Grid Service Requirements

A joint scientific and technological study undertaken by EPCC and the Poznan Supercomputing Centre for the ENACTS network

Sectoral report

JC Desplat, Judy Hardy, Mario Antonioletti (EPCC)
Jarek Nabrzyski, Maciej Stroinski, Norbert Meyer (PSNC)

January 2002

Table of Contents

List of figures

Part I The ENACTS project

I.1 Remits

ENACTS is a Co-operation Network in the 'Improving Human Potential Access to Research Infrastructures' Programme.

This Infrastructure Co-operation Network brings together High Performance Computing (HPC) Large Scale Facilities (LSF) funded by the DGXII's IHP programme and key user groups. The aim is to evaluate future trends in the way that computational science will be performed and the pan-European implications. As part of the Network's remit, it will run a Round Table to monitor and advise the operation of the four IHP LSFs in this area, EPCC (UK), CESCA-CEPBA (Spain), CINECA (Italy), and BCPL-Parallab (Norway).

This co-operation network follows on from the successful Framework IV Concerted Action (DIRECT: ERBFMECT970094) [1] and brings together many of the key players from around Europe who offer a rich diversity of High Performance Computing (HPC) systems and services. In ENACTS, our strategy involves close co-operation at a pan-European level – to review service provision and distil best-practice, to monitor users' changing requirements for value-added services, and to track technological advances. In HPC the key developments are in the area of Grid computing and are driven by large US programmes. In Europe we urgently need to evaluate the status and likely impacts of these technologies in order to move us towards our goal of European Grid computing, a 'virtual infrastructure' - where each researcher, regardless of nationality or geographical location, has access to the best resources and can conduct collaborative research with top quality scientific and technological support.

ENACTS provides participants with a co-operative structure within which to review the impact of Grid computing technologies, enabling them to formulate a strategy for increasing the quantity and quality of access provided.

I.2 Scope and Membership

The scope of our network is computational science: the HPC infrastructures which enable it and the researchers, primarily in the physical sciences, which use it.

Centre	*Role*	*Skills/Interests*
EPCC	IHP-LSF	Particle physics, materials science
ICCC Ltd	User	Optimisation techniques, control engineering
UNI-C	LSF	Statistical computing, bioinformatics, multimedia
CSC	User	Meteorology, chemistry
ENS-L	Society	Computational condensed matter physics, chemistry
FORTH	User	Computer science, computational physics, chemistry
TCD	User	Particle physics, pharmaceuticals
CINECA	IHP-LSF	Meteorology, VR
CSCISM	User	Molecular sciences
UiB	IHP-LSF	Computational physics
PSNC	User	Computer science, networking
UPC	IHP-LSF	Meteorology, computer science
NSC	User	Meteorology, CFD, engineering
ETH-Zurich	LSF	Computer science, physics

Table 1: ENACTS participants by role and skills

Three of the participants (EPCC, CINECA and CESCA-CEPBA) are LSFs providing Researchers' Access in HPC under the HCM and TMR programmes. All were successful in bidding to Framework Programme V (FP V) for IHP funding to continue their programmes. In this, they have been joined by the Parallab and the associated Bergen Computational Physics Laboratory (BCPL) and all four LSFs are full partners in this network proposal and plan to co-operative more closely in the Transnational Access programme.

Between them, these LSFs have already provided access to over 500 European researchers in a very wide range of disciplines and are thus well placed to understand the needs of academic and industrial researchers. The other 10 ENACTS members are drawn from a range of European organis ations with the aim of including representation from interested user groups and also by centres in economically less favoured regions. Their input will ensure that the Network's strategy is guided by users' needs and relevant to smaller start-up centres and to larger more established facilities.

A list of participants together with their role and skills is given in Table 1, page 9, whilst their geographical distribution is illustrated in Figure 1.

