Final Report

DFN-Project

“GRIDWELTEN: User Requirements and Environments for GRID-Computing”

5/30/2003

Peggy Lindner[1], Thomas Beisel[1], Michael M. Resch[1], Toshiyuki Imamura[2], Roger Menday[3], Philipp Wieder[3] Dietmar Erwin[3]

[1] High Performance Computing Center Stuttgart (HLRS), Germany
[2] Japan Atomic Energy Research Institute (JAERI), Japan and High Performance Computing Center Stuttgart (HLRS), Germany
[3] John von Neumann Institute for Computing (NIC), Germany

GRIDWELTEN

Abstract

The GRIDWELTEN project aims to evaluate and make recommendations regarding the use of high performance computing resources accessible through Grid software and architectures. The metric for this comparison will be the expectations and requirements of the user situated at the sites of the project partners and at other HPC-centers throughout Germany.

Typical user groups will come from many areas of science, and we must be careful to incorporate the expectations of each into a set of common requirements. With this in mind, a survey of the end users was conducted. The findings of other relevant research in the area of Grid usability will also be discussed.

Select Grid middleware and supporting software will be reviewed. In particular, we match and consolidate between the end user expectations and the features of some popular Grid infrastructures. For this, we consider the four Grid systems, Globus, UNICORE, AVAKI and TME.

In the report we also examine the implications of the Open Grid Services Architecture (OGSA) proposed by the Global Grid Forum, on the operation and future interoperability of the evaluated systems.

The final project documentation will be presented in three main sections: firstly, a summary of user requirements for Grid computing, secondly, an evaluation of selected Grid computing environments, and thirdly a detailed analysis.

The GRIDWELTEN project is commissioned by the German DFN (Deutsches Forschungsnetz).

Table of contents

Purpose & Motivation *4*

Introduction *5*
Types of Grid 6
Architecture of a Grid 7
Business interest in Grid Computing 10

User Requirements *11*
Introduction 11
Survey 11
Analysis of Results 12

Tools *23*
Introduction 23
Globus 23
UNICORE 26
AVAKI 31
TME 34

Installation and Test of Tools *39*
Introduction 39
Summary of features available within the tested tools 46

Summary and Conclusions *48*
Introduction 48
Summary of Systems 49
Conclusions from the Questionnaire 52
Standardization and Interoperability 52
Executive Summary 54

Appendices *56*
A Bibliography and References 56
B Terminology 59
C Questionnaire 60

Purpose & Motivation

In defining a starting point, the project faced two challenges which defined the methodology taken for the whole project.

- Despite the publicity, different users have a different understanding of the meaning of Grid computing and how they could use it to their benefit.
- Grid computing is a rapidly developing field.

Grid Computing is seen as a means to provide access to distributed compute resources, instruments and databases as well as know-how. We must gain an understanding of the user community in order to get a precise picture of the requirements of the people and groups working in the field. The reader should always be aware that this report refers to German users, although experiences from other countries are integrated due to affiliations and collaborations of the team of authors.

For this project, we evaluate the user's *understanding*, *requirements* and *expectations* regarding Grid computing. We achieve this through the distribution of a questionnaire, and we provide an analysis and summary of the results.

An evaluation of Grid software is also included. It is important that we are mindful of the user requirements as we proceed with this evaluation phase. Given the limited amount of time and resources, the evaluation of software environments is focused on four packages:

• GLOBUS	Argonne National Laboratories, USA
• AVAKI	AVAKI Corporation, USA
• TME	Japan Atomic Energy Research Center, Japan
• UNICORE	UNICORE Forum e.V., Germany

By choosing these we believe we have selected the most important environments in the field from the US, Japan and Europe. This should allow to get a general and thorough picture of the available functionality and potential in the field. In addition to an in-depth evaluation of these four packages we will also consider further environments. However, no further evaluation in terms of practical work will be done in this report. Wherever possible, practical experience of other groups and users both from research and industry will be included in this report.

Introduction

The evolution of the Grid within the research community initially came from a ‘globalization’ of scientific practices and collaborations. This included:

- Information exchange within globally dispersed groups.
- Accessing and aggregating remote computing resources.
- Storing of data in large distributed data stores.

This led to the definition of the Grid, as “resource sharing and coordinated problem solving in dynamic, multi-institutional virtual organizations (VO’s).” [1]

Within the research community and business, there is a demand for increasingly large, complex and resource-intensive applications. It is a common observation that single computational resources often fail to meet the demand of those applications. However, wide area networking improvements make it possible to aggregate distributed resources in various collaborating virtual organizations and to form a Computational Grid.

Whether the user accesses a single or multiple resources (aggregated as a coordinated ‘virtual supercomputer’), the Grid allows users to interact with the resources in a uniform way, providing a comprehensive platform for distributed computing.

From the end-users perspective, it is very important to provide an easy to use interface to this underlying infrastructure. Even if the basic middleware and Grid technologies that support the infrastructure are complex and sometimes not yet fully mature, the fundamental idea is to present to the end user a virtual computing environment with an uniform way to use a collection of distributed resources.

Grid computing implies more than just cycle scavenging on underused workstations or aggregating super-computers. Rather, Grid computing is the sum of these use-cases, and others. The wide ambitions for Grid computing include new environments, working practices and resource utilization.

However, at present very many of the people experimenting and using Grid technologies are based at computer centers, and have some background in high performance computing. Thus, the people invited to take the survey mainly come from such institutions and are either users or developers of applications running on large scale computer infrastructures and have a background in distributed and parallel computing.

The relationship between High Performance Computing and Grid technologies can be viewed from two angles, as reported in [5] “some eminent players believe that distributed ubiquitous computing resources will ultimately dispose of large HPC facilities. However, on the contrary, it can also be argued that Grid technologies will enhance their effectiveness by providing seamless access, interactivity and user friendliness to distributed clusters of powerful, tightly coupled systems”.

For the formulation of the questionnaire, the following questions are typical of those arising that we felt should be evaluated from the perspective of the expectations of users keen to exploit a Grid, and administrators keen to deploy a Grid.

- Is the Grid available for a computer centre to handle all its resources?
- Regarding future expandability, how open is the system?
- For coupled simulations, is there support for dependencies, workflow models, etc?

- For normal applications, how is the support for applications on different hosts at the same time? Is there any scheduling support?
- Are parameter sweep studies supported?
- Taking the following application types; 1. Parameter sweeps, 2. Coupled applications, 3. Normal applications, how is the transfer of files handled?
- Is there a distributed file system? Is a resource automatically added to the Global file system? If so, is there a Global view of the distributed files?
- How do you use the system?
- Security. Which access mechanism is used? Does the system support different usernames on different machines? How does the system interface with the security mechanism in place at the local sites?
- Is there any guarantee of integrity of the resources available? How does this work?
- How might one administrate the economic charging for resources consumed on the Grid?
- What is the relationship between High Performance Computing (HPC) and the Grid?

These themes and some others provided the subjects to be included in the survey, and helped to formulate the questions of the questionnaire itself.

Regarding similar work of this project, we refer the reader to the Enacts project, especially the report entitled 'Grid Service Requirements' [7].

Types of Grid

A recent analysis of Grids states [17] "Initially it was thought that the Grid would be the most useful in extending parallel computing paradigms from tightly coupled clusters to geographically distributed systems. However, in practice, the Grid has been utilized more as a platform for the integration of loosely coupled applications - each component of which might be running in parallel on a low-latency parallel machine - and for linking disparate resources (storage, computation, visualization, instruments)". This is interesting. It shows that the general perception of the scope for Grid computing has widened. Currently, Grid technology is being applied, or there is at least discussion of it being applied, to solve many different problems or to enable completely new scenarios.

We can summarize some of the current thinking of what types of Grid are possible and what can be achieved over such infrastructures. To agree on some common usage patterns [56] now will be useful later when we discuss the findings of the survey

- <u>Distributed supercomputing</u>
  This means the coupling of multiple potentially distributed computing resources. Normally, such an infrastructure is composed of HPC resources. Often there is a dedicated network between contributing computer centers. Consequently, as the environment is considered to be constructed of 'friendly' components the security is not as high a priority as with other Grid environments.
- <u>High-Throughput Computing</u>
  This is the acquisition of available and accessible computing resource to solve collectively many independent parts of a large scale problem. Such resources are often many underused desktop computers, and this has led to the expression 'cycle scavenging'. Such an approach was popularized by the SETI@home [36] project. Other more general software exists, for example, the Condor [41] and Entropia [37] software. High throughput computing also encompasses the division and scheduling of parameter space type problems over a number of the under-used desktop resources.

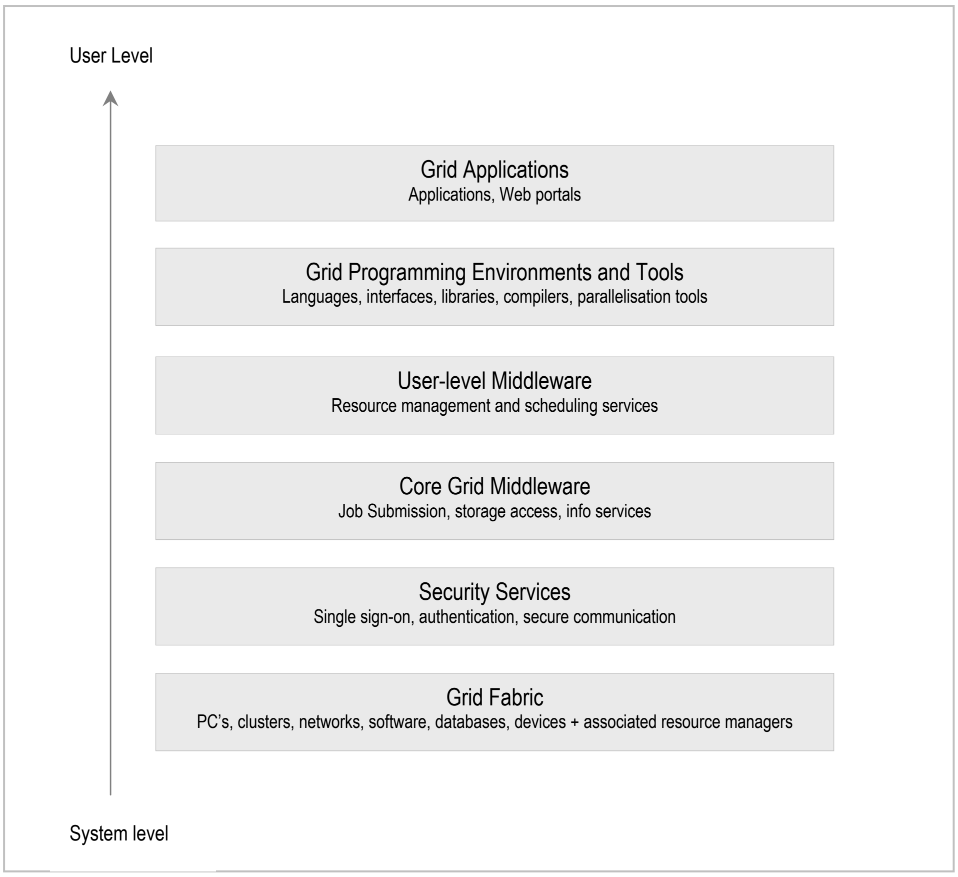
- Desktop Supercomputing
  This model of usage refers to making super-computers easily available to more users, i.e. seamless remote access, from the desktop, and without requiring in-depth knowledge of system configuration in order to run a job on the resource. This classification does not imply and should not be confused with the usage model of 'cycle scavenging', which is the aggregating of unused resources of many desktop machines together.
- Data-Intensive Computing
  This is the integration of data from multiple, distributed file systems, to form a large distributed data storage. This enables numerical simulations or the gathering and storage of experimental results at a massive scale.
- Collaborative Environments
  This is enabling and enhancing human interactions to solve a single, tightly coupled problem. The nature of a virtual organization often includes the coordination of human resources. Collaborative environments allow participants at multiple sites to engage in the control and observation of a Grid-wide activity.
- On-Demand Computing
  This is providing access to advanced capabilities to fulfill a short term demand. This mode of operation has an obvious parallel with the electric power grid, where power stations on the Grid are kept with the express purpose of generating power on demand, and at short notice.

Architecture of a Grid

A good starting point for discussing Grid architecture is the 'hourglass model' [20]. This model summarizes many of the core concepts common to all Grid architectures. The width of the base of the hourglass illustrates the number and variety of the physical resources which comprise the Grid. There is a thin neck of the hourglass, which represents the integration layer of the Grid middleware. This is the homogeneous layer of services on top of the heterogeneous resources which make up the Grid. The width of the top of the hourglass represents the number and functionality of the user-level (or higher level) services, applications, etc.

There are many viewpoints on the finer architectural description of a Grid, for example [13], [], [24], [38]. We take a commonly used description of the architecture, as described in [38], and adapt it a little. This particular model reflects our view of the Grid from a user perspective as taken in this report.

Therefore, in the next section, we shall use this model as a basis when we introduce the whole architecture which will make up a Grid enabled infrastructure. Figure 1 shows the various layers which make up the Grid infrastructure. At the system level we shall have the Grid fabric (the physical resources which make up the Grid) and the types of local services, such as resource managers, which are needed. Next, the security infrastructure which is essential for safe operation on the Grid. Then the core and higher level Grid services [31], which must be provided for the Grid to be useful. Finally, there are the Grid programming environments and applications, assisting the ease of use of the Grid. In practice we can only speak generally of the functions of the different layers, as there is some 'blurring' between the layers in all current implementations. We have applied the categorization that Grid Applications tend to have some kind of visual interface and have some characteristic of remote access, whereas a programming environment for a Grid lacks a visual interface. Sometimes a particular piece of software has components in both layers.


User Level
Grid Applications
Applications, Web portals
Grid Programming Environments and Tools
Languages, interfaces, libraries, compilers, parallelisation tools
User-level Middleware
Resource management and scheduling services
Core Grid Middleware
Job Submission, storage access, info services
Security Services
Single sign-on, authentication, secure communication
Grid Fabric
PC's, clusters, networks, software, databases, devices + associated resource managers
System level
Figure 1: Layers of Grid architecture

It is useful to establish a context from which to base subsequent discussions of the software components making up the Grid infrastructure.

Layers of Architecture

Fabric

These are the resources which already exist, but which we wish to be aggregated in the Grid infrastructure. This includes compute and data resources, and other more specialized resources such as monitoring and measuring equipment. Such resources are under the control of a local manager such as LoadLeveler, LSF, PBSPro, and OpenPBS. Such managers may also provide more advanced functionality such as advanced reservation capability, which in turn can be exploited by the high level Grid infrastructure in order to co-allocate multiple resources.

Security Infrastructure

The security infrastructure layer in the Grid architecture is positioned directly above the fabric layer, and below the core and high-level services layer, as it is a built upon the local security policies of the fabric. It is required for the Grid services layer to operate.

Authorization and authentication are essential functions of any distributed environment. The problem of providing these security services over a Grid are even more complex, as a Grid aggregates resources controlled by a number of independent organizations, each with its own security policy. Therefore, a key requirement of Grid security is the interaction with local security without change to the policy and its administration. In current systems this commonly involves mapping from the Grid security mechanism onto each local security mechanism, i.e. a local user name.

Actually, in addition to local security measures, community security services can also be deployed in a typical Grid. For example, the main basis for the security of UNICORE is in the form of such a service. An alternative example is that of the Community Authorization service of Globus.

An important and possibly controversial technique used is that of credential delegation. In order to support a dynamically changing job and the utilizations of resources at run-time, a resource on the Grid must be able to grant permission to a resource enabling it to act on behalf of the original resource, i.e. the first resource can make a signed request on behalf of the original resource to another resource. This method of credential delegation forms the basis of security infrastructure of Globus.

Core Grid Middleware

This is concerned with the management of individual resources in a Grid, and the integration services provided by the components of this layer makes it similar to the Resource layer as described in [1]. The aim of this layer is to provide a layer of abstraction on top of the underlying resources. Fundamental ideas to these aims include the following:

- Resource Management - services providing a homogenous and secure interface across the geographical and administrative domains for accessing the underlying resources, i.e. job submission. On a single resource, such services are responsible for negotiation with local policies and scheduling systems.

- Information services - service collecting and publishing data about the status of the resource (computing power, networks, storage systems, etc.). This information is used by a wide range of other services Aggregate Information services, i.e. containing information from multiple resources are classified as user-level middleware, and discussed in the next section.

- Data Transfer - services enabling secure, authenticated and authorized data transfer across administrative heterogeneous domains in the Grid.

User-level Middleware

Middleware in this section provides services across multiple nodes of the Grid. This includes information services, resource brokers and schedulers:

- A Grid information system, to store and provide information about the status of the entire Grid (individual resources, virtual organizations, networks...)
- Seamless allocation mechanisms for all the Grid resources provided by a high-level (user-level) standard mechanisms and languages for resources request specification
- Dynamic allocation of resources, including co-allocation and co-reservation mechanisms, for distributed parallel applications running on different machines
- Services providing virtualized access to distributed data stored throughout the Grid.

It is quite tricky to cleanly categorize the remaining software associated with a Grid, and in the literature there does not seem to be a consensus on either the definitions, for example "Problem Solving Environment" (PSE), or classification of software. In [1], the following is defined, "workload management services and collaboration frameworks – also known as problem solving environments – provide the description, use and management of multi-step, asynchronous, multi-component workflows". In [29], there is a more general description; "the terms – PSE, framework, workbench – are used to describe a wide variety of tools for developing and executing applications".

Grid Programming Environments and Tools

These provide a layer between applications and low-level Grid services, or the Grid enabling of applications, or indeed it is a toolkit for constructing Grid applications. They provide a programming environment for exploiting the resources of the underlying infrastructure.

This often happens through Grid-enabling of familiar programming environments. This can happen at various level of granularity. As an example of a low-level environment, MPICH-G2 offers the Message Passing Interface functionality over a Globus Grid.

A higher-level programming approach is provided by the CoG kits for Globus, where an application-level API is provided. This allows a programmer to use a familiar programming environment such as Java to construct a workflow of Globus tasks.

An example of an even higher-level programming environment is the Grid Portal Development Kit (GPDK), which uses the Java CoG kit, but is itself a toolkit for constructing web-based portals.

Grid Applications

As discussed above, we consider Grid Application as one often having a visual interface. Software in this section allows a user to focus on their own problem domain, and their application/portal deals with the interface to the underlying infrastructure.

Business interest in Grid Computing

Grid computing is an example of a predominantly academic research initiative making a jump into the commercial world. Like the adoption of the internet from a core user base in the scientific community, the business case for Grid computing is growing quickly. This is no surprise as exactly the same factors and pressures, such as formation of Virtual Organizations (VO's), where coordination of distributed resources both within and across organizations is increasingly important in a business context.

However, the structure and nature of the business community is such that, its participants may not be eager to openly share resources without security and renumeration guarantees. Additionally, the process to standardize the core Grid protocols is well underway, and the fact that businesses have become involved has motivated some of the new research, directions, use-cases, etc. Thus some factors which are not so important in scientific research, find a new momentum in the commercial world, and this drives new research and development.

Many companies are very active in the area of Grid computing and similar, offering commercial Grid solutions and contributing to open-source Grid solutions, including IBM, AVAKI, Sun, Platform, Entropia, Pallas etc.

User Requirements

Introduction

The survey was made available on the internet and the user base of German HPC centers was asked to fill in the form. Potential respondents were contacted via email, and after a brief introduction were invited to complete the questionnaire. The request for feedback was sent out to roughly 500 users. Of these 63 completed the questionnaire. A return rate of 13% is rather high for such a survey.

Survey

The questionnaire consists of 47 questions. In order to improve the quality of the analysis, we can divide the 47 questions in the survey into the following topic areas.

Awareness and Perceptions (questions 1-17)

Initially, the questions establish a grounding of the general awareness of the surveys audience, from which the remainder of the questionnaire stems. The following questions in this group attempt to gauge how respondents feel about the future impact of Grid computing in a wider, more general sense, including its impact on society itself. Some of the more sensational reporting in both the IT and non-IT press claim that the widespread acceptance and adoption of Grid computing will bring about the same evolutionary jump as did the World Wide Web. This is a typical statement for which we wish to gauge opinion.

