 1998.

[6] Foster, I—Kesselman, C.—Nick, J.—Tuecke, S.: The Physiology of the Grid. Draft 2/3/2002, www.globus.org/research/papers/ogsa.pdf.

[7] Global Grid Forum, www.gridforum.org.

[8] Globus project — GridFTP, www.globus.org/datagrid/gridftp.html.

[9] Bivens, H.: Workflow drafts: www.sass3186.sandia.gov/ hpbiven.

[10] Laszewski, G.—Foster, I.—Gawor, J.—Lane, P.: A Java Commodity Grid Kit. In CCPE, Vol. 13, Issue 8–9, 2001.

[11] Repository in a Box Project, www.nhse.org/RIB/.

[12] TENT Project, www.sistec.dlr.de/tent/.

[13] UNICORE Information is Available on the Following Web Sites: www.unicore.org — UNICORE Protocols and Source Code; www.unicore.de — The home page for the UNICORE Plus project; www.fz-juelich.de/unicore-test — A complete UNICORE environment where users can download a client, obtain a certificate, and trial the UNICORE environment.

[14] XML Activity Log www.w3.org/XML/Activity.

David Snelling obtained a BS and MS in Mathematics from the University of Denver and a PhD. in Computer Science from the University of Manchester. He has worked in parallel and high performance computing in industry (Denelcor in the 80's and Fujitsu in the late 90's). He spent about 10 years teaching and researching at the Universities of Leicester and Manchester.

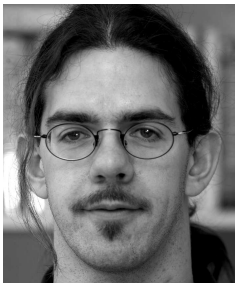
Dr. Snelling was involved in the Unicore project from it's inception and has worked on both the technical design of Unicore, within Unicore related projects, and on the licensing requirements for the Open Source strategy adopted by the Unicore Forum and the software developers. Within Fujitsu he has acted as project manager for the implementation of the server side components of the Unicore reference implementation. More recently, he has been appointed as co-chair of the OGSI Working Group in the GGF and is an enthusiastic supporter of, and contributor to, the Open Grid Services Architecture.

Sven van den Berghe has been working on Grid Computing at Fujitsu Laboratories of Europe for 6 years. During this time he has been designing and developing Unicore, with particular emphasis on the server components.

His previous work involved the application of supercomputing in weather forecasting and reservoir simulation.

Gregor von Laszewski is an Assistant Computer Scientists at Argonne National Laboratory and a Fellow of the Computation Institute at University of Chicago. He received a Master's in computer science at Bonn University, Germany and a Ph.D. from Syracuse University, U.S.A. His current research interests are in the areas of parallel, distributed, and Grid computing. Specifically, he is working on topics in the area of using commodity technologies within Grid services, applications, and portals. He is the principal investigator of the Java CoG Kit providing access to Grid services in pure Java through Globus protocols.

Philipp Wieder received his diploma in electrical engineering from RWTH Aachen, Germany. Since 2000 he is working at the Research Centre Jűlich as a research scientist, mainly focussed on job scheduling, resource management and Grid computing. He is involved in a few European Grid projects as a software architect and he enjoys object-oriented programming.



Dirk Breuer received the diploma in computer science from the RWTH Aachen in Germany 2001. Since February 2002 he is a research scientist at the Research Centre Jűlich in the Department of Operating Systems in the Central Institute of Applied Mathematics. He works in the field of Grid computing, especially security and file transfer of the EUROGRID project.

Jon MacLaren is currently a Software Engineer in the EUROGRID project, where he works on developing Resource Brokering technology for the UNICORE middleware. He is active in the growing UK e-Science community through Manchester's regional e-Science centre, E-Science NorthWest (ESNW). He is also active in the Global Grid Forum, where he co-chairs two groups: the first of these is developing a standard protocol for Advance Reservation (GRAAP-WG); the second is concerned with protocols for enabling the buying and selling of computational services (GESA-WG). He received his PhD from Manchester University in 2001, for his thesis on techniques for the semi-automatic parallelisation of Fortran programs. He received his MPhil, also in Parallel Computing, from Manchester in 1997. He completed his first degree, a BSc in mathematics and computer science, at York University, UK in 1993, and worked in industry until recommencing his studies in 1996.

Hans-Christian Hoppe received his diploma in computer science from the University of Bonn. Between 1989–1994 he had been with GMD, working on communication libraries for shared- and distributed-memory computers (PARMACS). Since 1994 he has been with Pallas GmbH, working in the design and implementation of message-passing libraries (MPI), and in the development of parallel performance tools and libraries (Vampir, Vampirtrace), as well as in the design and development of GRID portal systems. He is currently co-ordinating Pallas' UNICORE and EUROGRID efforts, and is representing Pallas' Grid activities in the GGF and in the EU Health-Grid and the NSF Grid Middleware initiative (MAGIC) activities.