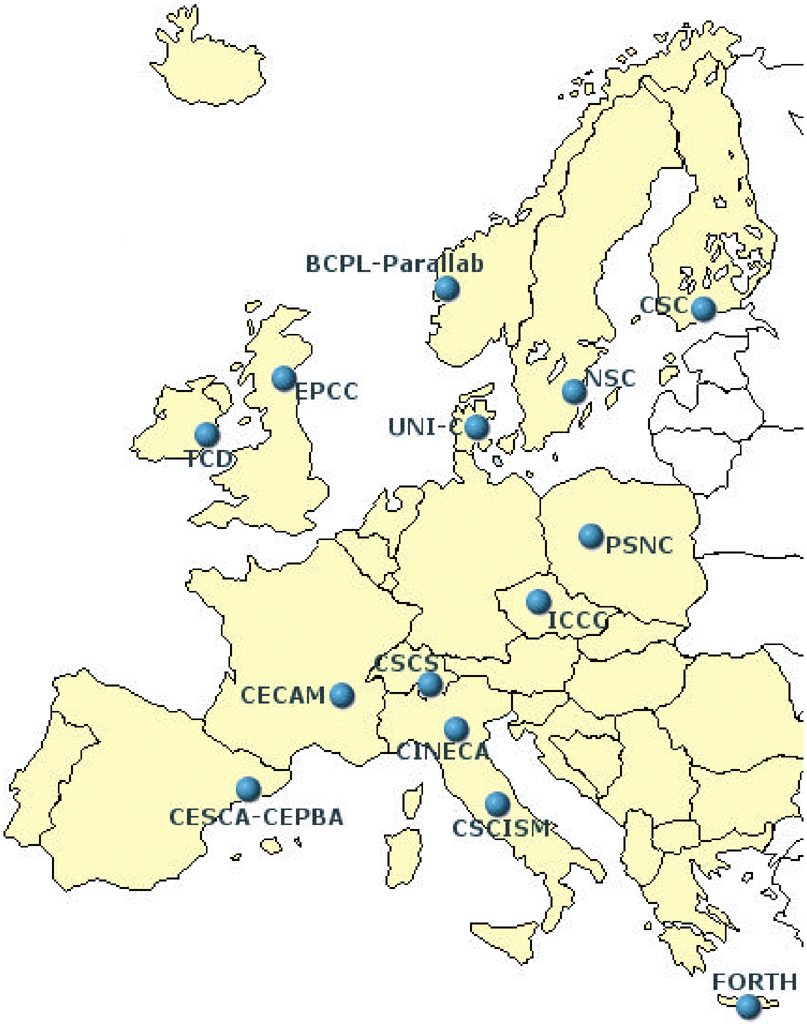


Figure 1: the ENACTS co-operation network

I.3 Workplan

The principal objective is to enable the formation of a pan-European HPC metacentre. Achieving this goal will require both capital investment and a careful study of the software and support implications for users and HPC centres. The latter is the core objective of this study. Bids for the former (e.g. RTD proposals) may also be deliverables.

The project is organised in two phases. A set of six studies of key enabling technologies will be undertaken during the first phase:

- Grid service requirements (EPCC, PSNC);
- the roadmap for HPC (NSC, CSCISM);
- Grid enabling technologies (ETH-Zurich, Forth);
- data management and assimilation (CINECA, TCD);
- distance learning and support (UNI-C, ICCC Ltd.); and
- software portability (UPC, UiB).

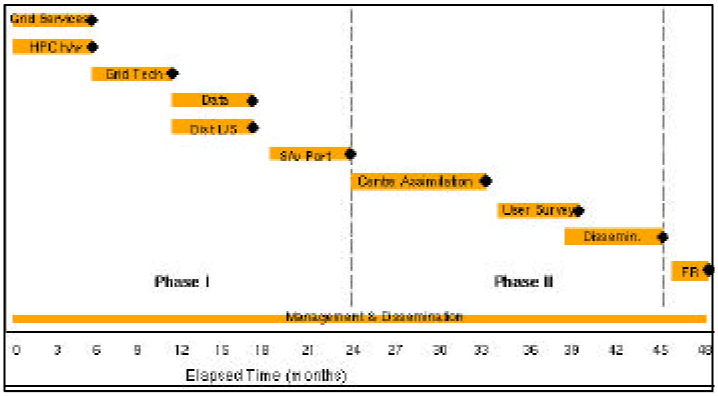


Figure 2: Gantt Chart for the Work of the ENACTS Network

Following on from the results of the Phase I projects, the Network will undertake a demonstrator of the usefulness and problems of Grid computing and its Europe-wide implementation. Prior to the practical test, the Network will undertake a user needs survey and assessment of how centres' current operating procedures would have to change with the advent of Grid Computing. As before, these studies will involve only a small number of participants, but with the results disseminated to all:

- European metacentre demonstrator: to determine what changes are required to be implemented by HPC centres to align them with a Grid-centric computing environment. This will include a demonstrator project to determine the practicality of running production codes in a computational Grid and migrating the data appropriately (EPCC, UiB, TCD);
- user survey : to assess needs and expectations of user groups across Europe (CINECA and CSC);
- dissemination: initiation of a general dissemination activity targeted on creating awareness of the results of the sectoral reports and their implications amongst user groups and HPC centres (UPC, UNI-C, ENS-L). The Network Co-ordinator and the participants will continue their own dissemination activities during Year 4.

I.4 Joint Scientific/Technological Activities and Studies – Grid Service Requirements

I.4.1 Objective

Computational Grids can be complex and difficult to use, but their usability can be enhanced by the provision of simple transparent user services. The objective of this study is to specify the level and quality of services users require from a computational Grid. The report will be written in consultation with a representative selection of users from different computational science groups across Europe.

This top-down approach will be complemented by a bottom-up examination of the enabling technologies.

EPCC and PSNC will undertake this activity. There are five workpackages in this project, totalling 6 staff months of effort. The elapsed time for the project is 6 months.

I.4.2 The partners

I.4.2.1 EPCC, UK

Edinburgh Parallel Computing Centre (EPCC) was established as a focus for the University of Edinburgh's work in high performance computing. The Centre's task is to accelerate the effective exploitation of high performance parallel computing systems throughout academia, industry and commerce. It houses an exceptional range of computers. EPCC's goal is achieved through a range of activities spanning undergraduate and advanced training programmes, service provision, industrial affiliation and contract work. EPCC offers software design and development for distributed and high-performance systems and offers training courses and materials covering all aspects of HPC.

Alongside their on-going technology transfer work with business, EPCC is also heavily involved in developing software for the Computational Grid. EPCC is a core part of the UK's e-Science programme, which enables Internet-based global collaborations using very large data collections, tera-scale computing resources and high performance visualisation. EPCC's particular interests lie in building on Grid software standards with platform-neutral distributed technologies like XML, Jini and Java in a high performance context to provide middleware solutions with potential commercial application. EPCC is also working in close collaboration with the UK National E-Science Centre (NeSC) to promote Grid technology within the UK scientific community [2].

See http://www.epcc.ed.ac.uk for further information.

I.4.2.2 PSNC, Poland

Poznan Supercomputing and Networking Centre (PSNC) was established in 1993 by the State Committee for Scientific Research. From the very beginning PSNC has been operating as the HPC service provider and the operator of Poznan Metropolitan Area Network (POZMAN). As the most active centre in Poland PSNC soon became an operator of the national research network POL34/155. Today it is also the operator of the Polish international link to the GÉANT network [3] and is the national networking excellence centre.

Each of the four PSNC's departments has an active computer science research group working on aspects such as: middleware, tools and methods for Grid computing, resource management for Grids, large scale Grid applications (Application Department), user accounting on the Grid, Grid security mechanisms and policies, Grid fabric management tools (Supercomputing Department), application portals, Grid portals, multimedia services, mobile user support technologies and services, digital libraries (Network Services Department), tools for network management, optical networks, QoS management (Networking Department). All the departments co-operate strongly with other institutes across Poland, Europe and the USA on Grid-related projects.

PSNC is the main designer of the national IST and GRID-related program, called "PIONIER - Polish Optical Internet, Applications, Technologies and Tools", for the Polish national information infrastructure, which has been accepted by the Polish Parliament for the years of 2001-2005.