We can safely assume that the great majority of the respondents will be connected or have some interest in High Performance Computing, and thus the questions in this section also address the future of HPC and HPC centers. As we have already mentioned, Grid computing is not the exclusive preserve of the HPC community, but this has been the test-bed for most Grid experiences to date.

Current Usage and Network Requirements Appraisal (question 18-29)

This section evaluates the current mode of usage (directly dealt with in question 18), and the network requirements of current applications and tasks. This includes requirements for file staging and post-processing, performance profiling and debugging support, and preferable Quality of Service (QoS) parameters for network latency and bandwidth

Future Expectations and Requirements (question 30-48)

In this section we attempt to gauge what the respondents realistically think they could expect from a Grid system. It discusses what is available today, and thus would also expect to be available in a production Grid environment. Finally, it attempts to evaluate the level of need for high-level or user-level services currently available in some Grid software, such as seamless access, resource brokering, graphical interfaces and Grid programming models.

Analysis of Results

In this section we review the results of the questionnaire, following the three sub-sections defined above. There were 63 completed questionnaire returned. The following summarizes the roles of the questionnaires respondents:

43	users
18	developers
1	Grid administrator
1	Grid developer

The majority of people are already either users or programmers of applications, who are either currently using or would be interested in how Grid computing could benefit them in the future.

Awareness and Perceptions (questions 1-17)

First of all the majority of the respondents were familiar with the expression "Grid" (Question **1**, *"Are you aware of the term 'Grid'?"*). This is a good start. Only 6 of those questioned were not familiar.

Question **2**, *"Name 5 terms that you associate with the term Grid?",* takes a straw poll of expressions commonly associated with "the Grid". The results can be summarized as follows:

- <u>Common responses</u>: as simple to use as a PC, coordinated, flexible, cheap, usage of a super computer
- <u>Rare responses</u>: consistent, reliable, secure, widely spread

These answers are the expressions for 'commonly associated', and not 'implemented features'. It shows some of the common understanding of when the Grid is mentioned. There is a clear message. The most publicized aspects of Grid computing are commonly associated, and conversely in areas such as topics which have proved difficult in the development of computing and distributed computing for a long time, there is less association. It has to be said that these qualities are notorious in their ability to leave people unconvinced.

Computer security is one such topic where current applications and Grid computing infrastructures are somewhat unproven. Security in a Grid is a highly complicated problem, addressing issues such as security across wide area networks, distinct administration domains and multiple local security mechanisms. This is compounded with the fact that most Grid middleware has had the majority of its use so far in relatively safe research environments.

Questions **3** and **4** deal with awareness of the term Problem Solving Environment (PSE) – *"Are you aware of the term PSE?"* and *"If so, does the Grid in your opinion represent a PSE?".* Overall, we can conclude that 'PSE' is not a particularly well recognized expression, and of those who do recognize it, opinion is divided to whether the Grid represents a PSE.

The following taken from [40], summarizes the role of a PSE very nicely, "a PSE is a computer system that provides all the computational facilities needed to solve a target class of problems. These features include advanced solution methods, automatic and semiautomatic selection of solution methods, and ways to easily incorporate novel solution methods. Moreover, PSEs use the language of the target class of problems, so users can run them without specialized knowledge of the underlying computer hardware or software. By exploiting modern technologies such as interactive color graphics, powerful processors, and networks of specialized services, PSEs can track extended problem solving tasks and allow users to review them easily. Overall, they create a framework that is all things to all people:

they solve simple or complex problems, support rapid prototyping or detailed analysis, and can be used in introductory education or at the frontiers of science."

Actually, a Grid provides the ‘backend’ services on which a PSE depends. Through a PSE a user is presented with a user-friendly view of these services, and the user only needs to be vaguely aware of the underlying infrastructure on which the problem PSE depends. Perhaps this is a case of well known functionality with a little known name. Maybe that the currently available technology is not well understood, and that more information and promotion change its lack of recognition and reputation. Certainly, the functionality which a PSE provides would appear compelling to facilitate the usage of resources available on Grids.

Question **5** is *“What role will Grid computing in your opinion play in the future?”*. Obviously, those questioned are more critical and careful about this technology than those working in the Grid marketing divisions of the businesses selling Grid software or support. Marketing materials commonly say that as the World Wide Web offers global information dissemination, the Grid will provide a global environment for computing, collaboration, and communication, and that Grid computing technology matures and becomes widely available, we are promised a revolution in technology and in society. Our respondents only partially concur with these bold claims, and a good proportion of the answer believe the Grid will be “not so important“. It would be very interesting to conduct a similar survey in 5 or 10 years time, and observe how attitudes have shifted.

The remainder of the questions in the “Awareness and Perception Section” (questions 6-17) are summarized in Figure 2. These questions share the same choices of answer, ranging from “I totally agree” to “I do not agree at all”. In general, respondents tended not to agree with broad, sweeping statements about great possibilities for Grids. Respondents did agree with statements that placed the value of Grids in a very limited context.

Questions 6 and 7 relate to some of the statements from the popular press with regard to Grid computing, and evaluate the respondent’s opinion on the degree to which they are accurate.

Question **6**, *“In the future Grid computing will provide computing power just like today’s electric Grid”* has become the classic statement one reads in conjunction with Grid computing. Similarly, question **7**, *“Grid computing is a bubble that will soon burst”*, gauges the opinion to how Grid computing is currently ‘sensationalized’ through hype and marketing. For both questions, the middle answer, “partially agree”, is the most popular.

It could be said that the answers to these general questions are particularly dependent on the personalities of the respondents; we can expect varying levels of cynicism, pessimism or indeed optimism from the respondents. As reported in [6], “Cynics reckon that the Grid is merely an excuse by computer scientists to milk the political system for more research grants so they can write yet more lines of useless code.” However, we are pleased to report that, although it looks as if the replies are tempered with some degree of healthy caution, there is an impression that the Grid is an infrastructure worth having!

Questions **10** and **16** assess the general perception of Grid computing from the users perspective, and how it will change their working practices. For both questions *“Grid computing will make life easier for the user”* and *“Grid computing constrains the user more than it helps him”*, the results of the survey are inconclusive.

As discussed in the analysis for questions 6 and 7, there is a degree of negative response regarding the impact of Grid computing. Evidently it is too early in the game to make conclusive comments regarding the Grid and the user experience. We can conclude however, that the feedback is not completely negative, and in the fullness of time we shall see to what extent the Grid computing will satisfy the requirements of its users.

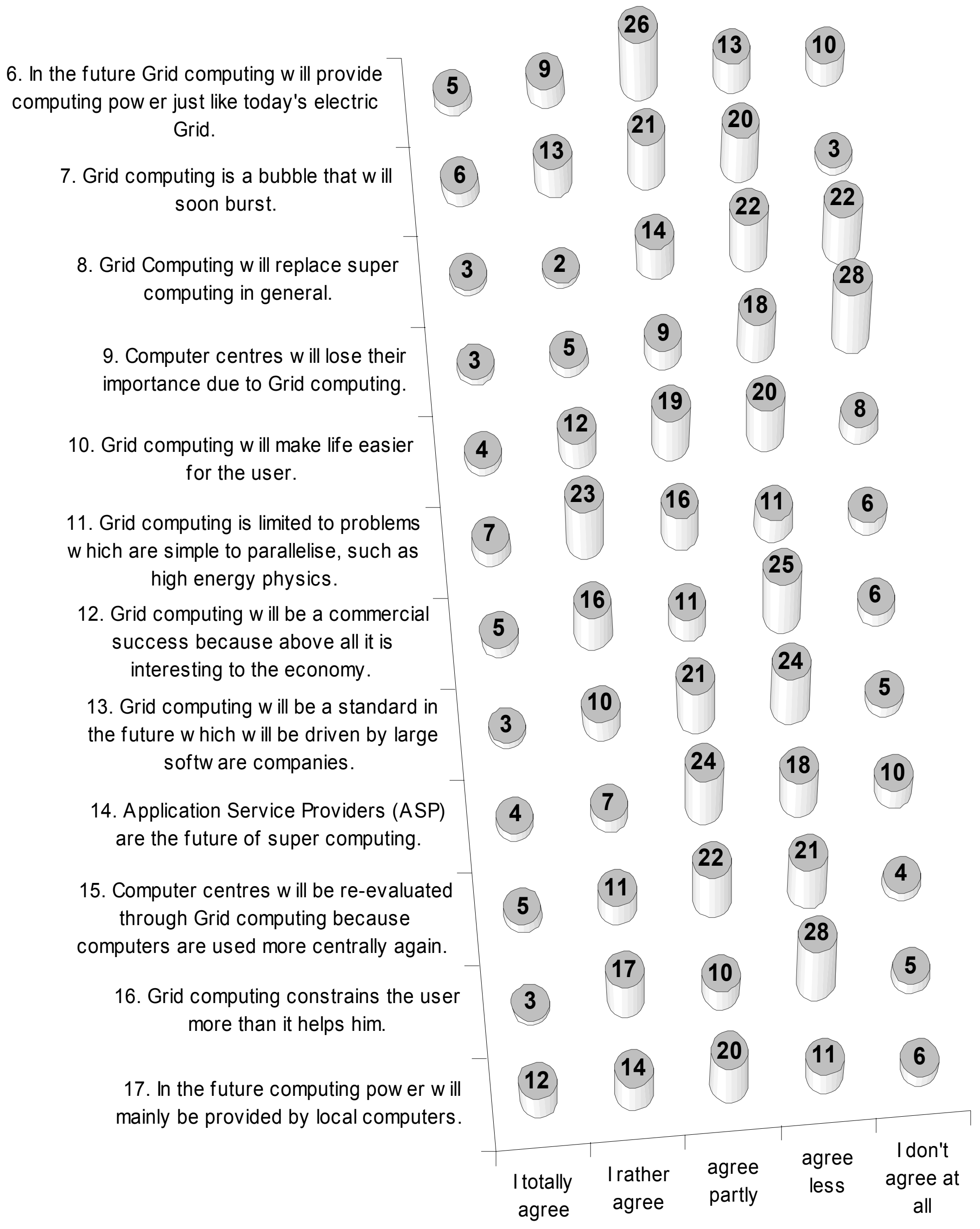


Figure 2: Questions 6-17

Questions 12 and 13 are general questions regarding commercial and economic influences on the future development of Grid computing. From question **12**, there is a slightly greater level of disagreement with *"Grid computing will be a standard in the future which will be driven by large software companies,"* than agreement.

Until reasonably recently Grid computing has been a phenomenon largely developed and exploited by those in the research community. The evolution of Grid computing is similar to that of the World Wide Web, initially within the research institution, and later on adopted by the commercial sector. In the case of the World Wide Web, the standards have been controlled through an organization such as the W3C consortium [35]. Building upon the initial genesis of the ideas, the stewardship of standards through organizations such as the W3C has been very successful, and the web has consequently exploded because it is open. It has developed so rapidly because the creative forces of thousands of organizations companies are building on the same platform, and whilst some companies may wish to introduce propriety mechanisms in order to bind its customers to their particular platform or system, in practice, the drive to remain interoperable is too strong.

For Grids, there is currently a committed drive towards the standardization [14] of the required protocols needed in order to build a Grid. This is discussed in detail later in the report, but so far this would appear to be a relatively harmonious partnership between interested parties, including both research communities and commercial companies, with the result of the derivation of a set of open standards for the operation of a Grid.

So, Grid computing is a subject initially evolved within the research community, and more recently commercial enterprises are beginning to exploit the technology too as well as with the Internet before it, the Grid infrastructure and standards are being developed in a relatively open manner, and not necessarily driven by large software companies. This is reflected in the answers to question 13, and indicates that respondents belief in the honesty and openness of this process.

Question **12** asks, "*Grid computing will be a commercial success because above all it is interesting to the economy*". Overall, there is some disagreement with this statement.

During a phase of commercialization a new idea, particularly in the computer industry, there is a great deal of investment, excitement and business generated. Often the new approach is marketed as a 'paradigm shift'. Customers have to decide whether they believe the promises have value, or they will contribute to a bubble economy without substance. With Grid computing, the industry recently has certainly witnessed lots of publicity. This is at a time when the economy is not at its most buoyant, suggesting that there is a real belief in the value of Grid computing.

On one hand, Grid computing encourages more efficient usage of resources. This statement does not imply just cycle scavenging on desktop computers, but usage of super-computers to their full capacity and potential, and the coordinated usage of multiple resources. Crucially, the implication is also the enabling of human resources as opposed to machine resources, and their collaborations and communities – often referred to as Virtual Organizations.

Often, the businesses marketing and promotion of Grid technologies are also involved the selling of computer hardware, such as IBM, HP, Sun. Assuming that Grid technologies encourage the more efficient use of existing resources, it would appear the activities of selling Grid technologies and computer hardware are incompatible. However, these companies claim to understand that what they are selling is really a suite of tools designed to make companies more effective, efficient, or innovative. In these sense there is a move towards the selling of "services" and "service solutions" rather than hardware. The investment of large computer hardware vendors in Grids suggest that these companies believe Grids will result in better services to sell to their customers.

We think that Grid computing shows a lot of promise, and that it will make the more efficient use of resources possible, as well as the dynamic creation of virtual organizations. Such initiatives should prove to be effective, and thus be commercially successful. However, there are a number of driving forces behind Grid computing, and economics is only one of them.

Questions 8, 9, 15 and 17 relate to the future computing infrastructures, particularly the future of super-computers and computer centers [8].

The consensus of respondents to the statements of questions **8** and **9**, *"Grid computing will replace super computing in general"* and *"Computer centers will lose their importance due to Grid computing",* is that the Grid will not spell the end for computer centers or High Performance Computing in general. What is more likely to happen is that the demand grows to make use of the newly added capability and there is an opening of new channels through which these resources can be exploited and used to their full potential. So, for example, an application which previously ran at a single site on one supercomputer may now be distributed across multiple supercomputers.

One of the many arrangements utilizing the Grid infrastructure building-blocks is the creation of 'cycle scavenging' Grids or high throughput systems [41]. These are typically constructed from lots of desktop computing resources, to exploit the unused capacity of the machines. This aggregated computing power can provide a good source of computing or storage capacity.

However, many problems require tightly coupled computers, with low latencies and high bandwidth communication. "Embarrassingly parallel" applications and parameter studies which do not require much communication between processors are good candidates for computation over a "cycle scavenging" Grid of workstations. However, tightly integrated parallel applications will run most efficiently on super-computers. Hence, the consensus was that HPC and computer centers will have an important role in the future.

The results from question **11**, *"Grid computing is limited to problems which are simple to parallelize such as high energy physics"***,** are interesting in this regard. There is some level of agreement with this statement. We note that there seems to be some lack of awareness for the scope of Grid computing. Grid computing enables all of the following: parallel processing across geographically distributed, (super) computing resources, collaboration of participants in virtual organizations, sequencing a process requiring multiple resources (including, for example, specialist visualization resources) and utilizing otherwise unused resources.

It is true that the Grid allows the extending of parallel computing paradigms from tightly coupled clusters to geographically distributed systems, and due to networking properties over a wide-area, parallel applications with less communication (and thus simple to parallelize) will run more efficiently on a collection of widely distributed resources. However, this does not imply that Grid computing is limited to such applications.

A small majority disagree with number **15**, *"Computer centers will be re-valued through Grid computing because computers are used more centrally again"*. Conversely, a small majority thinks *"In the future computing power will mainly be provided by local computers"* (question **17**). The to-and-fro between using computers centrally or locally has been witnessed several times if one looks back through the past 3 decades of computer evolution.

Certainly the widespread acceptance of Grid computing should make super-computing resources much more widely accessible. Indeed, one of the top answers to question 2 about terms associated with the Grid was "usage of a supercomputer". Additionally, the authors feel that the Grid model offers a hybridized model for computer use where all resources big or small, offering compute power, storage or specialized devices can be all integrated through the Grid. This does not suggest a move to computers being used more centrally again. The rapid and continuing growth in the power of desktop computers, and its cheap availability, will ensure that the compute power of local machines will play an important role in the future of Grids.

Another direction for computer centers will include more deployments of Linux clusters, constructed from cheap, of-the-shelf components. In addition to super-computers, computer centers could become hosts for specialized devices, such as visualization equipment, accessible over the Grid.

We think that computer centers will be positively re-valued through Grid computing, and that local computing resources will continue to play an important and contributory role, but certainly not to the exclusion of other resources.

The answers to question **14** *"Application Service Providers (ASP) are the future of super computing",* show a small majority of people disagree with this statement. Application Service Providers offer implementation and subsequent operations management of one or more networked application, on behalf of its customers. They are usually associated with business operations, rather than in research/academia. In the business context, this model for outsourcing some of their IT operations is quite attractive to small and medium sized businesses, as it is a viable alternative to managing such operations in-house.

Possibly a re-worded question implying 'Grid enabled ASPs' might have gained more agreement. We feel that this could be a useful future direction for ASP's. As discussed in [1], a typical ASP does not provide dynamic provisioning, load balancing, or indeed interaction with another ASP or Storage Service Provider. These are features which a Grid enabled ASP would be able to provide. The EU project GRASP [4] operates in this area. This project aims to use GRID technology in order to realize current and future ASP business models that integrate distributed and heterogeneous resources.

Current research directions such as Service Level Agreements for resource allocation [42], Quality of Service (QoS) mechanisms and economic resource brokering, and the involvement of businesses, for which payment mechanisms for Grid usage are necessary, are important services for Grid-enabled ASP's of the future.

Current Usage and Network Requirements Appraisal (question 18-29)

Question **18**, *"What type of application do you mainly use",* evaluates the types of application, and which are the most common. We can summarize the most popular application types as follows,

- Popular application types: pure batch mode, workflow/piped applications, parallel applications (with varying levels of communication).

Steered, or controlled applications are rather specialized, particularly for simulations involving resources across administrative domain. This is a fairly advanced form of a distributed application which is being enabled through Grid technologies. At presence, current requirements or capabilities do not commonly point to this type of application, although we envisage that such requirements and capabilities will become more sought-after and requested in the future.

Questions 19 to 29 evaluate the user's requirements for a Grid infrastructure based on their present usage and priorities. Taking each question in term, the results can be summarized as follows

19. Network requirements of applications vary widely, but the most demanding requirement "frequent and intensive communication between computers" is actually the top answer.
20. Typical programs are normally/strongly dependent on network latency.

21. The majority of users require network latencies to be in range of 0 up to 100 microseconds.
22. Typical programs are strongly dependent on network bandwidth.
23. The majority of users require network bandwidth greater than 100Mbit/s, and a good proportion require a bandwidth of over 1 Gbit/s.
24. No consensus found for size of input data. It ranges from "up to 1 MB" up to "over 1 GB".
25. The response to the size of output data is weighted towards to larger files, i.e. "over 1 GB".
26. A majority of respondents do not, or only to some extent, use parallel I/O. The most popular answer was "no".
27. A majority of people require post-processing of generated output data, at least sometimes. Very few respondents never require this functionality.
28. A majority of respondents sometimes use debugging tools on high performance systems.
29. A majority of respondents sometimes use performance profiling and analysis tools on high performance systems.

From this, we can conclude that a high performance networking infrastructure is an essential requirement in a Grid infrastructure. We can take the answers to questions 19-29 summarized above as a 'wish list' of network requirements when establishing a Grid infrastructure.

Future Expectations and Requirements (question 30-48)

The final group of questions, approach the subject of future expectations and requirements for Grid computing. Aware that a number of approaches can be taken to a current application and then make it "Grid-enabled", this section attempts to establish a list of fundamental capabilities which would make it more attractive for a user to migrate their application to the Grid.

In general, we felt that those who answered the questionnaire had a realistic and pragmatic attitude to the Grid, and what they could expect from it. This, no doubt reflects the respondents experience and knowledge of high performance and distributed computing, and its limitations. For example, the majority of answers did not agree that using a Grid would make their application easier to handle, and that their application would require some changes, to make it Grid enabled. This is more practical, than the vision of "plug-in" computational power. Evangelistic statements regarding the future of Grid computing, and the merging of the concepts of Grid computing, Semantic Web, and Web Services, and the consequent transparency of using any networked resource, seamlessly, and by *any* non-expert user, are current research topics, and the availability of such technology is some way off.