PSNC was involved in building the European Grid Forum (EGrid). The first EGrid workshop was held in Poznan, in April 2000. The EGrid rapidly evolved into a large forum, with several working groups and involving over 300 individuals from Europe. Lately it has merged with the Grid Forum to form together the Global Grid Forum [4].

Currently PSNC is the coordinator of the GridLab project [5] and participates in the CrossGrid project [6].

See http://www.man.poznan.pl/english for further information.

I.4.3 The authors

I.4.3.1 Dr. Mario Antonioletti

Mario is an applications consultant at EPCC, the University of Edinburgh, UK. He has worked there for seven years and has many varied duties during that time. He started off in the Training and Education Centre and was heavily involved teaching HPC and putting the course material up on the web. He has worked in various other areas most recently: developing a question authoring CGI system that interfaced with WebCT, the production of on-line training materials, developing an E-mail parallel job submission harness for an industrial company which is to act as the basis for a future Grid project [103]. Mario wrote the web version of the questionnaire and did the initial processing of the data.

I.4.3.2 Dr. Jean-Christophe Desplat

Jean-Christophe is a Principal Consultant at the Edinburgh Parallel Computing Centre (EPCC), UK. His main duties include management responsibilities for the EC-funded 'Training and Research on Advanced Computing Systems' (TRACS) [7] and 'European Network for Advanced Computing Technology for Science' (ENACTS) programmes, as well as code development and resource management for Prof. Michael E. Cates' consortium 'High-Performance Computing for Complex Fluids' [8]. This particular collaboration resulted in eight joint publications in major scientific publications over the past three years.

Since he has started his involvement with the TRACS programme in 1995, Jean-Christophe has personally provided support to over 67 European researchers from all areas of computational science and engineering. This allowed him to gain a considerable expertise on all aspects of HPC and scientific numerical algorithms, but also to develop a sharp understanding of the needs and diversity of European HPC users and their applications, as well as the importance of added services (training, support, visualisation) and the best means for their delivery.

Jean-Christophe is currently involved in the UK EPSRC E-Science pilot project 'the Reality Grid' [9].

Contact: j-c.desplat@epcc.ed.ac.uk.

I.4.3.3 Dr. Judy Hardy

Judy works for EPCC as an Applications Consultant in the Academic Research, Training and Support group. Her work includes the development of web-based training materials for the SocratesAINN project [10]. This is an EC-funded collaborative programme developing innovative tools for on-line distance learning. She is also the course co-ordinator for EPCC's recently-established MSc in High Performance Computing [11].

I.4.3.4 Dr. Norbert Meyer

Norbert received the MSc degree in Computer Science from the Poznan University of Technology (1993) and PhD degree from the same university. Currently he is the head of the Supercomputing Department at Poznan Supercomputing and Networking Centre. He participated in several national projects concerning HPC technology, leader of the concept of connecting Polish supercomputer centres into Polish national cluster. His research interests concern resource management in GRID environment, GRID accounting (Global Grid Forum), network security (mainly in the aspects of connecting independent, geographically aware Grid domains), data management, technology of development graphical user interfaces. He is also an author and co-author of several reports and over 20 papers in conference proceedings and professional journals.

I.4.3.5 Dr. Jarek Nabrzyski

Jarek received his MSc and PhD degrees in computer science from Poznan University of Technology. Currently he occupies a research position at the Poznan Supercomputing and Networking Centre, where he heads the Applications Department. His research interests over the last 10 years have focused on knowledge-based multiobjective project scheduling, and resource management for parallel and distributed computing. For the last five years he has been working on tools and middleware technologies for Computational Grids. He currently leads several Grid-related projects at PSNC: MCRM Portal - multicriteria resource management architecture for the Grids with Portal Access, GridPred - job runtime and queue wait time prediction system.

Jarek is a co-founder of the European Grid Forum and the Global Grid Forum. He is a member of the Global Grid Forum Steering Group and he actively promotes Grid ideas and concepts in Europe. He was the Program Committee Chair of the First Global Grid Forum meeting held in Amsterdam in March 2001. Currently he is the co-coordinator of the EU founded project, called GridLab (IST-2001-32133) [5].

Contact: naber@man.poznan.pl.

I.4.3.6 Dr. Maciej Stroinski

Maciej received a PhD degree in Computer Science from the Technical University of Gdansk in 1987. Currently he is the Technical Director of the Poznan Supercomputing and Networking Centre. He is also a lecturer in the Institute of Computing Science at the Poznan University of Technology. His research interests concern computer network protocols and management. He is the author or co-author of over 100 papers in major professional journals and conference proceedings.

I.4.4 Rationale

With a few notable exceptions (e.g. see [12]) the actual beneficiaries and users of tomorrow's Grid technology have not yet established a dialogue with the people involved in the standardisation and development of the Grid middleware and the implementation of higher-level services. Their requirements are thus often at best assumed or second guessed, or simply ignored in the worst case. Clearly, a rapid acceptance and following uptake of such a technology in the user community will be key to its success. Indeed, Grid computing does not escape the phenomenon so well described by Metcalfe's law [13]:

> *"The usefulness, or utility, of a network equals the square of the number of users."*

Just as for the Internet, the development of the Grid infrastructure is anticipated to become naturally exponential once a critical mass has embraced it.

This first study is composed of four main parts:

- Part I: Presentation of the ENACTS project, its scope and membership (this section).
- Part II: Presentation of the different types of computational Grids, with a discussion of their features, limitations, and FAQ for prospective users. Since the material presented in this section is relatively detailed and tackles technical considerations which are mostly irrelevant to end-users, the authors expect this section to be of particular interest to operators of HPC systems and systems administrators who want to find out more about Grid computing. Sections specifically aimed at end-users (such as a FAQ) have also been included in this second part.

  This section has been written by PSNC.
- Part III: Results from a comprehensive survey undertaken among 85 major European research groups. This covered aspects such as awareness and expectations of Grid computing, groups and applications profile, working practices, bottlenecks and requirements, future needs, etc.

  This section has been written by EPCC.
- Part IV: Conclusions and recommendations. The purpose of the survey presented in Part III is to gain a better understanding of these key user groups' composition and working practices in view of

establishing their key requirements. The analysis of this survey, together with Part II, which presented technical and implementation issues, will be instrumental in establishing key recommendations for policy makers and resource providers. This section will provide a summary of the features offered by the leading-edge Grid middleware presented in Part II, compare this to the users' requirements established throughout Part III, and following these, make recommendations on a possible strategy for a successful uptake of Grid technology within the user community.

This section has been written by EPCC.

The authors would like to acknowledge the support of the European Commission. The ENACTS project is funded under the 'Improving Human Potential, Access to Research Infrastructures' programme. Project reference: HPRI-2000-40015.

See http://www.enacts.org/ for up-to-date information on the ENACTS project.

Part II Grid Models

II.1 Introduction

Part II of the report describes those Grid implementations which are already in use or can realistically be implemented within the next couple of years. The rationale is to propose practical solutions which are the most likely to fulfil the users' requirements. These Grid implementations include:

a) *Basic portal computing (workbenches):* suitable for groups using a set of 'standard' (i.e. stable in time) executables. Examples of this approach are already in use (e.g. CFDNet) or are currently being developed (e.g. EPCC's MHD portal). We will focus on the workbenches, problem solving environments and supporting toolkits.

b) *Advanced portal computing:* this is an evolution of the previous model with support for remote compilation and user options (e.g., users are given control of which features will be built into the binary, using which code revision, etc.). This model is particularly relevant to groups with codes requiring compile-time customisation (e.g. 'pick-and-mix package' such as DL_POLY, custom codes with user-selectable options, etc.)

c) *The toolkit approach:* Globus, Legion and Unicore.

d) *Small-scale Grids:* Condor, LSF.

e) *Peer-to-peer computing and capacity computing (Entropia, Seti@Home).*

All the Grid models mentioned above will be described in four main sections. The following topics will constitute the technical content of each section:

- Locating and characterising remote resources in the computational or information Grid in a uniform, location-independent fashion, dependent on the requirements of the application:
  - application description: will describe the methods/mechanisms that allow a user to describe the application;
  - resource description: will describe the mechanisms for resource description;
  - contract specification methods: will describe how a user can specify the requirements regarding dynamic resource usage, contract violation, etc;
  - Grid information services;
  - brokering services;
  - advanced reservation;
  - scheduling mechanisms;
  - co-scheduling (running multiple jobs on multiple machines);
  - application support (APIs).