As such, the responses to this section of the questionnaire evaluate the 'near term' future requirements. For example, on issues which are achievable today, we feel that the respondents exhibited strong convictions regarding a number of requirements for Grids which are seem as absolutely essential, for example Single Sign-on.

Questions 30-34 cover programming models to run applications on the Grid. These are summarized in Figure 3.

The most popular answer to Question **30** *"... your application will be easier to handle"* is 'agree less'. We are arguably in the early adoption phase of Grid evolution. Grid users must now be to some extent technology orientated. As progress is made, the Grid will become ubiquitous, and consequently users will not need to be technical experts, as users of the World Wide Web are today. The deployment of the infrastructure will become easier, knowledge more widespread, something like running a HTTP server today. This vision for Grid computing is currently someway off, and thus the current feeling is that applications on the Grid will not make applications easier to handle, although we expect this to change as the current software evolves, and new software developed.

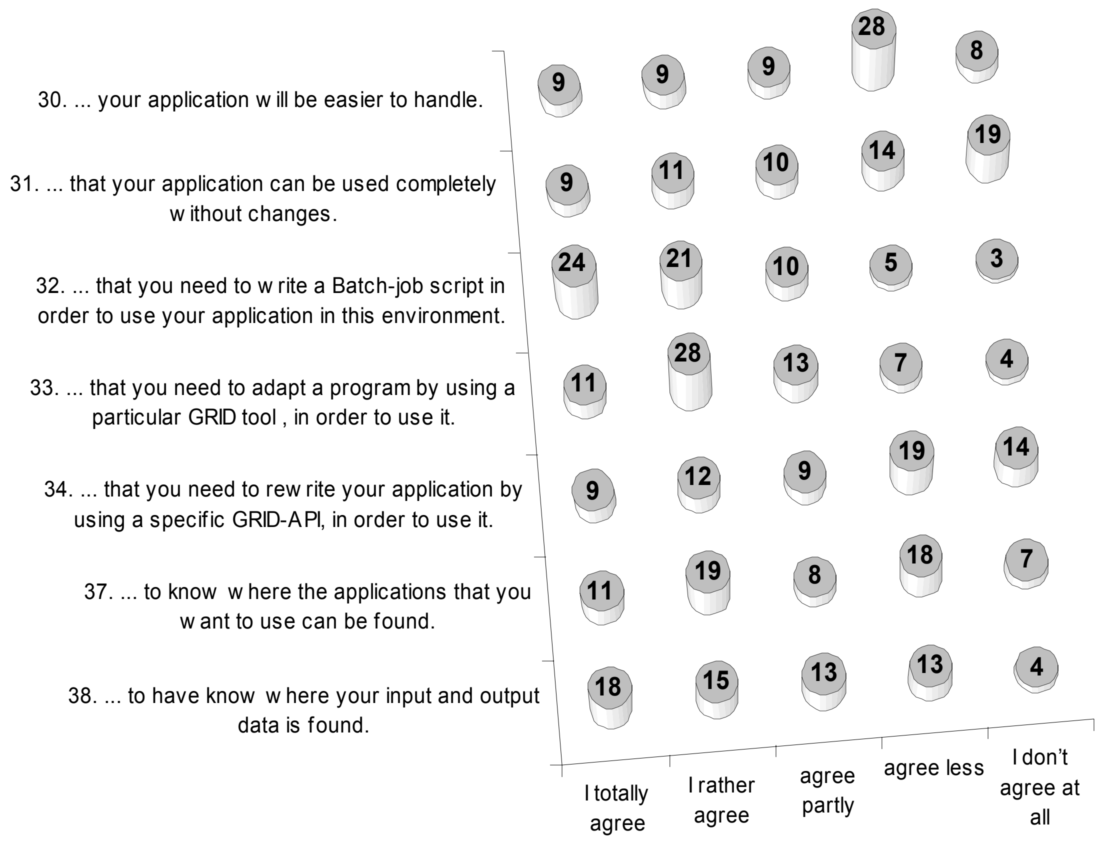


Figure 3: Questions 30-34, 37 and 38. When using a GRID you expect …

Question **31** questions the users expectation *"... that your application can be used completely without changes."* A majority do not agree with this. Our respondents expect some of alteration will be necessary.

An overwhelming majority of people expect *"... that you need to write a Batch-job script in order to use your application in this environment"* (question **32**). Batch submission of jobs to compute resources is an effective and proven method, and very much still in use today.

Of course, the aims for the broad application of Grid computing technologies extends further than providing seamless access to remote resources in order to submit a batch job script. Nevertheless, the more realistic ambitions for Grid computing and certainly how it is most often used today include batch scripts as a core mechanism for resource management.

In the future, an alternative to batch access is proposed by the methodology of the Service Oriented Architecture (SOA). This forms the basis of the thinking of the Open Grid Services Architecture [14], and promotes a change from thinking in terms of resources and jobs, to that of services. As defined in [17], "a service is a network-enabled entity that provides some capability." This includes the notion of *virtualization*; implementation encapsulation through a common interface. Consequently, this also has the advantage of being able to disallow user's direct access to a local account. However, such prescriptive use of a remote resource may prove too inflexible for a typical user of a compute resource, and they may require access to an account on a local machine and direct access to the batch scheduler in order to meet their particular requirements.

Most people expect *"... that you need to adapt a program by using a particular GRID tool, in order to use it"* (question **33**). The key word is 'adapt'. The respondents to the questionnaire were unable to come to a consensus regarding the expectation *"... that you need to rewrite your application by using a specific GRID-API, in order to use it"* (question **34**). However, we can conclude that there is understandably more reluctance to rewrite an application, than to simply adopt it, for use on the Grid. A good example of the approach queried in question 33, is that of MPICH-G2 in Globus. This is a Globus-enabled implementation of the Message Passing Interface standard. MPI applications should port relatively easily to a Globus Grid. However, the programmer should be aware that the processing nodes over which the program runs are less likely to have the closely-coupled, high-bandwidth/low latency characteristics which are typical when deploying a parallel program over a cluster or super-computer.

Questions 35-38 address ease-of-use, front-end requirements, seamless access (including automatic file staging). The results are summarized in Figure 3.

Question **35** evaluates the expectation "*... that you have to login per session ...".* This is addressing the expectation that a Single Sign-on (SSO) mechanism would be in place when using the Grid. Not surprisingly the result is overwhelmingly in favor of SSO. Extending the use of a single resource to the use of multiple resources over a Grid, the obvious inconvenience of manually presenting a security credential to each is recognized.

Question **36***." ... as an interface ..."* gauges if users expect to use a graphical or console based to perform their tasks on the Grid. Both graphical *and* console are the most popular answer, and additionally console access (only) is popular. At first glance, the popularity of console access is surprisingly popular, considering the user-friendly gains from using a graphical interface. However, console access is presently the most prevalent method of accessing remote resources, and additionally there is a perception perhaps that a graphical interface can sometimes lack the flexibility provided by console access. Choosing both console and graphical access gives the best of both worlds.

There is no definite answer to the expectation of question **37**, *"... to know where the applications that you want to use can be found".* Such transparency of application location would appear to be a useful functionality of a Grid use. In order to provide such a function, one could either use a dedicated PSE, where the user does not need to be aware of the application to be found, and this 'virtualization' is a function of the PSE. UNICORE provides such a mechanism in the form of an application specific plug-in, and the abstraction of software resources through the UNICORE client. Use of the Grid where the user must explicitly state the location of applications is possible for all Grid toolkits.

The results to *"... to have to know where your input and output data is found"* (question **38)** are marginal, but weighted in favor of some agreement. This result is quite surprising as having not to be overly concerned with location of the input and output data provides easier operation for the user of the Grid.

Questions 39-42 deal with resource selection, resource brokering and scheduling.

There is a agreement with the statements **39**, **41** and **42** *"...that you can explicitly choose the resources (hardware and software), which you wish to use",* "*... that the GRID automatically selects the best resource for you."* and *"... that the GRID offers both options (automatic and user controlled search for resources)***".** These questions cover two scenarios for resource selection on the Grid; explicit and automatic or assisted selection.

The first scenario is where the user of the Grid directs a job to a specific machine. This is the simplest mode of job submission and as such is supported by almost all Grid systems.

The second scenario involves the use of resource brokering functionality. Referring to Figure 1, such services exist in the user-level middleware. The four systems we have evaluated have varying degrees of support for this mode of operation. In the case of Globus, such functionality is not provided by the core middleware, but there is good support by additional global-enabled brokering and super-scheduling software. Currently the support for resource brokering in UNICORE is limited, but improving through the results of the EuroGrid project [9]. This is providing assisted, but not automatic resource selection.

For both scenarios, the questionnaires respondents would expect that such functionality would be available, and that automatic resource selection must have a 'manual override' facility.

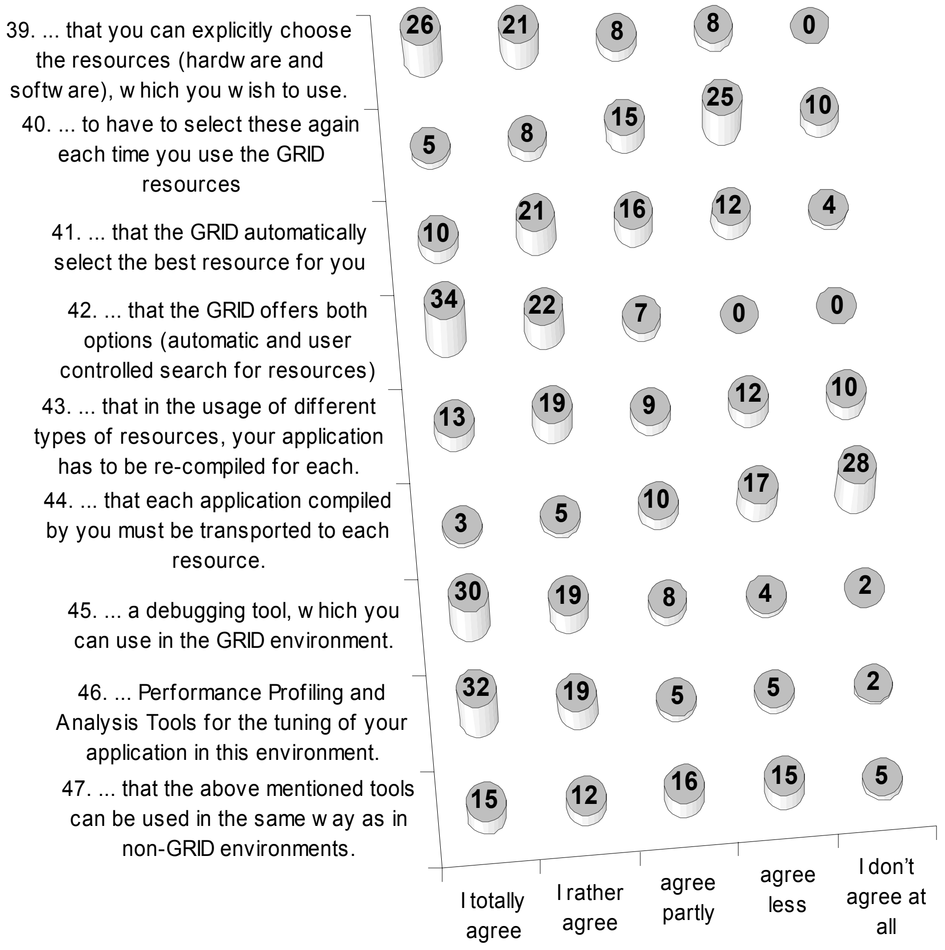


Figure 4: Questions 39-47 When using a GRID you expect…

Question **40** reads *"... to have to select these again each time you use the GRID resources",* and 'rather do not agree' is the most popular answer to this. It seems to be on the users "wish list" to have a saving method for the whole run of their jobs, including some information about the used resources. For instance, UNICORE supports the user in this area: a job description can be saved and edited off-line if necessary, where the job description includes the resources for the jobs, and then this can be opened and re-used.

Questions 43-47 cover compilation and debugging issues.

Question **43** asks the respondents do they expect *"... that in the usage of different types of resources, your application has to be re-compiled for each"*, and the majority of people disagree with this statement. In practice, given the heterogeneous nature of the resources on the Grid, this is hard to achieve. What is achievable is the abstraction of compiler tasks for each platform. Application migration becomes possible when compiled applications can be freely moved between resources. The 'write once, run anywhere' promise of Java, where applications run within a virtual environment, recommends it for providing this desirable quality for application execution. However, Java is somewhat unproven as a programming language for HPC, and there are often complaints of poor performance in HPC applications written in Java.

A strong majority of people do not expect *"... that each application compiled by you must be transported to each resource"* (question **44**). This aspect of seamless use of resources on the Grid would appear to be high on the list of requirements from users, and is a functionality of some currently available Grid software. Connected to this requirement is the seamless staging of input and output data files.

In the answers to both questions **45** and **46**, "*... a debugging tool, which you can use in the GRID environment"* and *"... Performance Profiling and Analysis Tools for the tuning of your application in this environment"* the respondents displayed strong agreement. On this evidence we can conclude that these are desirable tools which should be available with a Grid infrastructure.

However, the respondents expectations were inconclusive regarding the expectations *"... that the above mentioned tools can be used in the same way as in non-GRID environments"* (question **47**). In order to minimize the effort required to migrate an application to the Grid, it is clear that keeping current operational and developmental practices as un-changed as possible would be preferable. We suspect that many respondents to this question will in practice expect that this will not be possible, and that some adjustment will be necessary.

Tools

Introduction

In the first chapter we introduced a general model for Grid architecture, and described the primary functionalities of the various layers. For the remainder of this chapter, we will discuss the principles and features of the following Grid systems: UNICORE, Globus, AVAKI and TME.

Globus

Globus is an open-source initiative to produce a standard Grid architecture for distributed resources. In recent years, a Grid middleware standard *de facto* has emerged in the form of the Globus Toolkit.

The choice of Globus as a Grid middleware provides functionality in the Security Infrastructure and Core Grid middleware layers, and some components in the User-Level middleware layer. Essentially, this is the extent of the Globus software – it is not an end-to-end solution. However, these integration services at the neck of the hourglass architectural model provide a homogenous and secure interface across the geographical and administrative domains for accessing the underlying resources, on top of which the higher level services can be built. Additionally, it is possible to use the Globus command line utilities to interact with the core middleware layer directly.

However, when installing Globus, it is usual to install some additional, higher and lower level packages. These provide additional services built around the core middleware, and although not 'official' Globus components these additional services are often present in typical Globus-based Grids. These include batch systems at the lower layer, and often include Grid portals or PSEs at the upper layer.

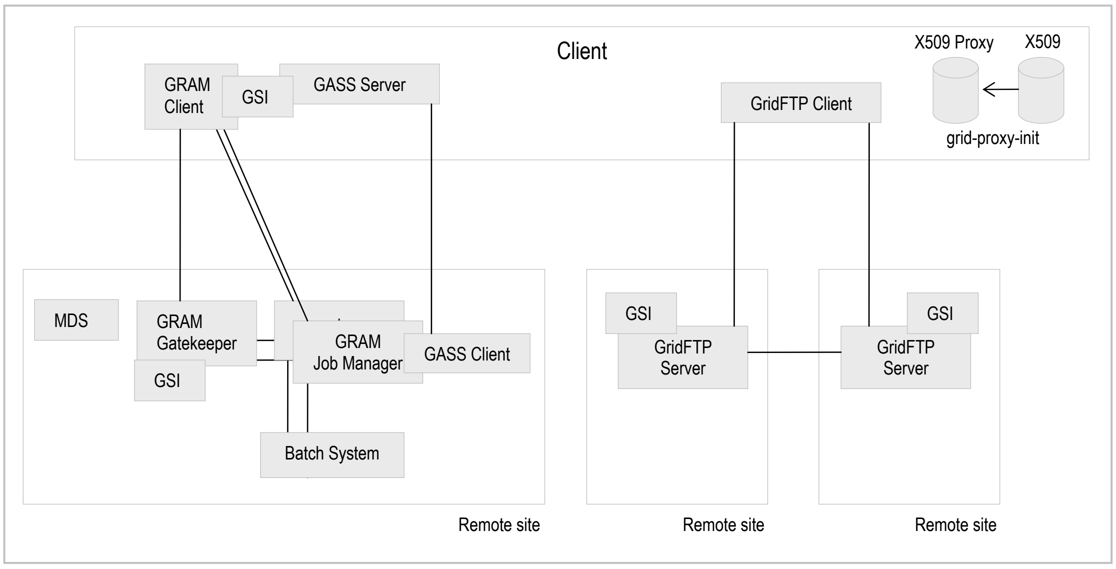


Figure 5 Globus Architecture

A conceptual overview of the Globus architecture is shown in Figure 5. As Globus services can be deployed relatively independently on each node in the Grid, a large number of deployment options are available.

Fabric

The Globus Resource Management service co-operates with local scheduling systems, for example LoadLeveler, LSF, PBSPro and OpenPBS. In order to support co-scheduling of resources, advanced reservation must be provided as a additional service. Some modern schedulers provide such capabilities, or an additional software component such as the Maui software can add such functionality. Alternatively, the experimental Globus GARA service provides advanced reservation capabilities, if the local schedulers do not support these features.

Security Infrastructure

The Globus security infrastructure is called the Grid Security Infrastructure (GSI). This works using the mechanism of delegated credentials. This allows the construction of new capacities dynamically and transparently from distributed services, an often stated requirement for a Grid. Credential delegation allows a resource to access further resources by using the temporary credentials from the client, almost as if the client made the request directly.

Core Grid Middleware

Most of the Globus components are present at this level. As per the Globus documentation, there are three 'pillars' of services, all built upon the security of the GSI. These pillars are:

- The first pillar provides Resource Management, which involves the allocation of Grid resources. It includes such packages as the Globus Resource Allocation Manager (GRAM), and the Globus Access to Secondary Storage (GASS). GRAM enables users to schedule and manage remote computations, whilst GASS facilitates the staging of executables and files.

  Globus provides a number of command line interfaces to the GRAM Resource Manager. Specifically, GRAM services allow users to submit jobs, bind to already submitted jobs, and cancel jobs on remote computers. Other services allow users to determine whether they can submit jobs to a specific resource (through a Globus gatekeeper) and to monitor the job status.

  Clients first connect to the Globus gatekeeper to initiate a Job Manager and user job. Communications then take place from the client and the JM to manage the job, stage data and/or executables, and return job output to the client. These connections are initiated in both directions.

  Prior to Globus version 2.2, the Globus client had to explicitly initiate the transfer of necessary files to the remote machine. This is the function of the GASS services, enabling transfer input/output files and executables between the client and the remote resource. Globus releases after version 2.2 added automatic file staging to Globus resources. This makes the regular operation of Globus less long-winded.

- The second pillar is the Information Services, which provide information about the Grid resources. In Globus this is undertaken by the Monitoring and Directory Service (MDS), providing the Grid Resource Information Service (GRIS) and the Grid Index Information Service (GIIS). The MDS Globus component is based on the Lightweight Directory Access Protocol (LDAP).

  The MDS simplifies Grid information services by providing a single standard interface for the many different information services and sources used within a Virtual Organization (VO). The MDS provides a mechanism for publishing and accessing both static and dynamic system and application data for the Grid. A key feature is that the MDS supports a dynamically changing Grid, where the information services reflecting these changes.

  The information stored in a GRIS is extensible. Local systems may have a number of information gathering mechanisms, and through the MDS 'provider' mechanism, these can be plugged in to a GRIS. For example, the Globus GRAM reporter provides information from a GRAM service, such as queue information from a local queuing or scheduling system such as LSF or PBS. Alternatively, Ganglia [11] can be integrated as an information provider for a cluster installation.

  The information service for a single resource is the GRIS, and a collection of GRIS's for a VO can be aggregated as a GIIS. This aggregated information service is considered as a user-level service, and covering the next section.