II.2 Taxonomy of the Grid

The way we use computers today is dramatically changing, and those changes are caused mainly by the popularity of the Internet and the availability of powerful computers and high-speed networks. These new technologies allow us to use networks of computers as a single, unified computing resource. It is possible to cluster or couple a wide variety of resources including supercomputers, storage systems, data sources, and special classes of devices distributed geographically and use them as a single unified resource. This virtual computer, or better "network computer" is popularly known as a "computational Grid". But, before computational Grids, we used to call that kind of computing "metacomputing".

Catlett and Smarr have related the term metacomputing to "the use of powerful computing resources transparently available to the user via a networked environment". Their view is that a metacomputer is a networked virtual supercomputer. To an extent our usage of the term metacomputing still holds true to this definition apart from explicitly referring to "powerful" computing resources. Other terms have also been used to describe this computing paradigm, such as seamless, scalable or global computing and more recently Grid computing.

The concept of Grid computing started as a project to link supercomputing sites, but now it has grown far beyond its original intent. In fact, there are many applications that can benefit from the Grid infrastructure, including collaborative engineering, data exploration, high throughput computing, and of course distributed supercomputing.

Grid functions can be divided into two logical Grids: the *computational* Grid and the *access Grid*. Through the *computational Grid,* scientists will be able to access virtually unlimited computing and distributed data resources. The *access Grid* provides a group collaboration environment. Through a Web browser, users will be able to view and select all the Grid resources and services in a *virtual infinite machine room*. To build a Grid requires the development and deployment of a number of services, including those for: resource discovery, scheduling configuration management, security, and payment mechanisms in an open environment.

Grid applications (multi-disciplinary applications) couple resources that cannot be replicated at a single site or which may be remotely located for other practical reasons. These are some of the driving forces behind the inception of Grids. In this light, Grids let users solve larger or new problems by pooling together resources that could not be coupled easily before.

The *Grid* is not just a computing paradigm for providing computational resources for grand-challenge applications. It is also an infrastructure that can bond and unify globally remote and diverse resources ranging from meteorological sensors to data vaults, from parallel supercomputers to personal digital organisers. As such, it will provide pervasive services to all users that need them.

The idealised design features that are required by a Grid to provide users with a seamless computing environment are discussed below. There are three main issues that characterise computational Grids:

- *Heterogeneit*y: a Grid involves a multiplicity of resources that are heterogeneous in nature and might span numerous administrative domains across wide geographical distances.
- *Scalabilit*y: a Grid might grow from few resources to millions. This raises the problem of potential performance degradation as a Grids size increases. Consequently, applications that require a large number of geographically located resources must be designed to be extremely latency tolerant.
- *Dynamicity or Adaptabilit*y: in a Grid, a resource failure is the rule, not the exception. In fact, with so many resources in a Grid, the probability of some resource failing is naturally high. The resource managers or applications must tailor their behaviour dynamically so as to extract the maximum performance from the available resources and services.

The steps necessary to realise a computational Grid include:

- The integration of individual software and hardware components into a combined networked resource.
- The implementation of *middleware* to provide a transparent view of the resources available.

- The development of tools that allows management and control of Grid applications and infrastructure.
- The development and optimisation of distributed applications to take advantage of the resources.

The components that are necessary to form a Grid are briefly discussed below:

- *Grid Fabri*c: It comprises all the resources geographically distributed (across the globe) and accessible from anywhere on the Internet. They could be computers (such as PCs or Workstations running operating systems such as UNIX or NT), clusters (running cluster operating systems or resource management systems such as LSF, Condor or PBS), storage devices, databases, and special scientific instruments such as a radio telescope.
- *Grid Middlewar*e: It offers core services such as remote process management, co-allocation of resources, storage access, information (registry), security, authentication, and Quality of Service (QoS) such as resource reservation and trading.
- *Grid Development Environments and Tool*s: These offer high-level services that allow programmers to develop applications and *brokers* that act as user agents that can manage or schedule computations across global resources.
- *Grid Applications and Portals*: They are developed using Grid-enabled languages such as HPC++, C, Fortran, Java and message-passing systems such as MPI. Applications, such as parameter simulations and grand-challenge problems often require considerable computational power, require access to remote data sets, and may need to interact with scientific instruments. *Grid portals* offer web-enabled application services — i.e., users can submit and collect results for their jobs on remote resources through a web interface.

Although global Grids are still in their infancy, pilot projects are receiving a great deal of attention. A global Grid solution consists of a common set of rules and protocols required for the sharing of data and resources in a manner acceptable to all the individual campus Grids (such as defined in Globus). The global Grid framework must fulfil certain basic requirements:

- Security in both the computer centre and network.
- Flexibility in the sharing of resources.
- The ability to manage and account for a wide range of resources.
- The sharing of programs, data, and computers.
- Agreements on performance and cost.

Building a global Grid begins when all the resources are “Grid ready.” For the most part the Grid can rely on the basic TCP/IP protocol stack (IP, TCP, UDP, DNS, etc.) for transport, routing and naming requirements; it can rely on FTP for bulk data transfer (which supports third-party transfers). In addition the user should have a single sign-on (authentication) to be able to access a wide range of Grid resources via one of a number of different security protocols such as Kerberos or Unix security. Finally, the resources must be able to identify themselves using a protocol such as LDAP.

Next, a layer of middleware is needed. The middleware must be able to identify and authenticate users, discover and monitor resources, provide remote data access and data movement, and act as a cross-organisation resource broker and job scheduler. This is the layer that is designed to take the user’s requirements and match them with available resources anywhere within the global Grid using a set of resource and connectivity protocols. These network connections must have a wide range of bandwidths and latencies, operating systems, security software, and other resources, so it is important that the middleware be able to deal with all these possibilities. Specifically, this layer must provide:

- Directory services
- Allocation, scheduling and brokering services

- Monitoring, error handling and diagnostics
- Transparent data movement
- Software discovery services
- Grid policy enforcement

The third step in a global Grid solution deals with a resource scheduling and management system, which must be installed on each individual virtual organisation/campus Grid to receive the data and control information. This should be built upon a local campus resource management system (called small-scale Grids in this document) such as Sun Grid Engine, PBS, LSF, or NQE. The only requirement for the local resource managers is that they provide interfaces to the global Grid scheduler to trap requests from the global Grid and translate the requirements into actual job scheduling. Then they must execute the job and return any requested data.

Finally, to make this process seamless to the end user, a portal interface (such as the one described above) is needed to present choices and information about all the different phases in a simple and comprehensible manner. This portal can be application specific, or can represent a conglomeration of information about work being performed across the Grid.

Further we describe these Grid implementations which are already in use or can realistically be implemented within the next couple of years.

II.3 Basic and Advanced Portal Computing

II.3.1 Introduction

To solve the growing number of increasingly complex scientific, engineering, and business problems, a new model of computing has developed. In the past, data was created, stored, processed, and analysed at a single site on a campus. Later, with workstations becoming ubiquitous within departments, small amounts of data and computing power began to be distributed around the campus. At the time, the workstations were connected by a slow network and data was distributed either via FTP or NFS, which left each user to obtain what data they needed and compute locally on their desktop machine or send off jobs to a central computer. That model of distributing data and computing was extremely wasteful because desktop computers are generally under-utilised and because important computing and data resources were distributed all over the world. A new model was needed to harness those resources and thus we have the Grid model of computing.

Closely aligned with the concept of Grid computing is the idea of “portal computing”. A user or application portal is a web-based collection of information presented on a browser page. In many cases a portal can replace the need for users to log on to a number of different computers to gain access to the various resources that exist. A portal also allows IT managers to focus on maintaining services rather than allocating resources to individual users. In portal computing, well-defined services are delivered to the portals instead of giving users direct access to the UNIX prompt. As a result, resource management and security can be exercised at a higher level.