- The third pillar is Data Management and provides facilities to access and manage data in a Grid environment. These are the facilities provided by GridFTP service.

  The GridFTP service provides a secure way to transfer files in a Grid environment.
  It extends the standard FTP protocol and supports third-party control of data transfer.
  Thus, files can be moved directly between servers while the user controls the transfer from a third machine.

User-level Middleware

Another component of the MDS is the Grid Index Information Service (GIIS). This is an aggregated information service. An individual GRIS is responsible for registering itself with a GIIS. Each individual GRIS contains static and dynamic information for the resource it is monitoring. The GIIS consequently gives a picture of static and dynamic information across the resources it is federating. This is a powerful concept. For example, it allows a client to query the GIIS for details of resources matching hardware requirements, and also dynamic information criteria, such as current CPU load.

Higher-level resource scheduling and allocation across multiple nodes can be handled by some Grid schedulers such as Nimrod/G and Condor-G which are both integrated with the GRAM and MDS modules of Globus.

Additionally, the DUROC Globus component can fulfill a similar role. Much like the GRAM service, the DUROC accepts an RSL request, detailing resource requirements. DUROC furnishes the RSL with details of resources satisfying the criteria, and then allocates the multiple resources.

Grid Programming Environments and Tools

Due to the 'bag of services' approach taken by Globus, and the availability of technologies, this is an area where Globus is particularly strong, and where a wide choice is available to the developer. In many cases, this has involved the 'Globus enabling' of familiar programming environments.

There is a version of the Message Passing Interface (MPI) operating over a Grid, as implemented by MPICH-G2. As discussed in [25], while MPICH-G2 can be considered as a high-level programming model, it does offer a number of desirable features. Not least, it is a compliant implementation of the MPI standard, and as such parallel programs developed using MPI should be able to run using MPICH-G2 without excessive modification.

It is also possible to link to the Globus libraries and develop applications directly using the Globus API's. The Globus Commodity kits provide a higher-level application level API for developing an application integrating available Grid resources, using common programming languages such as Java or Python. This application level API allows the construction of workflows of Globus services using familiar high level programming languages.

The Grid Portal Development Kit (GPDK) [44] is layered on the services of the Java CoG kit and allows the construction of web-based Grid portals.

Grid Applications

Computational science portals are emerging as useful and necessary interfaces for performing operations on the Grid. The Grid Portal Development Kit (GPDK) facilitates the development of Grid portals and provides several key reusable components for accessing various Grid services. A Grid Portal provides a customizable interface allowing scientists to perform a variety of Grid operations including remote program submission, file staging, and querying of information services from a single, secure gateway. Still, Globus is middleware, and creating an application that uses Globus, or "Globus-enabling" an existing application, requires a lot of work.

UNICORE

The UNICORE software is a well established Grid solution which allows users to design complex componentied High Performance Computing jobs [32]. Using a GUI interface to capture the workflow of the job from user; the user can then run this job at any of their accounts on multiple, heterogeneous UNICORE-enabled resources.

Key features of UNICORE are:

- Full control over the jobs through a graphical user interface
- Construction of a workflow of tasks described in an target system independent form
- Abstraction of system functions, commands, and user actions to achieve system and installation independence
- The automatic staging of files
- Job monitoring and steering through the UNICORE client
- A plug-in architecture providing custom-built interfaces for individual applications

The UNICORE Gateway component authenticates connection requests through application and server certificates, and can co-operate with firewalls comfortably. Support for file transfer using GridFTP is added using results from the GRIP[18] and EUROGRID[9] projects.

UNICORE can be alternatively described as possessing a 'stovepipe' architecture, or as a vertical integration environment for a Grid. The most 'visible' component of UNICORE is the graphical user interface, and it is the "World Wide Web browser" of the UNICORE Grid. Additionally, the remainder of the vertical architecture comprises of security filtering and middleware server components.

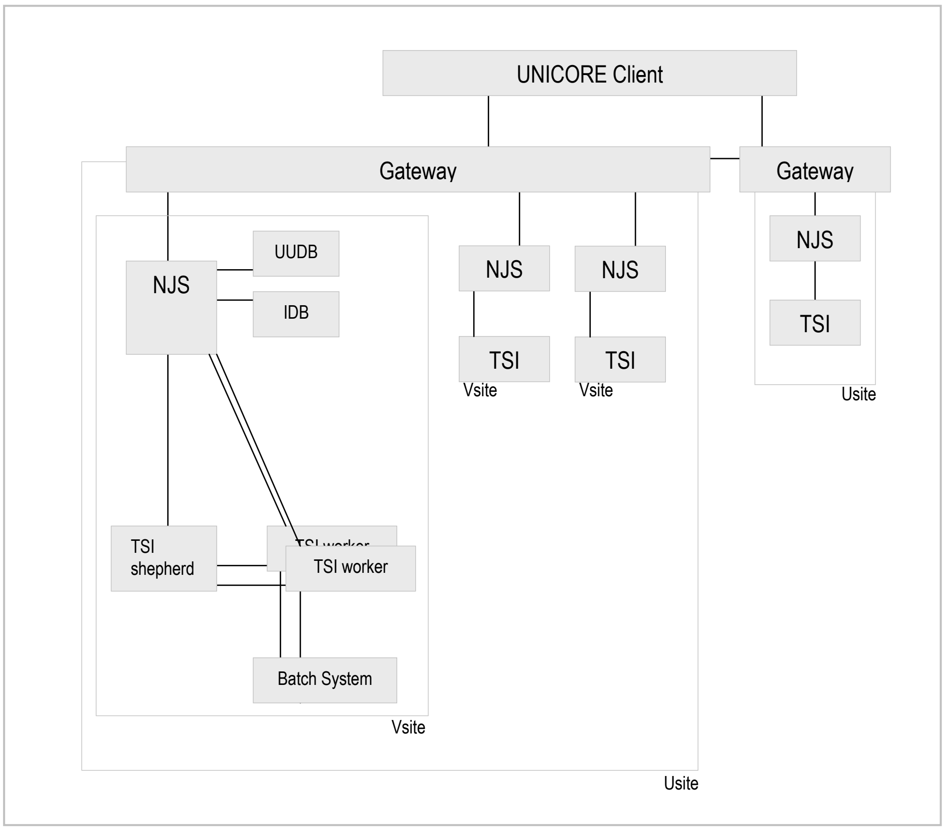


Figure 6: UNICORE architecture

The UNICORE architecture is also shown in Figure 6. This is shown in the form of a typical configuration. As will be discussed below, for a typical UNICORE Grid deployment, each administrative domain is protected by a Gateway component, and the name given to this collection of resources is a Usite (UNICORE site). A Usite can contain a number of Vsites (Virtual site). The Vsite on the left in Figure 6 shows a more detailed view of the components which are present.

The UNICORE client constructs an abstract workflow of jobs to be submitted, and assigns each job to a (possibly difference) machine. This job description is in an abstract form called an Abstract Job Object (AJO). This is submitted to a Network Job Supervisor (NJS) component, which is responsible for *incarnating* each job into a form which can be executed of on the specified target system, and for distributing other jobs in the workflow to other Vsites.

Following the Grid architecture of Figure 1, we can describe how the UNICORE components fit into the various layers of Grid architecture [33]. Due to the stovepipe nature of the UNICORE architecture, its components can be found in most of the layers.

Fabric

The Target System Interface (TSI) is the UNICORE component present in the Fabric layer of the architecture. This receives commands from the NJS, which are in their incarnated form and ready for execution. The TSI is implemented as a shepherd process which forks a new process for each job submission. It interfaces with the batch systems present on the target system, and provides job monitoring information to the NJS and also arranges data transfers.

Security Infrastructure

The UNICORE security infrastructure is based on X.509 certificates, which are used to sign a job to be submitted. The temporary proxy certificate created by the GSI of Globus is not used with UNICORE. The resources to be used by a multi-site workflow are specified and signed by the user *before* submission.

The first UNICORE component dealing with security is the gateway. Before we discuss its operation, we emphasize that this aspect of the UNICORE architecture does not fit in neatly with our scheme for discussing Grid architecture, due to the differences between a security policy enforced at the resource itself, or having security policy enforced by a single component for a number of resources (usually all the resources in a local network). Specifically, *authentication* occurs at a 'door-man' service called the Gateway, whilst *authorization* occurs at the resource itself.

- Authentication
  The task of the UNICORE Gateway component is to act as a 'filtering' service for authenticating requests of the UNICORE Grid. This component ideally sits within a Demilitarized Zone (DMZ) around a network protected with a firewall, or similar. This Gateway component effectively acts a filter, assuring that requests from clients can pass through, if the certificate used to sign the job is issued by a recognized Certificate Authority.

- Authorization
  The second security activity performed by UNICORE occurs in the NJS. Using the UNICORE User Database (UUDB), the credentials of a signed job are checked for authorization permission. The UUDB maps the subject obtained from the users certificate, on to a Unix account. In contrast to the Gateway, this procedure occurs at the resource itself.

The differences between the Globus security based on credential delegation and the model used by UNICORE is briefly discussed in [31], "… the need for delegation depends on the underlying concept of the site-integration. A trust model based on a strongly connected web of trust as used in the UNICORE project would allow to avoid the general delegation".

Thus the extent of the 'web of trust' is defined by the UNICORE Gateway component. Once a request for a resource has passed through the Gateway component, the user must be configured in the each UUDB to access each resource. In this sense it can be said that the UNICORE model allows some limited form of credential delegation.

Core Grid Middleware

As well as its higher-level role directing components of a workflow to various Vsites, the NJS is also responsible for marshalling an 'atomic' job unit, through the incarnation process, and on to its execution at the TSI, and subsequent monitoring of the execution. Referring again to the hourglass model of Grid architecture, this aspect of the concepts of UNICORE functionality corresponds to the neck of the hourglass in the model.

UNICORE goes a long way to achieve the aim of seamless access to computational resources. Seamlessness is achieved at two levels. Firstly, there is the issue of diverse scheduling mechanisms on different resources. UNICORE offers a layer of abstraction on top of this, and this is also true of Globus. Secondly, the IDB can contain site or system specific information, for example compiler options, specific software resources, etc. Through a process of incarnation, Abstract Jobs are then translated in concrete commands for each target system.

User-level Middleware

Using the UNICORE client, a job workflow is defined which is a description of a group of interdependent jobs to be performed on a collection of sites. A UNICORE job, or more precisely job group, is a recursive structure which can contain other job groups.

During the process of submitting the job the UNICORE client constructs an Abstract Job Object (AJO), from the job description. The AJO can include of other sub-AJO's. As its name implies, the AJO is abstract, and does not include site or system specific information, e.g. paths to executables, etc. The client submits the AJO to the destination of the highest level AJO (the NJS of the primary Vsite).

The responsibilities of the NJS places both, the core and user-level middleware categories. The NJS of the primary Vsite examines the received AJO and sends any sub-AJO's to their respective destinations. This functionality of the NJS is to act as a type of resource aggregator, dispatching jobs to various UNICORE Vsites, and as such classifies this component of the NJS as user-level middleware. The other role of the NJS is only concerned with the resources on the particular machine that it is supervising, in other words this part of the NJS is fulfilling the role of core middleware.

Grid Programming Environments and Tools

With UNICORE there was a deliberate intension not to support tightly coupled meta-computing or application level development explicitly. When initially conceived, the target users had existing applications that did not use Metacomputing environments. Moreover, these applications were frequently provided by third parties.

The UNICORE plugin mechanism provides some measure of a programming environment for the underlying Grid. However it is quite limited in its ability to let the programmer directly interact with the Grid infrastructure underneath it. However, in the UNICORE Plus project [34], [57] work was done to integrate PACX-MPI [55] into UNICORE. The integration was assisted through the use of a client plugin. During the UNICORE Plus project some work on a super-scheduler for UNICORE working on CCS was also undertaken.

On the whole due to its original motivations and design of the system, support for meta-computing, and provision of a programming environment for a UNICORE Grid, are areas where UNICORE is weak in comparison to some other Grid middleware and Globus in particular.

Grid Applications

The server-side UNICORE components form the UNICORE Grid. The UNICORE client allows remote access to the UNICORE Grid, with authorized access provided by the UNICORE Gateway component.

Whilst the UNICORE model from the perspective of the client conforms to the 3-tier of portal architecture (the middle tier is the Gateway and NJS services, and the underlying Grid resources form the bottom-tier). However, UNICORE uses a 'thick' java application client, instead of a HTML based client. The advantage of this approach over a thin-client, is that more functionality can be pushed to the client, resulting in less communication with an application server. The result is also that the client can be used for job preparation whilst offline.

There are a number of layers of 'virtualization' present in the operation of the UNICORE client. The lowest level of virtualization is witnessed in the submission of batch scripts through the UNICORE interface. This is provided to allow existing applications to be integrated instantly into a UNICORE job flow. A medium level of virtualization is enabled by the use of the abstract task descriptions which are incarnated for each target platform, such as the compile task. Finally, a high level of virtualization is achieved through the principle of site abstraction in conjunction with the use of custom-built interfaces supplied as plugins.

It can be said of UNICORE that there is a tight-coupling between the various layers of the architecture (for example, storing the job workflow as java serialized objects). For comparison, taking for example Globus, each service tends to use different but well-known protocols at their interfaces. Furthermore, the issue of closed communication channels between the components is more evident in UNICORE due to its layered stovepipe architecture, whereas the horizontal architecture of Globus has a single layer providing homogeneous access to underlying resources at a single layer in the Grid middleware.

However, on this point it is important to note that, as with other Grid software, UNICORE is an evolving Grid infrastructure. The original UNICORE project and its follow-on project UNICORE Plus [57], have both completed. For the most part, the scope of this initial project was to architect, plan and implement the core UNICORE software, which was achieved.

Currently, important initiatives regarding on-going development of UNICORE include the EUROGRID and GRIP projects. EUROGRID aims to provide a European Grid network of leading HPC centers, deploying the UNICORE software. Included in the work packages are enhancements to the software in the form of GridFTP integration as an alternative file transfer mechanism for UNICORE, and the development of a resource brokering functionality. Additionally, the project aims to demonstrate the feasibility of the proposed work through the integration of a number of large scale applications in areas such as bio-molecular simulations and weather prediction, into the UNICORE system, including the development of UNICORE plug-ins to the client, providing user-friendly, custom-built interface to the application, within the UNICORE graphical user interface. During the course of this project, the maturing of the core UNICORE environment has been witnessed, and its competency satisfying the original aims of the software demonstrated.

The GRIP project implements interoperability between UNICORE and Globus, at the middleware level. The intention is not to embed the Globus services and protocols throughout the UNICORE architectural stack. Instead, the aims of the project are to provide interoperability with the minimum of interference to the existing systems. The integration occurs at the 'integration' layer of the two architectures. Namely, the number of potential UNICORE target systems is enlarged by providing access to Globus resources, through the deployment of a special Globus Target System Interface (TSI). Briefly, the mechanism of this

Globus TSI is: the job is specified in the client as a script task, and this is submitted. After the abstract job description is incarnated in the NJS and handed to the TSI for execution. At the Globus TSI, there is a module translating UNICORE requests for job submission, output retrieval, and status queries to the corresponding Globus constructs. Additionally, there is the issue of different security mechanisms in Globus and UNICORE. This is solved by installing a small plugin into the UNICORE client to initialize a Globus proxy credential. This is passed to the Globus TSI through the secure UNICORE communication channels.

Through the standardization of Grid middleware which is currently receiving much attention, UNICORE stands to benefit greatly from 'opening up its interfaces' through adoption of relevant new standards, but without losing sight of the merits of the UNICORE architecture. The results from the GRIP project are proving to be instrumental for the future evolution of UNICORE. Central to the focus of the project is the participation and contribution to the activities of the Global Grid Forum (GGF). Of direct importance to the project are information services, resources management, security, applications, computing environments, and Grid architectures. One of the results of the GGF process is the Open Grid Services Architecture (OGSA). Exposing the functionality of the UNICORE system through web service interfaces and following the OGSI specification, is the instinctive path for GRIP to follow in order to enable further interoperability, which is indeed a core topic for the focus of the project.

AVAKI

AVAKI is an object-based operating system for Grids which was started in 1993. AVAKI originates from the Legion project from the University of Virginia. Following commercialization of the software, AVAKI is now the corporate distributor of this product, and all future development and marketing of the software occurs through AVAKI.

AVAKI attempts to hide the complexities of distributed Grid resources (scheduling, data transfer, communication, etc), from the user through adding a layer of abstraction, creating the impression of global virtual computer. It addresses the technical and administrative challenges faced by organizations such as research, development, and engineering groups with computing resources in multiple locations, on heterogeneous platforms, and under multiple administrative jurisdictions.

AVAKI provides three different software products: the "AVAKI Data Grid", the "AVAKI Compute Grid", and a combination of both - the "AVAKI Comprehensive Grid". In this report we only examine the AVAKI Compute Grid as we mainly want to consider GRID computing environments.

AVAKI Grid software is composed of three services (Figure 7):

- The Grid Protocol layer, which provides protocol adapters, security, and naming and binding.
- The Systems Management Services layer, which provides capabilities for implementing and managing distributed solutions.
- The Applications Services layer, which provides high-level services that can be used to construct file sharing, collaboration, and high performance computing applications.

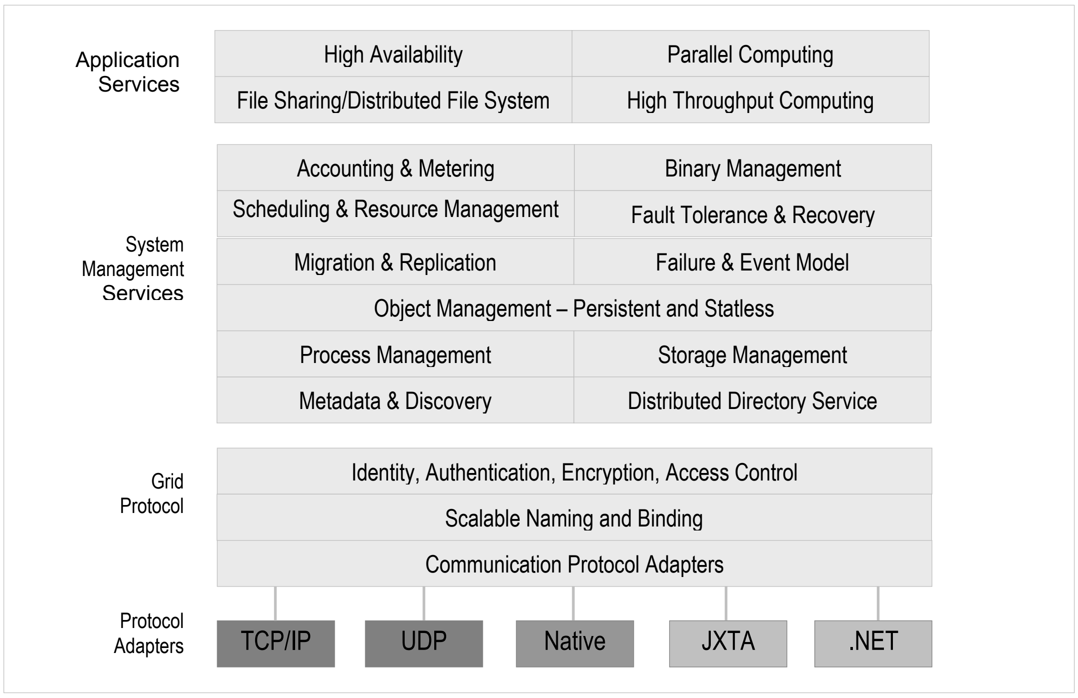

Application Services
High Availability
Parallel Computing
File Sharing/Distributed File System
High Throughput Computing
System Management Services
Accounting & Metering
Binary Management
Scheduling & Resource Management
Fault Tolerance & Recovery
Migration & Replication
Failure & Event Model
Object Management – Persistent and Statless
Process Management
Storage Management
Metadata & Discovery
Distributed Directory Service
Grid Protocol
Identity, Authentication, Encryption, Access Control
Scalable Naming and Binding
Communication Protocol Adapters
Protocol Adapters
TCP/IP
UDP
Native
JXTA
.NET

Figure 7: AVAKI Grid software layer

The AVAKI Data Grid as well the new version (Version 3.0) of the Compute Grid are written in Java to ensure a better portability to an heterogeneous environment.

Fabric

AVAKI interoperates with native file systems, creating a layer on top of the native file system in order to build a single uniform operating environment – the AVAKI Grid.[48]. For the user this means a simplification of the process of interacting with resources in multiple locations, on multiple platforms, and under multiple administrative domains. For instance, the user can access files by a virtual name in a virtual directory. He does not have to know the physical location of the file.

An AVAKI Grid can be built from individual computers and local area networks or from clusters enabled by software such as LSF (Load Sharing Facility), PBS (Portable Batch System) or SGE (Sun Grid Engine). If one or more queuing systems, load management or scheduling systems already are in place, the AVAKI software can interoperate with them. Thereby it allows sharing of resources across the cluster and creates virtual queues.

With AVAKI software users can execute their applications interactively or submit to a queue. Queues allow to group resources in order to take advantage of the shared resources mainly computing power. The AVAKI queue server provides multiple logical queues and allows the configuration of different options, such as number of jobs running at once, number of times to try to restart a job which has failed, and the number of jobs within a queue. It also supports the selection of an appropriate scheduler for the queue, selection of priority of a job within a queue and purging and killing of jobs currently running in a queue.

Security Infrastructure

The security of AVAKI is a result of several separate capabilities that work together.

As users access files, run applications, or submit jobs to queues, AVAKI authenticates the resources and ensure that the requested procession is allowed. The authentication is based on Public Key Infrastructure (PKI). Each resource made available on the AVAKI Grid has a unique identifier that is related to but independent of its user-visible name. The unique identifier of the resource is also independent of its physical location and it is used to authenticate resources.

The built-in encryption and message digest feature are used to ensure message integrity.

To control the access to resources on the Grid AVAKI provides a fine-grained security that local administrators can use to control access to their resources. The configuration of allowing/denying of user-access respectively resource-access is supported. Access to individual resources, including data, applications, queues, and hosts, is controlled individually. Files can be associated with multiple user groups, users can define user groups without administrative intervention, and per-file exclusion lists (allow-all-but/deny-all-but) are supported. Through the same mechanism, administrators control the access privileges of resources that use other resources, such as applications that use data files, or queues that run applications and use the processing power of hosts.

Core Grid middleware

The AVAKI system runs on top of the unmodified operating system of each participating machine in the AVAKI grid. Therefore it does not need to manage very low-level resources. The underlying local operating system does that. At the AVAKI level, the resource base consists instead of multiple heterogeneous processors and storage devices.

AVAKI supports various types of heterogeneous platforms to act as a Grid service server: HP (Tru64 Unix), IBM (AIX), Intel (Red Hat Linux, Microsoft Windows 2000, Microsoft Windows NT), Sun (Solaris) and SGI (Irix).

Grid Programming environment and tools

AVAKI is structured as a system of distributed objects. All of these entities are represented by independent, active objects that communicate using a uniform remote method invocation service. This approach enables the interoperability of the components between multiple programming languages and heterogeneous execution platforms.

Grid applications

Most applications that use the AVAKI Grid can be written in any language and do not need to use a specific API. They can be run anywhere on the AVAKI Grid resources without source code modification as long as resources are available that match the application’s need.

To take further advantage of distributed processing power, AVAKI’s queuing facility also supports two forms of parallel program execution:

- AVAKI provides a simple support for performing many executions of a single application, each with different parameter data. This doesn’t include dependencies or work flow elements. Each of these runs is called a “job”. These jobs do not communicate with one another, i.e. they can run in parallel.
- AVAKI supports existing code written in MPI with a “native MPI wrapper”. In this case it makes use of a preinstalled version of MPI on the target system.

Currently there is no AVAKI MPI Grid implementation to support running a distributed job on heterogeneous platforms.

TME

TME (Task Mapping Editor) is a visual programming tool developed at the Center for Promotion of Computational Science and Technology at the Japan Atomic Energy Research Institute. The tool facilitates seamlessly integrating parallel programs distributed over networks into a single meta-application. It provides an intuitive graphical user interface to specify relations among programs and identify computing resources. Based on the specification, the execution of programs is automatically controlled.

Currently, two kinds of meta-applications are implemented on TME: a risk management system for environmental crisis and a distributed parallel data analysis system for nuclear fusion plasma. They allow performing of distributed parallel processing without being conscious of the underlying computing resources.

The TME software is available for the following target platforms:

Vector Parallel Computer:	Fujitsu VPP300, NEC SX-4, Cray T90
Scalar Parallel Computer:	Fujitsu AP3000, Hitachi SR2201, SR8000, IBM SP2, SP3, Cray T3E, IntelParagon
WS Server:	Fujitsu GP8000S, HP9000, NEC TX-7, Sun Enterprise Server, SGI Onyx
WS, PC Cluster :	Compaq, HP, Sun WS, Linux Cluster

The communication infrastructure as well as the security environment of TME currently based on the ITBL system. Therefore, the following description of the various layers of the Grid architecture [33] of TME concern also ITBL infrastructure.

<u>Fabric</u>

TME consists of 4 kinds of subsystems (see figure 8): the TME-GUI, the Meta-scheduler, the Resource Information Monitor (RIM), and the Execution Manager (EM).

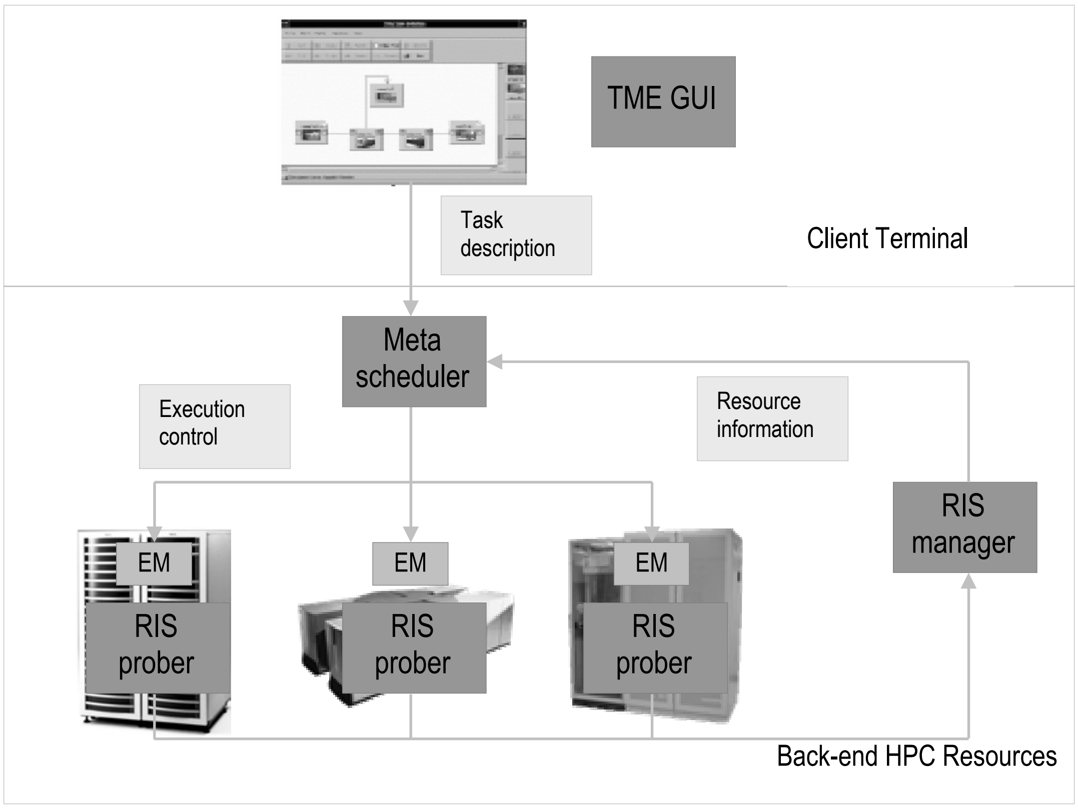


Figure 8 Software architecture of TME

The Meta-scheduler resides on one of the backend computing resources and communicates with users through the TME-GUI. Main tasks of the Meta-scheduler are the determination of the execution order of components according to the parsed task description and the control of the execution of a meta-application. When a user asks TME to select a suitable computing resource for a component, the Meta-scheduler decides the target computer with the help of RIM.

RIM (Resource Information Monitor) handles the information on computers and networks. In order to gather resource information of each computer, RIM servers reside on each computer to monitor information. They send the monitoring results periodically to the RIM client, which stores the information and provides it to the Meta-scheduler or users. RIM treats two kinds of resource information, which are (i) static information such as CPU performance, size of memory, and type of hardware architecture of computers, and (ii) dynamic information such as latency and throughput of network, loads of computers, and the number of waiting jobs in a batch queue. In addition, RIM predicts future resource information as well. At present, two kinds of methods are implemented on a RIM prototype system. The first returns the most recent past resource information. The second provides more accurate information by searching the most similar sequence of resource information from the gathered data.

EM (Execution Manager) is created on each target computer by Meta-scheduler to stage the execution of a meta-application under the control of Meta-scheduler, that is, receive input data for a component, execute a program, check a program termination, and transfer output data. When the program is to be executed in a batch mode, EM generates a batch script based on the information, which the user specifies in the registration phase. At present, EM has an interface to submit a job to NQS.

Core Grid middleware

TME is currently one of the tools implemented on the ITBL (Information Technology Based Laboratory) system. The objective of this project is to establish high-speed networking supercomputers to be distributed in different research organizations so that computational resources like software, research databases and computational power can be shared by all organizations in Japan. The ITBL project is the follow on of the STA (Seamless Thinking Aid) basic system. This is a software system for supporting a series of works on distributed scientific and technological computations by providing a work environment to enable seamless thinking of users. The STA system provides both communication infrastructure for supporting communications among heterogeneous computers and several computational tools integrated on the communication infrastructure, so that a user can work just like on a single computer without noticing the presence of multiple different computers.

Figure 9 shows the architecture of the ITBL system. It is divided into several layers. In order to realize secure and easy access to resources, a single sign-on authentication mechanism and protocol changeable communication infrastructure on the VPN-based secure network exist. Various kinds of primitive services such as process manager, task scheduler, resource monitor, data manager are implemented on the foundation. In addition, the system will provide tool kits which interact with users and control the underlying primitive services in place of the users. There are two kinds of tools, a tool kit for component programming and a tool kit for community access. The former supports the development and execution of applications on the ITBL system. The latter supports sharing and exchanging of information among researchers. These tool kits are the most important for users, because they become an access interface to the ITBL system.

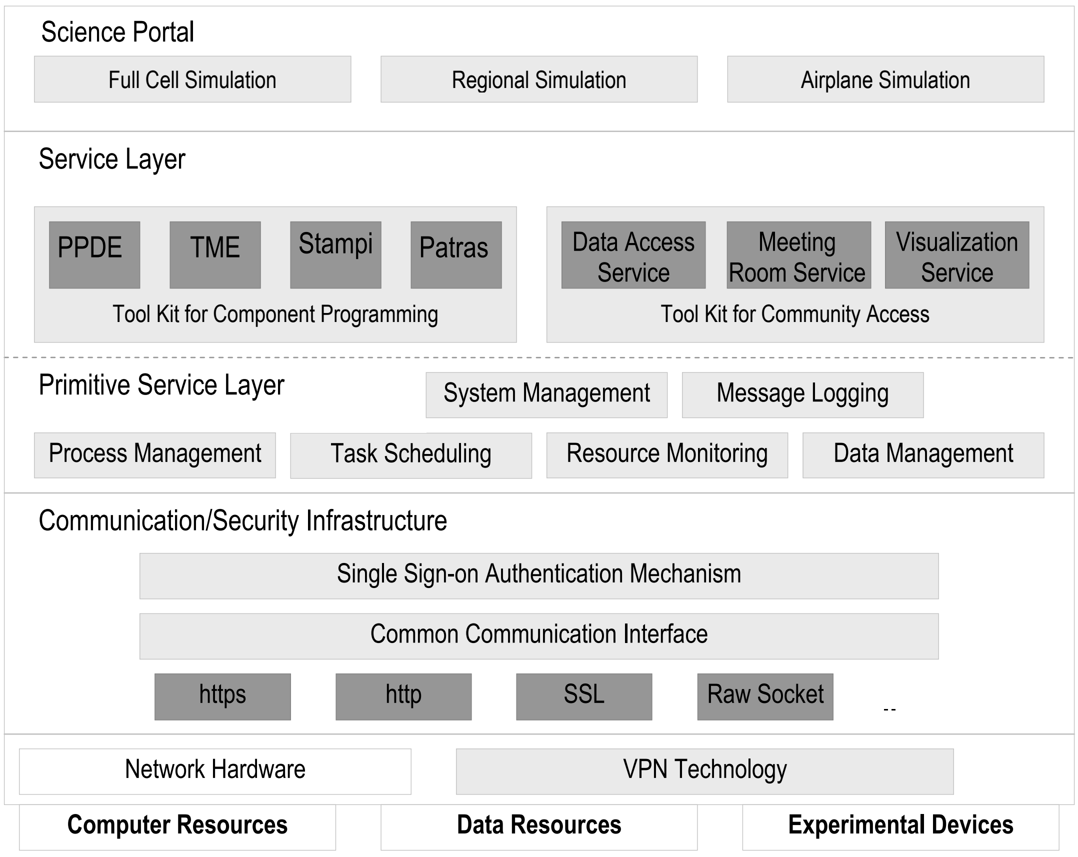


Figure 9: Software Architecture of the ITBL system

<u>Security Infrastructure</u>
As mentioned before TME makes use of the ITBL security infrastructure. The security issues of ITBL comprise from mainly three parts:
i) https and SSL based secured connection
ii) X.509 based authentication
iii) DMZ (DeMilitarized Zone)

The ITBL system assumes that normal users access the ITBL server from the outside of an institution, in other words, normal users come from outside LAN. Thus ITBL should take care of the access through the firewall. Therefore, an ITBL (front) server is running on the DMZ, which is one of sub-networks connected to the firewall and DMZ can be also treated as outside from LAN. Inside the LAN where computational resources (supercomputers, databases are located), relay processes are running on the ITBL relay server, which directly relays the messages from any machine inside LAN and the user client outside the LAN.

In order to protect from an attack by an intruder, authentication on the ITBL server and computational resources are completely separated. That is, two-step authentication is required. In connection to other institutes, the same procedures is required, but automated behind the ITBL server. The user does not have to take care of multi-step authentication rather then it is realized as the single sign-on procedure. Any user information regarding the ITBL system is restored on the database server which is physically connected to the ITBL server, therefore significant information cannot be referred from any machine even though inside LAN, but only from the ITBL server. Security is assured by network configuration at first.

For all connections between client and servers, and ITBL servers and proxy (or relay) servers, authentication is carried out, and X.509-based certificates are required on the mutual (cross) checking. Thus the authentication on the ITBL system is redundant and almost all accesses from outside can be considered as valid connections.

Between proxy servers and computers which are allocated inside the LAN, a communication API is implemented using the nexus library developed by Argonne National Laboratory, and its security depends on the local policy and sometimes it is assured by SSL and SSH (but this is optional).

Core Grid Middleware
All communication APIs are provided from the Starpc library, which is originally developed by JAERI. Starpc [48] is a RPC (Remote Procedure Call) based communication library for a parallel computer cluster. It supports communication between Java applets and C programs as well as C programs. It has the following features. (a)It enables communication between Java applets and C programs on an arbitrary computer without security violation, although Java applets are supposed to communicate only with programs on the specific computer (Web server) in subject to a restriction on security. (b)Diverse network communication protocols are available on Starpc, because of using Nexus communication library developed at Argonne National Laboratory.

User Level Middleware
On the TME canvas, data files and programs are expressed as icons. By linking program icons and data icons, users can easily specify data dependencies and execution order. Moreover, users can choose computers to conduct programs interactively.
The visual specification of an application is translated into a task description. The description is sent to the Meta-scheduler which decides the execution order of the components. When a component is executed, the meta-scheduler selects the target computer with the help of RIM which provides information about computers and networks. In order to gather information, the probers reside on each computer to monitor information. They send monitoring results periodically to the manager. Finally, the manager provides it to the Meta-scheduler. After selecting the target computer, meta-scheduler makes requests to the execution manager on the target computer to allocate the task and stage data.

Grid Programming environment and tools
The TME_GUI is a web-based intuitive visual interface to integrate component programs. Users can specify relations among programs on a TME canvas. TME provides various ways to realize I/O linking: file transfer, pipe-lining and MPI I/O can be specified. For process creation, TME provides three kinds of mechanisms based on MPI semantics: spawning of processes, client/server connection and MPI-1 based start-up. The process creation as well as the MPI I/O is realized by the Stampi [47] communication library.

Grid applications
As mentioned before, TME supports the development and execution of a meta-application, which consists of many programs and data residing on distributed computing resources. It focuses on the scenario in which users integrate components by two kinds of methods: executing a parallel program on a parallel computer cluster, and connecting input and output of independent programs.

In order to realize a GUI for end users, TME provides a users interface to supply file selection mechanisms, parameter input methods, etc. To realize this GUI, the concept of a modular GUI was introduced. Application developer can construct a module network by using these modules to integrate their own application into the TME environment.

At run-time, TME manages the execution of the meta-application, which means automatically stage of date and execution of programs according to the user specification. After the execution of a meta-application the user can monitor the execution status through the GUI. TME provides a single view to check the execution status of all component programs.

Installation and Test of Tools

Introduction

The test installation for the systems under consideration was a cluster at HLRS, Stuttgart. In order to simulate a typical user environment, the client portion of the software was installed on a workstation in Forschungszentrum, Jülich. Additionally, this configuration allowed the evaluation of the software with respect to its cooperation with firewalls, and other issues arising from dealing with two separate domains. The TME environment was installed only on one workstation.

The platform used for the tests consists of a front node for interactive access (crossi.hww.de) and several nodes for execution of parallel programs. The cluster is a heterogeneous cluster consisting of 10 PIII nodes with 1GHz and 24 Xeon/PIV nodes with 2.4GHz and a small number of test nodes.

We also wished to conduct some performance tests and comparisons between the systems in the test environment. However, this proved to be quite difficult. Firstly, we cannot assume a consistent environment with which to conduct the tests; the network between Jülich and Stuttgart is unpredictable and we could not be assured of a consistent environment on the cluster. Secondly, the structure of the systems makes them difficult to derive tests producing comparable results. For instance, both Globus and AVAKI support console access, whereas the graphical interface of abstract which must be accounted for.

We installed simple test applications on the cluster, and we interfaced all the systems to this test application. We used this application to test the usability and function of the systems. It was decided however that timing accessing this application through our test Grid systems would merely result in measuring the performance of the cluster and the test application, and not of the Grid middleware.

From the results of the questionnaire, parallel jobs as well as single batch jobs are both important, but we did not explicitly address requirements for Grid-enabled MPI (such as MPICH-G2). In fact, when submitting single jobs or 'local' MPI jobs to a cluster or a supercomputer, the overhead of adding the additional layer(s) of Grid middleware in terms of performance is usually negligible in comparison with the run time of the job. From the questionnaire, we can conclude that Meta-computing across multiple super-computers, is of limited appeal. Indeed, this is not really covered by UNICORE, and AVAKI.

Additionally, in some systems it is difficult to establish when a submission has completed. For example, in UNICORE, the client informs itself of the completion of a job by regularly polling the NJS. Clearly, there is some lag between task completion and the user being informed, however, using the approach of polling enabled good co-cooperation between UNICORE and firewalls. Similarly, in the alpha versions of the Globus 3 toolkit, the job manager service appears extremely slow for the submission of trivial jobs, and again this is due to polling frequency.

Tests were derived and performed submitting jobs to the test environment. We wished to time the period from submission of the job to its completion. However, as we discussed, it is difficult if not impossible to specify tests and conditions which would fairly compare all the systems under test. However, we felt that it might be possible in some cases to compare file transfer times. This is still difficult to achieve in UNICORE (due to polling for results). However, for console based scripts in Globus and AVAKI, it is possible to time the submission of a simple job, including a file transfer, until a response is received at the client.

UNICORE

The UNICORE installation at HLRS (see figure 10) was used to accomplish the analysis of the Grid system. Currently the UNICORE 4.0 release is in operation and configured for 3 target systems NEC SX5, VOLVOX (cluster of SMPs, 10 nodes NEC Express Server 5800) and AZUSA (cluster of 16 x Intel Itanium C0). Most of the tests of the UNICORE system were performed with release 4.0 using Java version 1.4.

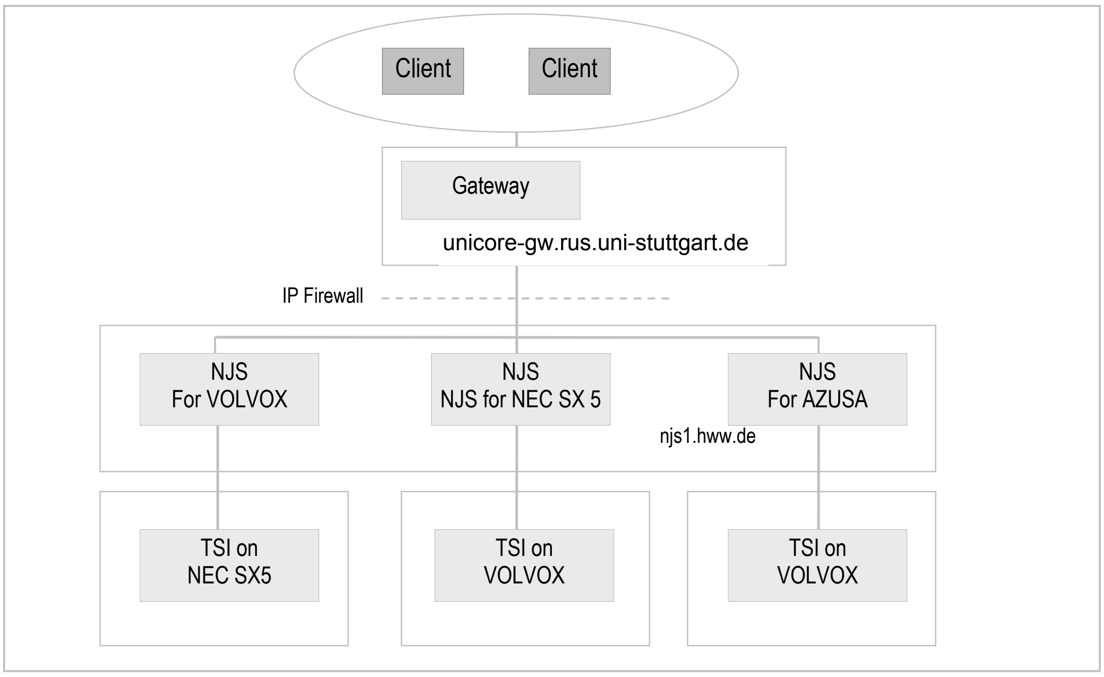


Figure 10: UNICORE Installation at HLRS

To enable the work with the HLRS firewall one dedicated port between the Gateway machine and the NJS machine had to be opened by the firewall administrators. On the Gateway machine IP filters were installed to adjust the access to this machine and to avoid attacks from outside. Only one port is open for any client outside the firewall and used to establish the SSL connection between these clients and the Gateway.

The Gateway software was installed on a SUN Workstation running SunOS 5.7. There were no problems during installation and with the reliability of the software. The NJS software component was also installed on a SUN Workstation and worked correctly. For each target system an individually NJS was created running on the same machine. The setup of the different IDB's (Incarnation DataBase) had been done together with the system administrators of the target platforms. The UNICORE Gateway has been integrated into the automatic startup/shutdown of the underlying operation systems. The automatic startup of the NJS component is not possible due to the necessity of a tty device within the startup phase of the software. Therefore the NJS has to be started interactively by the system administrator.

The UUDB (UNICORE User DataBase) is also running on the NJS machine. The integration of the user certificates was supported by scripts and worked fine.

The installation of the TSI (Target System Interface) on the Linux cluster (crossi.hww.de) was no problem. We used the TSI version written for Linux clusters using PBS with no. If the UNICORE system should run in privileged mode, the TSI processes had to be executed as root. Only in privileged mode the user mapping will work at all. If the system is running unprivileged (not in root mode), all UNICORE jobs will execute under a special user account defined in the IDB.

The installation of the client software on two PC's running WindowsXP and RedHat Linux also worked fine since Java 1.4 was preinstalled. A detailed integrated online help for the client is available and was helpful for the first time use of the client. After starting the client a keystore had to be created which contains all certificates. This process is supported by the UNICORE client.

With the help of intuitive graphical user interface of the UNICORE client we prepared a job consisting of two sub-jobs including a dependency to run our application. We used the *Command-* and *Script-Task.* The file transfer for input and output files was included into the sub-jobs. After successful submission of the job, we could check the status of the job at the resource panel. This is not an automatic task and has to be done individually by the user. An integrated "*WatchDog*" tool helps to automates this task if necessary, for instance for long time jobs. Sometimes there were problems to refresh the resource list. To fix this, we removed the `resoureCache.bin` file from the UNICORE home directory.

During run-time of the job some logging information can be fetched also to analyze errors. The level of information can be configured by the user. After successful run, we fetched our output, this means we could look at `stout` and `sterr` and the output files were transferred to our workstation.

GLOBUS

For the purposes of the evaluation, Globus was installed on one of the nodes of the cluster at HLRS (Stuttgart). During the test phase of the project we installed and tested various versions of the Globus software as they became available. Initially we used version 2.2.4 and more recently we worked with version 2.4.0. With the introduction of version 2 of the Globus toolkit, the software has been distributed with the Globus Packaging Technology (GPT). We are aware that organizations such as the National Science Foundation Middleware Initiative [50] distribute selected Grid middleware. For our tests we obtained software from the Globus website [51].

There is reasonable agreement that older incarnations of the Globus software were often troublesome to install and configure. However for the recent versions of Globus this has greatly improved. We found that it was relatively straightforward to install specific Globus services on a particular resource, and to test this by submitting jobs from user's workstation. In order to install and configure Globus on a number of machines it would be preferable to automate this process to some extent. We also installed and used the Java CoG kit in order to integrate Globus services in a Java program, but we did not install any of the other higher level Globus programming environment and toolkits.

The project partners followed an internal installation guide targeted for Linux installations which described, step-by-step, the interactions at the command line. We also chose to run our own test Certificate Authority in order to quicken the process of administrating security credentials.

With the help of a document [52] describing the Globus firewall requirements we were able to configure the HLRS as well as the FZ Jülich firewall to work with Globus. Some specific ports must be allowed by the server site to allow clients to access the specific services. Additional an ephemeral port range must be open at both sites, to allow callbacks from Grid services to clients, and at server sites in order to allow connection to transient Grid services.

The firewall community does not like the usage of incoming connections to non-deterministic ephemeral ports. We constricted the ephemeral port range to 10 ports. To configure the Gatekeeper and Job-Manager to work with this port range we used the environment variable GLOBUS_TCP_PORT_RANGE. By setting this variable also on all clients the Globus libraries will only choose port numbers for controllable ports in that specified range.

The contents of these instructions are discussed briefly in the following. These should not be taken as a full installation guide. For more information please refer to the Globus website, [52].

- Firstly, the installation process can be done as an un-privileged user, for example 'globus'. When configuring the Globus services root access is required. Define a location in the file system where Globus is to be located. Unpack the gpt package in this location, and create another direction (for example, 'TK'), to hold the installed software. Also, the `GLOBUS_LOCATION` and `GPT_LOCATION` environment variables must be set. This procedure included the following key commands:

  ```
  tar zxvf $DD/gpt-2.2.5-src.tar.gz
  export GLOBUS_LOCATION=$ROOTDIR/GLOBUS22/TK
  export GPT_LOCATION=$ROOTDIR/GLOBUS22/gpt-2.2.5
  ```

- After building the GPT (using `build_gpt`) the GPT can be used to install the relevant software bundles. We found the 'globus-all' binary bundle to be the easiest to install, and this is followed by `gpt-postinstall`. Thus:

  ```
  $GPT_LOCATION/sbin/gpt-install globus-all-2.2.4-i686-pc-linux-gnu-bin.tar.gz
  $GPT_LOCATION/sbin/gpt-postinstall
  ```

- The following lines show an example as to how the `globus_simple_ca` is installed using gpt as an un-privileged user.

  ```
  $GPT_LOCATION/sbin/gpt-build $DD/globus_simple_ca_bundle-latest.tar.gz gcc32dbg
  $GPT_LOCATION/sbin/gpt-postinstall
  ```

- The gpt-postinstall will install the necessary CA files into a determined CA directory, for example, `/home/globus/.globus/simpleCA.`

- Next, the GSI must be setup. This must be done as root. Firstly, the globus environment must be initialized, using `source $GLOBUS_LOCATION/etc/globus-user-env.sh`. Additionally, you must ensure that the GLOBUS_LOCATION environment variable is set. Then issue the following command:

  ```
  $GLOBUS_LOCATION/setup/globus_simple_ca_dcdd709d_setup/setup-gsi –default
  ```

  Self-explanatory input from the simpleCA administrator will be requested when running the setup-gsi command. The –default option indicates that the simpleCA should be considered the default CA.

- Now we are at a stage where certificates can be requested. This is achieved by the following as root for a host certificate,

  ```
  grid-cert-request -host zam589.zam.kfa-juelich.de
  ```

  For a user certificate, a regular user can issue the following command,

  ```
  grid-cert-request
  ```

- The simpleCA owner must sign the generated requests, and then the requestor must move the signed credentials to their correct locations.

- Correct security credentials for the host should now be present. Next, the following command initializes the Globus GRAM job manager, which requires the credentials to be present:

  ```
  $GLOBUS_LOCATION/setup/globus/setup-globus-gram-job-manager
  ```

  Now, start the gatekeeper as root (a more permanent solution involves using xinetd)

  ```
  globus-gatekeeper -conf $GLOBUS_LOCATION/etc/globus-gatekeeper.conf
  ```

- In order for the user to be authorized at the GRAM service, the subject of the users X.509 certificate must be added to the grid-mapfile, along with the uid of a local account. A command line utility does this. For example,

  ```
  grid-mapfile-add-entry -dn "/O=Grid/OU=GlobusTest/OU=simpleCA-zam589.zam.kfa-
  juelich.de/OU=zam.kfa-juelich.de/CN=Roger" -ln roger
  ```

- Finally, back as user roger, we must first create a proxy certificate from the permanent certificate with grid-proxy-init. Following this we can submit a test script to the GRAM service, in this case at the machine zam589.zam.kfa-juelich.de.

  ```
  globusrun -o -r "zam589.zam.kfa-juelich.de" '&(executable=/bin/date)'
  ```

A similar process must be followed in order to configure the information and data management services.

AVAKI

At HLRS, the AVAKI Compute Grid version 2.6 was installed in a production like environment. Figure 11 shows the participating machines.

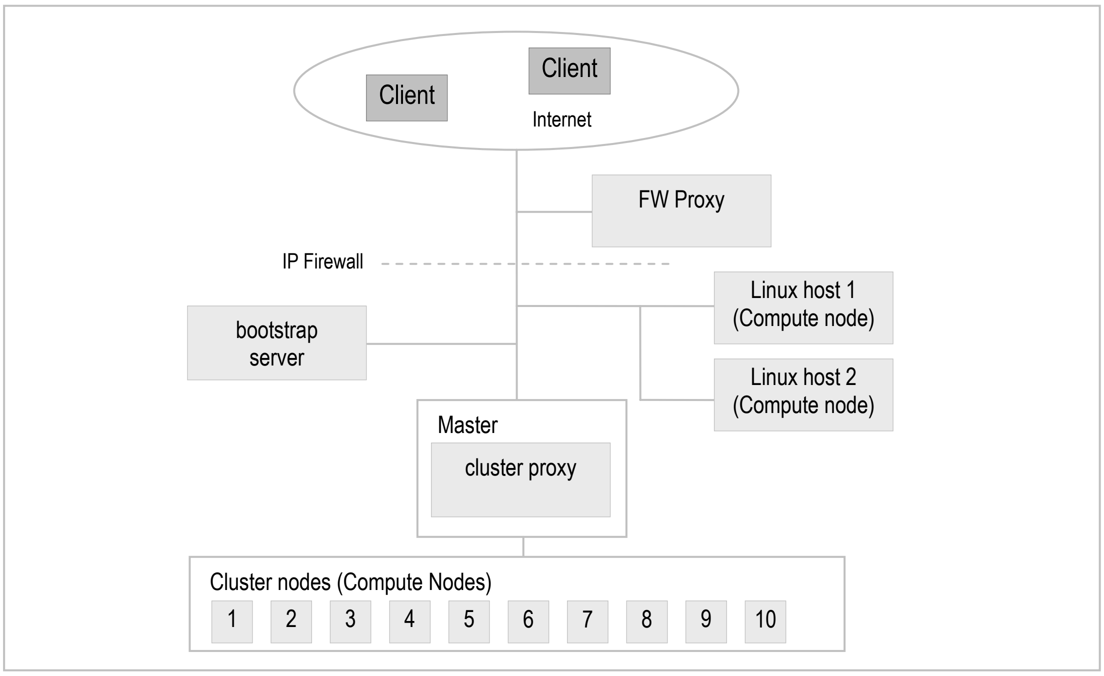


Figure 11: AVAKI installation at HLRS

During the installation process also one difficulty had to be solved. A firewall proxy had to be installed to get access to the target hosts through the firewall. This proxy tunnels the in- and outgoing signals of the clients from outside the firewall. To enable these connections the firewall has to be opened on one dedicated port on the *Proxy server* machine for all clients how want to use the AVAKI Grid system.

The following points describe the main steps of the installation process:
1. Creation of communication-files which contain information about the AVAKI Grid constellation (e.g. IP-addresses of *Proxy* and *bootstrap-server*)
2. Installation of the AVAKI software on all machines that will be part of the Grid and distribution of the communication-files
3. Starting of the system (1. *Proxies*, 2. *bootstrap-server*)
4. Adding computers to the Grid. To register clients and compute nodes to the AVAKI Grid each of these machines has to "join" the Grid.
5. Initialization of compute nodes as *grid service server*.

For the integration of a Third Party Queuing Sever an additional *cluster proxy* was installed on the front-end machine of the cluster. Each node of the cluster had to be initialized to the AVAKI Grid individually. After that, an AVAKI command that created some interfaces scripts and a special initialization process had to be run.

After successful installation the administration can log on to the Grid system and add users and specify security and utilization policies for the Grid, users, and computers.

To support MPI applications, the configuration of the so called "native MPI support" had to be done by the administrator. This means, first running a command to create a wrapper script for the MPI environment which should be used in the AVAKI Grid. AVAKI supports wrapper for MPICH, IBMMPI, LSFMPI, SGIMPI and SunMPI. Second, accomplish a special configuration command on one of the nodes which should be available for MPI jobs to integrate these nodes.

Sometimes it was hard to understand some problems which occurred during the installation process and runtime. For instance, after starting the bootstrap-server no other compute node could join the AVAKI Grid. There was no really helpful advice to detect the problem. If this problem was happen, the whole system has to be restarted, which means a time exposure of around 30 minutes. Another problem occurred after a complete new-start of the Grid system. The initialization process for each participation resource had to be redone. This process was very time consuming especially for the cluster nodes.

To run an application on the AVAKI compute Grid it is necessary firstly to register this application on the Grid with a special command. The user can specify which users/groups of users can access them. After that users log in, define application parameters and submit a program to run on available resources. Input data is securely read from distributed sources. Once an application is complete, computational resources are cleared of application remnants and output is written to one or many of the physical storage resource available in the Grid. The detailed help file (pdf) supports the user in execution all the steps.

TME

The TME installation at HLRS based only on the STA system, since the ITBL system was under developing at this point of time. It was installed on a SGI workstation. The following describes the installation process in detail:

1. Installation of Nexus-4.1.1. Nexus is a portable library that provides multi-threaded communication facilities within heterogeneous parallel and distributed computing environments.
2. Building of the Starpc (STA-RPC) library. This library is used for the communication between the web server and the backend machines.
3. Building of the control daemons and adapter programs running on the backend machine. They control the TME resources and invoke applications. Both communicate with a web-server and the applications. The adapters watch the state of application and passe some arguments and so on. The main control daemons are shown in Figure 12.
4. Preparation of the TME applet distribution files on the web-server. Finally, we kicked the proxy-daemon for the web-server to start it by hand. The system assumed that Apache or other http server is already installed on the machine.

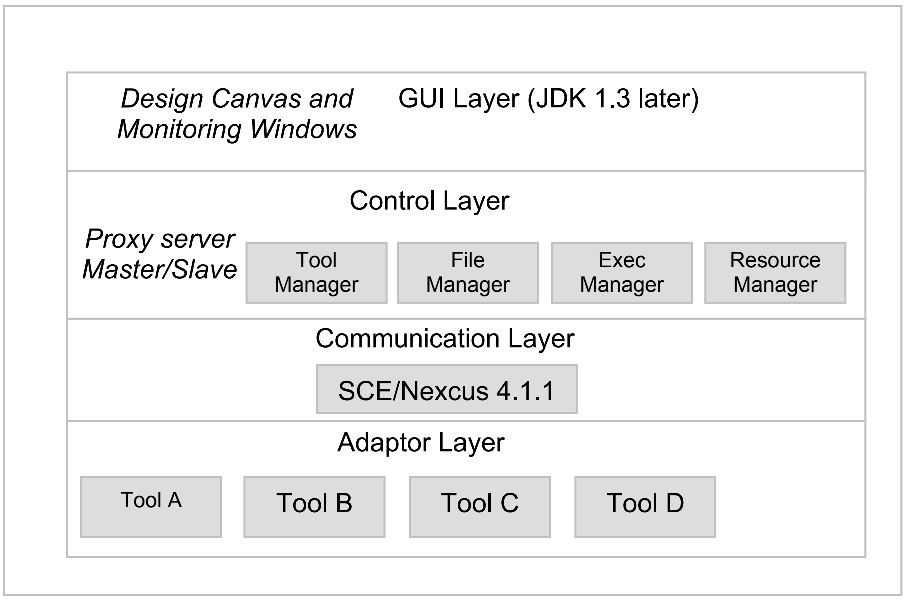


Figure 12: Layered structure of the TME software

On the client side a web browser is necessary to access the web interface of the TME system. After login on the STA system the TME GUI appears with the so called *Module Window*. In TME, the content of the parallel processing is defined by connecting programs and data. The programs and data shown in the form of icons are called “modules”. There are different modules to represent programs, data, network connections and applications.

The flow of distributed parallel processing using TME is as follows:
1. Importing programs and I/O files
   The user has to import programs and data files to be used into TME, and register them as modules in the *Module Window*. Some properties, such as file names and names of the computers storing them, must be defined when registering programs and data files.
2. Creating module networks
   The creation of network modules has to done by connecting registered modules with lines in the *Module Network Window*.

3. Setting up runtime properties of modules
   After creation of the modules the user has to setup runtime properties, such as the name of the host to execute the program, specification of tss/batch execution, the queue names used to execute batch processing, and the number of processors in the *Module Network Window*.
4. Instructing the execution of program module networks
   Starting of the execution of the defined modules has to be done in the *Network Window*.
5. Monitoring execution status of network modules
   Monitor the execution status of the executed network modules in the *Monitor Window*. There is the possibility to show execution logs, to cancel execution of selected modules or to delete them from a list of running network modules.

Summary of features available within the tested tools

In the following we have collected the main features of the analyzed systems. The first table gives on general overview.

	Globus	***UNICORE***	***AVAKI Compute Grid***	***TME***
Version	2.4	4.0	2.6	-
Distribution	Public	Public and commercial	Commercial	Public
Open source	Yes	Other: Community Source Licence	No	Yes
Documentation	Good	Good for installation and using of servers / Excellent for client	Good for installation /Excellent for admini-stration and using	Poor for installation / Good for client
Updates and roadmap	Yes	Yes	Yes	Yes
Supported client OS	Unix-Like	All Java	Windows/Linux	Web-based
Supported computing platforms	Unix Like	Hitachi SR 8000 IBM SP Fujitsu VPP series NEC SX series Cray T3E, SV1 IA32 clusters SGI O2000/3000, Onyx	HP IBM Intel Sun	Fujitsu VPP300, AP3000, GP8000S, HP9000 NEC SX-4, TX-7 Cray T90, T3E Hitachi SR2201, SR8000 IBM SP2, SP3 Intel Paragon Sun Enterprise Server SGI Onyx Compaq WS cluster HP WS cluster
Impact on owners of computational resources	Variable (interfaces to the local resource management system)	Variable (Perl scripts)	Small	Small
User interface	Command line client and GUI	Command line client and GUI	Command line client	Web-based GUI
API	Yes	Plugin-based interface	No	Adapters for application integration
Authentication	SSL	SSL	Yes (based on *unique identifiers*)	SSL
Authorization	Yes	Yes	Yes	Yes based on ITBL
Encryption	Yes	Yes	Yes	Yes

Table 1: Overview of the Grid system, supported platforms, interfaces and security issues

The majority of the systems are available in public domain, except for AVAKI which is a commercial product. The documentation of most of the systems is good for the client side, but sometimes the installation instructions leads to be desired. The majority of the systems operate under several versions of UNIX, including Linux. AVAKI also supports WindowsNT/2000. UNICORE provides a Windows client. The impact on the owners of the computational nodes is typically small. Most of the systems provide a graphical and a command-line user interface. For the AVAKI Compute Grid only the command-line is available. Regarding authentication mechanisms most of the systems use SSL and X.509 certificates. All systems support authorization and encryption.

	Globus	***UNICORE***	***AVAKI Compute Grid***	***TME***
Batch jobs	Yes	Yes	Yes	Yes
Interactive jobs	Yes	No, although some work is being done in the Eurogrid[9] project	Yes	Yes
Parallel jobs	Limited	Limited	Limited	Yes
Resource requests	Yes	Yes	Very limited	Yes
Limits on resources	No?	No	No	No
Flexible Scheduling	Depends on the underlying system	Depends on the underlying system	Yes	Yes
Job priorities	Depends on the underlying system	Depends on the underlying system	Yes	No
Job monitoring	Monitoring tools and logs	Monitoring tool and logs	No	Monitoring tools and logs
Accounting	No	Yes	Yes	No

Table 2: Resource management

Table 2 summarized the resource management aspects of the systems. All systems support batch jobs. Interactive jobs are under development for UNICORE. Parallel jobs are fully supported by TME, although the other systems also support the user in this area. All job management systems supports resource requests from users, only for AVAKI this is very limited. AVAKI and TME provides an own scheduling. Most of the systems provide support for job priorities assigned by users depending on the lower level system, except for TME. Globus, UNICORE and TME provide monitoring tools and detailed logs. Accounting is available in UNICORE and AVAKI.

	Globus	***UNICORE***	***AVAKI Compute Grid***	***TME***
Stage-in, stage-out	Yes	Yes	Yes	Yes
Timesharing	Jobs	Jobs	Jobs	Jobs
Load balancing	Yes (static)	No	No	Yes
Scalability	High	High	High	High for ITBL Limited for STA
Suspending, resuming, killing jobs	User only	User only	User only	User only
Fault tolerance	Limited	Site dependent	Limited	No

Table 3: Utilization

Table 3 composes the utilization of the systems. Stage-in and stage-out are supported by all tested systems, as well time-sharing for jobs. Load-balancing mechanisms are available for Globus and TME. The scalability of all analyzed systems can be characterized as high. All systems support the user in suspending, resuming and killing jobs. Fault tolerance is not really supported by the systems, only Globus and AVAKI provides some limited support for fault tolerance.

Summary and Conclusions

Introduction

As we have reiterated a number of times in this report, Grid middleware can be viewed as providing an abstract layer across heterogeneous resources and hence producing a uniform layer for access. This approach is common to all the systems we looked at, although there are substantial differences in how this is actually realized.

- UNICORE works on the principle of using abstraction built into the design of the infrastructure. Specifically, the abstraction is facilitated through the description of tasks in an abstract and target system independent format. The core middleware component is the Network Job Supervisor (NJS) which consumes the abstract description, distributes the component job tasks in the workflow to the various resources, and incarnates the tasks for specific target systems.
- Globus works a similar notion of Grid services provided. A user interfaces with these services directly through command-line tools, or through a programming environment or toolkits built upon the core Globus services.
- AVAKI provides a familiar environment for UNIX users. AVAKI commands often mirror those found in a UNIX shell, for example, `AVAKI ls`. As such, the AVAKI middleware provides something like console access to a 'virtual operating system for a global virtual computer'.
- TME focuses on the seamless integration of parallel programs. With the help of TME the user can design a workflow diagram of the distributed application. TME realizes a higher-level view of schematizing the structure of meta-applications.

Additionally, we can make a distinction between the four systems with regard to licensing arrangements. Globus is developed under the 'Globus Toolkit Public License (GTPL)'. This is 'a liberal open source license' allowing the software to be used by anyone and for any purpose, without restriction.

UNICORE is currently distributed under the UNICORE Forum License. This is modeled after the Sun Community Source License, and allows free access to the source code. UNICORE is also free for Research and Development purposes, i.e. for Grid projects using UNICORE. For production use a license from Pallas GmbH must be purchased.

AVAKI is a commercial product and can be purchased in different license variations. The Science + Computing AG is the German distributor of the AVAKI software.

TME and the underlying system STA is available under a special JAERI Open Source License [43] excluding the commercial use of the software.

Summary of Systems

Globus

Regardless of technical merits, Globus has a strong advantage over other Grid software due to its position as the *de facto* standard for Grid middleware and the plan by it's developers for it to become the first production-ready implementation of the Open Grid Services Architecture (OGSA). It difficult to completely verify the claims regarding it's dominance and it's proclaimed *de facto* status, i.e. we can not prove that this is the case with statistical facts. However, making a informal survey of Grid projects and deployments running currently, it would appear that a majority of them are based on Globus rather than any other system. This benefit should not be underestimated. In addition, the Globus community of users is very active (in the form of mailing lists, etc), and in our opinion there is more activity in this area than for the other three systems.. There are also a large number of projects which build upon Globus services to provide other higher-level Grid services, programming environments, etc.

The toolkit approach taken by Globus works well, and the software is mature. The Globus security infrastructure is implemented as a mandatory service which all the other services use. These other services are in the areas of : resource management, information services and data management. Globus offers powerful support for these core services. Infact, these core services *are* the extent of the Globus toolkit. Globus can be found at a relatively low-level in the software stack for a Grid, and one should bear in mind that the success of a Globus Grid is also dependent on the software installed on top of the Globus services. If Globus is deployed with other components, one can build a user-friendly Globus-based Grid environment. Alternatively, one can use Globus alone and manage and use the Grid using the Globus client tools. As such, we view Globus as a developer oriented system, as this is definitely the impression a users gets when using the tools directly.

The three core services use different protocols for access. For example, the usage of LDAP querying for the information services. While this enables powerful querying of both static and dynamic information, the syntax is unwieldy, and targeted at developers rather than regular users. This is in contrast to UNICORE. Here, although the information service is less comprehensive, the information provided by the IDB is viewable from the UNICORE client in a simple tabular form, and thus more easily accessible by end users.

Regarding Globus security a quote from [30] is interesting, "the philosophy of Globus is to enable sharing of computational resources across sites that have a relatively high level of trust in each other". The default security model provides rather wide privileges to remote users. Also, our experience was that Globus does not happily cooperate with firewalls. Often Globus deployments such as the Teragrid do not assume that firewalls exist between the various resources. Such factors tend to associate Globus with 'private' or dedicated Grids. Given such experiences, we feel that Globus is not particularly well suited to enabling desktop access to (super)computer resources, as one particular mode of operation important to users. This is the expertise of UNICORE, but a hybrid-solution perhaps offers the best possibilities.

UNICORE

UNICORE can be most easily categorised as facilitating Desktop access to Supercomputer resources, in as much that it greatly assists remote seamless access to super-computers from desktop machines. UNICORE can also be categorised as Distributed Supercomputing since a UNICORE workflow can describe the execution of multiple, interconnected jobs over many supercomputer resources. However, UNICORE does not provide the basis for a Data-intensive Grid, and does not address High Throughput computing. That said, it operates according to its initial specification very well, and it satisfies the majority of average user requirements. Indeed, our view of Globus as a developer oriented system, is in contrast to our view of UNICORE as a user oriented system.

The top-to-bottom architecture is both the strength and weakness of the system. For example, the UNICORE enables a strong security model, albeit one that is relatively inflexible. The integrated nature of the components, including the client, also makes for a trouble-free installation procedure for both administrator and user. The java implementation is helpful in this regard. Conversely, its vertical architecture and design of interfaces makes it relatively hard to extend or provide additional services for.

To allow a job to be assigned to a number of contributing machines, a user must sign a whole workflow before submitting it. This aspect of the design of UNICORE has allowed the use of a much tighter security model. However, one of the many definitions of Grid computing coined after UNICORE's conception is "constructing new capacities dynamically and transparently from distributed services". It is not clear if it is true to say that UNICORE conforms to this particular definition of a Grid. At the same time, it is not clear that this statement is the most important way to define a Grid.

The UNICORE client provides a firewall friendly access to the Grid. In terms of an e-commerce architecture classification, the UNICORE client is the top tier, the gateway and NJS the middle tier. With Globus it is typical that security administrators will require that a firewall is pre-configured with the IP addresses of machines able to reach Globus services. UNICORE allows a 'roaming' client and additionally the use of a installed application on the client machine allows some job construction off-line, before re-connection in order to submit.

In addition to the firewall-friendly nature of the UNICORE software, some argue that UNICORE is more secure than Globus. This argument is based mainly around the fact that UNICORE does not use proxy credentials, and the private key remains in the client. This is a relatively compelling argument. However, the bottom line is that, as with Globus, UNICORE allows remote users to run remote code, and as discussed in [30] the recommendation is to protect critical resources and be aware of potential security risks. Potentially, a UNICORE site could remove *script task* UNICORE access, and require that all software resources be defined in the Incarnation Database (IDB). This adds an extra layer of abstraction and thus a layer of security. This virtualisation of software resources is likely to be further enabled through the introduction of OGSA into the UNICORE software.

The UNICORE IDB does not provide dynamic information of the minute-to-minute state of the system. However, this kind of information can be obtained through a portal. Integration of dynamic information directly into UNICORE would be desirable, and would allow the development of a sophisticated resource broker, such as the one developed as part of the EuroGrid project.

UNICORE is competent at satisfying a significant proportional of common usage patterns, particular those happy to use resources in the 'traditional' manner (i.e. batch submission to a single resource). UNICORE additionally adds strong support from workflow and seamless access, including automatic executable and file staging. . UNICORE is strongest supporting a batch model for job submission. There is much less support for using UNICORE for parallel applications across resources and security domains. This usage is better enabled through the toolkit approach of Globus.

In practice, no one tool covers all aspects of user-requirements, but we feel UNICORE is appropriate for the majority of usage requirements. However, for UNICORE it is difficult to extend the services to implement any unsupported functionality. This is apparent from the lack of support from UNICORE in the Grid Programming Environments layer of the architecture model. Notably, the Globus model is particularly strong in this respect, and its architecture lends itself to such extensions.

AVAKI

AVAKI provides an environment to collect resources – processing power, data files, and applications – to be used as a single uniform operating environment. This set of shared resources is called an *AVAKI Grid*. An AVAKI Grid can represent resources from different platforms at a single site within a single department, or it can represent resources from multiple sites, heterogeneous platforms, and separate administrative domains.

AVAKI ensures secure access to files on the Grid. If files on participating machines are shared or made explicitly available for the grid they will become part of the Grid. Any subset of the resources existing in the environment can be shared. The administrator can decide which resources are visible to the Grid user. By the same token, a user of an individual computer or network that participates in the AVAKI grid is not automatically a grid user with access rights to grid files.

The approach to collecting and sharing resources is supported by a global naming scheme that creates a unique identity for each user and resource. This is also the foundation of the AVAKI idea of authentication. The software supports local firewalls by introducing a firewall proxy client that manages the communication through a firewall.

On the whole, however, AVAKI lacks many sorts of functionality found in Globus or UNICORE, and in addition seems challenging to use.

TME

TME (Task Mapping Editor) supports execution of parallel computing on supercomputer cluster. Users can describe contents about computing (program and data) and computer independently. Therefore TME supports combination of various computing by programs and data files on various parallel computers.

TME is one of the components of the STA or ITBL environments and provides a visual programming environment. TME features include the separation of processing detail descriptions and the specifications of computers that will perform the processing. By dynamically specifying the machines using TME, the mapping of routine processing can be easily changed according to the load condition of each computer at the start of processing.

Currently, basic functions of TME have been implemented, and the tool was used to create various applications of distributed parallel processing.

Conclusions from the Questionnaire

The majority of the respondents were familiar with the term “GRID”. Asking for associations with this concept most of them pointed to “supercomputing” and “flexible”. The first clearly indicates the nature of early Grid computing idea as well as the background of the users, while the latter addresses the increase in flexibility which is often attributed to future Grid solutions. Other high relevant associations were “cheap” and “coordinated”. On the other hand terms like “secure”, “reliable” and “consistent” seem to be less well associated with the term Grid. for the respondents. This probably results from the justifiable skepticism of experienced computer experts in the face of the vast quantities of marketing hype being produced at present.

It is interesting to note that only 24% of the respondents expect that the Grid will make the life easier for the users. However, nearly half of the respondents disagree with the statement that “Grid computing constrains the user more than it helps them”. This seems to indicate that users are not quite confident about what the future of Grid computing will bring with respect to ease of usage.

Asking the users about their impressions of the economical future of Grid computing one third of the respondents expect the Grid to be a commercial success. This is also reflected by the fact that another one third sees Grid as a bubble that will soon burst. Users are obviously not entirely confident about the future of the concept.

To get an understanding of the future requirements of the users we asked them about their type of applications and the size/type of the associated data traffic. As a result we can say that there will be a smaller number of applications that can be called Grid specific, but also that there are some applications that may benefit from high speed networks when running in a Grid environment. There are, however, a number of application scenarios for which network bandwidth is crucial even in the frame of traditional work flows. So it seems that quality of service for these applications is more important than a simple increase in speed.

The key issues in field of the user’s expectations on Grid software are the impact on their applications and the support for finding and using Grid resources. The users expect to get good support in choosing resources in the Grid environment, but for most of the respondents it is not so important to have a graphical user interface. With respect to ease of usage they seem to be pessimistic, maybe because a majority of them expect to have to change their applications for using in a Grid Computing environment.

Standardization and Interoperability

A lifecycle proceeds as follows: pregnancy, birth, childhood, adolescence, maturity, and decline. In terms of computer science, this description is often applicable, however one could argue that for some fundamental breakthroughs, such as the internet, the technology would appear so fundamental that it may live forever. Presently, Grid technology is perhaps going through its adolescent phase, and receiving lots of coverage, some might argue ‘hype’. This opinion is shared in [7], "... the impression is that the current phase corresponds to an early phase for Grid Computing characterized by a difficult interoperability between different Grid technologies, a complex installation and configuration process for Grid products, and finally a very limited number of end users”.

On the other hand, there is a strong argument for Grid technology and a "paradigm shift that will provide the next big boost in corporate productivity since the Internet and World Wide Web" [39]. In order for this to occur, there must a period of standardization, and once this is complete and the standards implemented in production systems, then this will signify the maturing of the Grid.

Therefore, there is a need to provide a uniform interface to the services offered by Grid middleware and standardized languages for the exchange of the messages themselves, thus motivating the development of suitable sets of native standards and protocols, which resources and their providers would adhere to in order to make them Grid enabled. The standardization work involved is currently being accomplished through organizations such as the GGF (and W3C, OASIS, for more omnipresent web and web services protocols).

Taking, for example Globus in its version 2 incarnation, it offers three core services of job submission, information and data transfer. With regard to the protocols used to communicate with the middleware, Globus takes a rather piecemeal approach, and as a result there is little consistency among products. For example, the GRAM protocol is http-based, the MDS protocol is based on LDAP. Recently, there has been a visible shift in the Globus project from API's and custom protocols, to standard protocols, and finally to web services.

The Open Grid Services Infrastructure (OGSI) provides a consistent interface for accessing any Grid resource. In simple terms OGSI is a framework for doing secure messaging between organized stateful services on a Grid. One such message initiates a job-submission, another could inspect the information associated with the node on the Grid, and another message could initiate a data transfer between two nodes on the Grid. This encourages thinking in terms of *messages* between *services*. It is an exciting development in the Grid Computing world as it promises to make the technology more easily accessible to a greater number of people. This is due in part to the adoption of web services standards, and the steering of the specification by the GGF. It is an excellent foundation, i.e. establishing the interface to all Grid resources, from which to base the future work of the other language standardization efforts necessary for a Grid to operate and interoperate.

Actually, application toolkits and graphical user interfaces should shield the user from the underlying protocols, and in this regard, the functionality of the software should be the determining factor in the software infrastructure selection. However, it is predicted that there will come a time when both open-source and proprietary Grid software will all implement to the same standards for interfaces and messages. Thus, this maturity and standardization of the software for building distributed systems supporting virtual organizations will result in direct benefit for all Grid users.

Of course, a lot of the benefits which the standardization efforts promise are completely dependent on the careful adoption of the specifications. History, here, however shows one instructive case. Novell networking, once extremely popular did not adhere to the relevant networking models. Novell's demise as a significant force in networking was at least partially the result of its failure to follow published, open standards. In time, it will be seen what will actually happen in practice. With regard to the current status of the software we are considering and their support for Grid standards, we will now examine the intentions, commitments and progress of the evaluated software.

The next major release of the Globus Toolkit is release 3. Globus Toolkit 3 (GT3) will implement the Open Grid Services Architecture. The alpha release of GT3 was available in January 2003, and the first production release is planned for July 2003.

Within the GRIP project, work is progressing with the integration of OGSA into the UNICORE architecture. Indeed, from early on, synergies between the UNICORE structure and OGSA have been recognized, and UNICORE has been active in the community to develop these standards. However, the GRIP project [18], for example, has recognized that although a common interface is very helpful, this is just the part of the solution. The Job description languages of Globus and UNICORE (RSL and AJO respectively) are different, and in order to interoperate, translation between these languages is necessary. Agreeing on common standards for such languages are major activities for the Globus Grid Forum.

As discussed in the specification [45], OGSI does not really represent a complete distributed object model, but it does have many characteristics of such a model. The view of a Grid as a collection of distributed objects has been the core philosophy of the original Legion project, although we note that in recent AVAKI documentation, there seems to be a initiative to rename some components to reflect the language of the Service Orientated Architecture, rather that that of a distributed object model. Indeed, AVAKI have also committed to supporting OGSA in their products, see [19].

JAERI has a plan to implement the communication infrastructure of the ITBL system, Starpc, by trusted communication libraries like globus-IO and grid-rpc. A detailed plan is confidential and cannot be addressed here.

Executive Summary

Our final recommendations are detailed below. The merits of all of the systems are such that we find it impossible to make a statement such as: "the best Grid software currently available is …" Indeed this is not intention of this project. Rather, the intent of the project is to evaluate the leading software systems in what is yet a very young area, and point out the strengths and weaknesses of various systems.

The Grid will bring more wide-spread and easier access to powerful computing resources, including computation, data storage and coordination amongst resources. This infrastructure is built on two things: middleware and networks. Grid middleware is the integration software which makes this sharing and collaboration possible. Although, there appears to be consensus on the core functionality, the software is rapidly evolving, and clearly a 'work in progress', and in this report we have examined some popular choices of Grid software and evaluated their effectiveness.

In reality, Grid computing is not really a new subject. Its roots are clearly in distributed systems, a subject which has been extensively researched over a number of years. Most of the requirements of the services, issues and potential bottlenecks would appear to have been recognized. Obviously, work is ongoing enhancing existing solutions, but consolidation seems to be the order of the day, with the main thrust establishing common architectural, protocol and language standards.

Equally important is the network infrastructure. Indeed, a good network is the foundation on which the Grid is built. We found from the questionnaire that most of the respondents felt that network performance (e.g. high bandwidth, low latency) are important issues for their applications, and running them on the Grid.

It is possible that in 5 years time once the standards for Grid software have passed through a number of iterations, and implementations has stabilized, a similar comparison between Grid middleware will be judged purely on higher-level services, security policy, user access and other such criteria.

In many respects Globus is very strong. Its architectural model presents building block services from which a wide-variety of high-level services, programming/user environments can be constructed. Globus also has the advantage of being the *de-facto* Grid middleware, and undoubted there are more Globus deployments worldwide than any other. As such there is a relatively big community surrounding the use of Globus, and projects which use or are based on Global technology. The wide ranging applicability of Globus is illustrated when we review the Grid projects and deployments based on Globus. We can summarize that Globus can be used as a basis for Grids in many of the areas discussed in the introduction, “Types of Grid”.

The disadvantage of the horizontal architecture of Globus is that it is not so easily deployed as an ‘out of the box’ Grid solution. However, this is addressed by initiatives such as the NMI [50] which package a number of components with which a Globus based Grid can be constructed.

We found UNICORE to be a very comfortable environment to work with. Additionally, it satisfies many of the user requirements and hence should be considered for most mainstream Grid deployments. In some respects it is not as capable as Globus. However, it is an integrated system which we feel makes it easier to deploy and use. The java-based software also helps in this respect. However, we felt that step by step instructions for installing UNICORE, particularly addressing how all the components are integrated together. UNICORE also installs with minimum disruption to the target systems; a small Perl daemon is all that must be installed. The implication of this is that it is normally easy to port the TSI to new target systems. This highlights some of the advantages of the UNICORE security infrastructure, particularly that the security is (partially) implemented as a function of a Usite (usually a local area network), and not on each individual resource. We have previously discussed the disadvantages of this approach, and it also assumes the Usite to be secured by other security measures.

Due to basic idea of the AVAKI Compute Grid, it seems to be a very good solution for building a Grid from distributed workstations at a local site to run single jobs. If it is used in this way the system provides an easy-to-use, seamless computational environment. However, the authors believe that it does not provide a solution to support the creation of a wide area Grid including supercomputer resources, mainly because of the missing support for some of the current main Supercomputer systems (e.g. NEC).

TME as part of a system like STA or ITBL can be described as an application manager. We think it’s a useful tool to establish a Meta-application on distributed computer resources, as parallel applications, special experiments and data files.

Another viewpoint, particularly with regard to UNICORE and Globus, is that these can be seen as complimentary solutions; UNICORE provides the workflow and interface functionality for the underlying collection of Globus resources. Looking to the future we look forward to closer interoperability between Grid systems, and where the ‘Grid operating system’ as such, will be based on defined standards.

Finally, we feel that research and development of Grid computing is currently being driven forwards with lots of investment and enthusiasm, and that the subject will remain compelling and exciting. Furthermore, this will result in the integration of more features that end users expect to find. In spite of the hype, there is real value to Grid systems and their potential to increase the efficiency and capability of computing systems.

Appendices

A Bibliography and References

1. I. Foster, C. Kesselman, S. Tuecke: ‘The Anatomy of the Grid: Enabling Scalable Virtual Organizations’ International J. Supercomputer Applications, 15(3), 2001.
2. http://www.AVAKI.com
3. G.Smecher: ‘Grid Topology using Condor and Globus’ available at http://grid.phys.uvic.ca/docs/pdf/globus-condor-intro.pdf
4. http://eu-grasp.net
5. http://www.deisa.org/Documents/Deisa_IP_eoi.pdf
6. ‘Computing power on tap’, Economist, Jun 21st 2001
7. JC Desplat, Judy Hardy, Mario Antonioletti ,Jarek Nabrzyski, Maciej Stroinski, Norbert Meyer: ‘Grid Service Requirements’, ENACTS Sectoral report , January 2002, http://www.enacts.org
8. Jan Fagerström, Torgny Faxèn, Peter Münger, Anders Ynnerman, J-C Desplat, Filippo De Angelis, Francesco Mercuri, Marzio Rosi, Antonio Sgamellotti, Francesco Tarantelli and Giuseppe Vitillaro: ‘High Performance Computing Development for the Next Decade, and its Implications for Molecular Modelling Applications’, Sectoral report ENACTS project, April 25, 2002, http://www.enacts.org
9. http://www.eurogrid.org/
10. D.de Roure, N.Jennings, M.Baker, N.Shadbolt: ‘The evolution of the Grid’, International Journal of Concurrency and Computation: Practice and Experience 2003
11. http://ganglia.sourceforge.net/
12. http://www.computingportals.org
13. Gregor von Laszewski, Gail Pieper, and Patrick Wagstrom. *Performance Evaluation and Characterization of Parallel and Distributed Computing Tools*, chapter ‘Gestalt of the Grid’. Wiley Book Series on Parallel and Distributed Computing. to be published 2002
14. http://www.ggf.org
15. http://www.globus.org
16. http://www.gridlab.org
17. F.Berman, G.Fox, T.Hey: ‘The Grid: Past, Present and Future’, in *Grid Computing: Making the Global Infrastructure a Reality*, F. Berman, G. Fox and T. Hey (eds.), Wiley, 2003.
18. http://www.grid-interoperability.org/
19. http://www.AVAKI.com/news/releases/20030113_13.html
20. ‘Introduction to Grid Computing and the Globus Toolkit’, http://www.globus.org/training/grids-and-globus-toolkit/IntroToGridsAndGlobusToolkit.pdf
21. G, von Laszewski, I. Foster, J. Gawor, P. Lane: ‘A Java Commodity Grid Toolkit’, Concurrency: Practice and Experience, 13, 2001.
22. G. von Laszewski, et al: ‘The Java CoG Kit User Manual – draft version 1.1a’, 2003. http://www.globus.org/cog/manual-user.pdf
23. C. Lee, D. Talia: ‘Grid Programming Models: Current Tools, Issues and Directions’, in *Grid Computing: Making the Global Infrastructure a Reality*, F. Berman, G. Fox and T. Hey (eds.), Wiley, chapt. 21, pp. 555-578, 2003.
24. M.Lorch: ‘Symphony - A Java-based Composition and Manipulation Framework for Computational Grids’, http://zuni.cs.vt.edu/publications/symphony-thesis-lorch.pdf

25. N. Karonis, B. Toonen, and I. Foster: 'MPICH-G2: A Grid-Enabled Implementation of the Message Passing Interface', Journal of Parallel and Distributed Computing to appear 2003.
26. I.Foster: 'Internet Computing and the Emerging Grid', http://www.nature.com/nature/webmatters/Grid/Grid.html
27. I. Foster, C. Kesselman, J. Nick, S. Tuecke: 'The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration', Open Grid Service Infrastructure WG, Global Grid Forum, June 22, 2002.
28. C.Lee et al: 'A Grid Programming Primer', http://www.eece.unm.edu/~apm/docs/APM_Primer_0801.pdf
29. R. Buyya et al: 'Problem solving environment comparison', white paper http://www.eece.unm.edu/ apm, 2001.
30. M.Surridge: 'A Rough Guide to Grid Security', e-Science Technical Report Series (UKeS-2002-05)
31. V.Sander: 'Design and Evaluation of a Bandwidth Broker that Provides Network Quality of Service for Grid Applications', NIC Series Volume 16, 2003
32. http://www.unicore.org
33. D.Erwin: 'UNICORE – A Grid Computing Environment',. Grid Computing Environments 2001 Special Issue of Concurrency and Computation: Practice and Experience.
34. http://www.fz-juelich.de/zam/RD/coop/unicoreplus/
35. http://www.w3c.org
36. http://setiathome.ssl.berkeley.edu/
37. http://www.entropia.com/
38. M. Chetty and R. Buyya: 'Weaving Computational Grids: How analogous are they with electrical grids?' IEEE Computing in Science & Engineering, vol. 4, no. 4, pp. 61-71, July-Aug. 2002
39. http://techupdate.zdnet.com/techupdate/stories/main/0,14179,2865519,00.html
40. S. Gallopoulos, E. Houstis and J. Rice: 'Computer as Thinker/Doer: Problem-Solving Environments for Computational Science', IEEE Computational Science and Engineering, Summer 1994.
41. http://www.cs.wisc.edu/condor/
42. K. Czajkowski, I. Foster, C. Kesselman, V. Sander, S. Tuecke: 'SNAP: A Protocol for negotiating service level agreements and coordinating resource management in distributed systems'. *Lecture Notes in Computer Science*, 2537:153-183, 2002.
43. http://ccse.koma.jaeri.go.jp/program/eng/software/attention.html
44. http://doesciencegrid.org//projects/GPDK/
45. http://www.gridforum.org/ogsi-wg/
46. T. Imamura, Yukihiro Hasegawa, Hironobu Yamagishi and Hiroshi Takemiya: 'TME - a Distributed Resource Handling Tool', International Conference on Scientific and Engineering Computation (IC-SEC2002), Singapore, December 3rd-5th, 2002.
47. Yuishi Tsujita, Toshiyuki Imamura, Hiroshi Takemiya and Hironobu Yamagishi: 'Stampi-I/O: A Flexible Prallel-I/O Library for Heterogeneous Computing Environment', 9th EuroPVM/MPI conference, Linz, Austria, September 29th-October 2nd, 2002.
48. http://ccse.koma.jaeri.go.jp/publicity_open_software/starpc.html
49. Avaki coporation: AVAKI Grid Software: Concepts and Architecture (white paper), March 2002
50. http://www.nsf-middleware.org/
51. http://www.globus.org/gt2.4/download.html
52. http://www.globus.org/gt2.4/install.html
53. http://www.globus.org/security/simple-ca.html
54. http://www.globus.org/security/v2.0/Globus%20Firewall%20Requirements-0.3.pdf

55. Thomas Beisel, Edgar Gabriel, Michael Resch: 'An Extension to MPI for Distributed Computing on MPPs' in Marian Bubak, Jack Dongarra, Jerzy Wasniewski (Eds.) 'Recent Advances in Parallel Virtual Machine and Message Passing Interface', Lecture Notes in Computer Science, Springer, 1997, 75-83.
56. I. Foster, C. Kesselman: Globus: 'A Metacomputing Infrastructure Toolkit', Intl J. Supercomputer Applications, 11(2):115-128, 1997.
57. ‘Joint Project Report for the BMBF – Project UNICORE Plus’, Editor D.Erwin, ISBN 3-00-011592-7.

B Terminology

AJO	Abstract Job Object
ASP	Application Service Provider
CA	Certification Authority
CoG kit	Commodity Grid kit
DUROC	Dynamically-Updated Request Online Coallocator
EM	Execution Manager
GASS	Globus Access to Secondary Storage
GIIS	Grid Index Information Service
GGF	Global Grid Forum
GPDK	Grid Portal Development Kit
GPT	Globus Packaging Technology
GRAM	Globus Resource Allocation Manager
GRIS	Grid Resource Information Service
GSI	Grid Security Infrastructure
GT	Globus Toolkit
GUI	Graphical User Interface
IDB	Incarnation DataBase
ITBL	Information Technology Based Laboratory
LAN	Local Area Network
LDAP	Lightweight Directory Access Protocol
LSF	Load Sharing Facility
MDS	Monitoring and Directory Service
MPI	Message Passing Interface
NJS	Network Job Supervisor
OGSA	Open Grid Standard Architecture
OGSI	Open Grid Services Infrastructure
PBS	Portable Patch System
PKI	Public Key Infrastructure
PSE	Problem Solving Environment
RIS	Resource Information Monitor
RIM	Resource Information System
RPC	Remote Procedure Call
RSL	Resource Specification Language
SSO	Single-Sign-on
STA	Seamless Thinking Aid
TSI	Target System Interface
Usite	UNICORE site
UUDB	UNICORE user data base
VO	Virtual Organization
VPN	Virtual Private Network
Vsite	Virtual site

C Questionnaire

1. ***Are you aware of the term 'Grid'?***

57	Yes
6	No

2. ***Name 5 terms that you associate with the term Grid?***

13	As simple to use as a PC
26	Coordinated
15	Widely Spread
3	Consistent
8	Reliable
33	Flexible
8	Secure
26	Cheap
34	Usage of a super computer
10	Others: hip, dynamic, dynamic job distribution to resources, dynamic resource availability, usage of computing power where available, anonymous, high computing power directly available, SETI@home, a grid comprises of a network of resources, security risks, barriers, massively parallel, circuitous

3. ***Are you aware of the term PSE?***

20	Yes
43	No

4. ***If so, does the Grid in your opinion represent a PSE?***

13	Yes
11	No

5. ***What role will Grid computing in your opinion play in the future?***

10	Very important
27	Important
24	Not so important
2	Unimportant

Where do you stand on the following statements?

6. ***In the future Grid computing will provide computing power just like today's electric Grid.***

5	I totally agree
9	I rather agree
26	Partly
13	I agree less
10	I don't agree at all

7. ***Grid computing is a bubble that will soon burst.***

6	I totally agree
13	I rather agree
21	Partly
20	I agree less
3	I don't agree at all

8. ***Grid computing will replace super computing in general.***

3	I totally agree
2	I rather agree
14	Partly
22	I agree less
22	I don't agree at all

9. ***Computer centres will lose their importance due to Grid computing.***

3	I totally agree
5	I rather agree
9	Partly
18	I agree less
28	I don't agree at all

10. ***Grid computing will make life easier for the user.***

4	I totally agree
12	I rather agree
19	Partly
20	I agree less
8	I don't agree at all

11. ***Grid computing is limited to problems which are simple to parallelize, such as high energy physics.***

7	I totally agree
23	I rather agree
16	Partly
11	I agree less
6	I don’t agree at all

12. ***Grid computing will be a commercial success because above all it is interesting to the economy.***

5	I totally agree
16	I rather agree
11	Partly
25	I agree less
6	I don’t agree at all

13. ***Grid computing will be a standard in the future which will be driven by large software companies.***

3	I totally agree
10	I rather agree
21	Partly
24	I agree less
5	I don’t agree at all

14. ***Application Service Providers (ASP) are the future of super computing.***

4	I totally agree
7	I rather agree
24	Partly
18	I agree less
10	I don’t agree at all

15. ***Computer centres will be re-valued through Grid computing because computers are used more centrally again.***

5	I totally agree
11	I rather agree
22	Partly
21	I agree less
4	I don’t agree at all

16. Grid computing constrains the user more than it helps him.

3	I totally agree
17	I rather agree
10	Partly
28	I agree less
5	I don't agree at all

17. ***In the future computing power will mainly be provided by local computers.***

12	I totally agree
14	I rather agree
20	Partly
11	I agree less
6	I don't agree at all

Make an appraisal of your requirements

18. What type of application do you mainly use?

34	Pure batch mode
23	Workflow/piped applications (one run after the other)
13	(Online) controlled applications
12	Observed applications (online visualization of application)
4	Steered applications (online visualization of applications with feedback)
3	Coupled applications (applications such as independent fluid structure)
20	Parallel applications with low communication
38	Parallel applications with lots of communication
5	Controlled distributed applications
0	Observed distributed applications
1	Steered distributed applications
2	Others: vectorized application

19. Requirement type?

18	Only transfer of input/output data to computer
12	Seldom and low communication between the computers, as in online control
14	Seldom but intensive communication between computers as in online visualization
16	Frequent but small communication between computers
27	Frequent and intensive communication between computers

20. How dependent are your typical programs on network latency?

20	Strong
16	Less strong
20	Normal
7	Less
0	Not at all

21. In which area should the latencies lie?

19	Up to 10 microseconds
17	Between 10 and 100 microseconds
10	Between 100 and 1000 microseconds
4	Between 1 and 10 milliseconds
1	Between 10 and 100 milliseconds

22. How strongly dependent are the typical programs on bandwidth?

27	Strong
14	Less strong
15	Normal
7	Less
0	Not at all

23. In which range should that bandwidths lie?

3	Up to 1 Mbit/sec
4	Between 1 Mbit/s and 10 Mbit/s
11	Between 10 Mbit/s and 100 Mbit/s
22	Between 100 Mbit/s and 1 Gbit/s
18	Over 1 Gbit/s

24. In which area does the size of the input data lie?

20	Up to 1 MB
12	Between 1 MB and 10 MB
8	Between 10 MB and 100 MB
15	Between 100 MB and 1 GB
10	Over 1 GB

25. In which area does the size of the output data lie?

4	Up to 1 MB
4	Between 1 MB and 10 MB
12	Between 10 MB and 100 MB
16	Between 100 MB and 1 GB
28	Over 1 GB

26. Do your applications use parallel I/O?

11	Yes
22	To some extent
30	No

27. Is a post-processing of the generated output data necessary?

38	Always
20	Sometimes
5	Never

28. Do you use debugging tools on high performance systems for your application?

5	Always
39	Sometimes
19	Never

29. Do you use performance profiling and analysis tools on the High Performance Systems for your application?

11	Always
42	Sometimes
10	Never

When using a GRID you expect…..

30. ... your application will be easier to handle.

9	Totally agree
9	Rather agree
9	Partly
28	Rather do not agree
8	Definitely disagree

31. ... that your application can be used completely without changes.

9	Totally agree
11	Rather agree
10	Partly
14	Rather do not agree
19	Definitely disagree

32. ... that you need to write a Batch-job script in order to use your application in this environment.

24	Totally agree
21	Rather agree
10	Partly
5	Rather do not agree
3	Definitely disagree

33. ... that you need to adapt a program by using a particular GRID tool , in order to use it.

11	Totally agree
28	Rather agree
13	Partly
7	Rather do not agree
4	Definitely disagree

34. ... that you need to rewrite your application by using a specific GRID-API, in order to use it.

9	Totally agree
12	Rather agree
9	Partly
19	Rather do not agree
14	Definitely disagree

35. ... that you have to login per session ...

54	once
3	on every computer that you wish to use in the grid
3	each institution that is involved in the grid
3	on every computer at each institution that is involved in the grid

36. ... as an interface ...

11	a graphical user interface
23	a console based interface
26	both
3	none of the above

37. ... to know where the applications that you want to use can be found.

11	Totally agree
19	Rather agree
8	Partly
18	Rather do not agree
7	Definitely disagree

38. ... to have to know where your input and output data is found.

18	Totally agree
15	Rather agree
13	Partly
13	Rather do not agree
4	Definitely disagree

39. ... that you can explicitly choose the resources (hardware and software), which you wish to use.

26	Totally agree
21	Rather agree
8	Partly
8	Rather do not agree
0	Definitely disagree

40. ... to have to select these again each time you use the GRID resources

5	Totally agree
8	Rather agree
15	Partly
25	Rather do not agree
10	Definitely disagree

41. ... that the GRID automatically select the best resource for you

10	Totally agree
21	Rather agree
16	Partly
12	Rather do not agree
4	Definitely disagree

42. ... that the GRID offers both options (automatic and user controlled search for resources)

34	Totally agree
22	Rather agree
7	Partly
0	Rather do not agree
0	Definitely disagree

43. ... that in the usage of different types of resources, your application has to be re-compiled for each.

13	Totally agree
19	Rather agree
9	Partly
12	Rather do not agree
10	Definitely disagree

44. ... that each application compiled by you must be transported to each resource.

3	Totally agree
5	Rather agree
10	Partly
17	Rather do not agree
28	Definitely disagree

45. ... a debugging tool, which you can use in the GRID environment.

30	Totally agree
19	Rather agree
8	Partly
4	Rather do not agree
2	Definitely disagree

46. ... Performance Profiling and Analysis Tools for the tuning of your application in this environment.

32	Totally agree
19	Rather agree
5	Partly
5	Rather do not agree
2	Definitely disagree

47. ... that the above mentioned tools can be used in the same way as in non-GRID environments.

15	Totally agree
12	Rather agree
16	Partly
15	Rather do not agree
5	Definitely disagree