A Computing Portal – or equivalently a web-based Problem Solving Environment or PSE is an application that integrates access to the data, computers and the tools needed for a particular computational science area (Computing Portals, web). It must provide several services including security, fault tolerance, object lookup and registration, object persistence and database support (as in EIP’s), event and transaction services, job status (as in HotPage from NPACI and myGrid from NCSA), file services (as in NPACI Storage Resource Broker, support for computational science specific meta-data like MathML and XSIL, visualisation, application integration (chaining services viewed as backend compute filters), “seamless access” and integration of resources between different users/application domains, parameter specification service (get data from Web form into Fortran program wrapped as backend object), and finally collaboration. The technologies used (Java, Servlets, Applets, CGI etc.) as well as a number of services and features of the portal will decide whether we should call it a basic or an advanced portal.

We illustrate in the following figures the functional architecture of conceptual portals in three areas – Mesh generation, Space data analysis and Earthquake Science (after G.C. Fox).

In Figure 3, page 23, collaboration with experts is emphasised as mesh generation is still hard to automate and we assume that this will require input from a centre with expertise in this technique (which underlies the solution of partial differential equations from many fields). In Figure 4, page 23, we indicate the importance of multiple user interfaces as space missions will not wait, and interaction with users can be needed at any time. Further we try to isolate access to the Space Internet as one service but otherwise use infrastructure (databases, compute servers) from less esoteric applications. Let us focus on the last case GEM (General Earthquake model) which is a portal supporting all aspects of earthquake science from real time interaction with data from the “big-one” to decade long theoretical studies of the underlying non-linear systems. Most of the features of GEM (which we have studied quite deeply) translate to other areas such as those of Figure 3, page 23 and Figure 4, page 23.

In GEM, everything is a “distributed object” whether it be a simulation on a supercomputer; the basic GEM Web pages; the notes from a field trip entered on a palm top; CNN real-time coverage of the latest earthquake or the data streaming in from sensors. GEM provides an integrated view of these diverse resources with XML definitions for the raw objects themselves and the data they produce. The services shown in Figure 5, page 24, from collaboration, security, object discovery, visualisation and computer access, are generic to all computing portals.

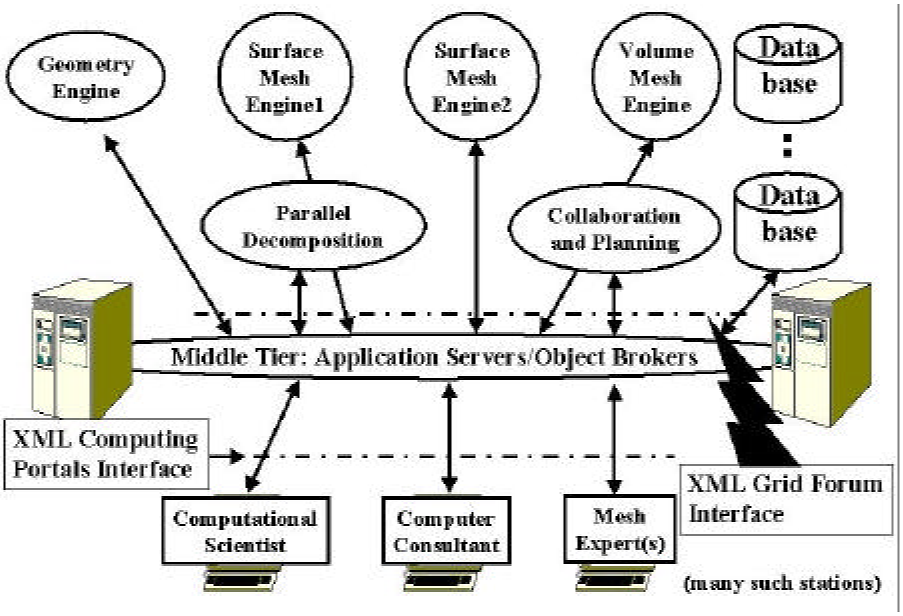


Figure 3: a portal for mesh generation

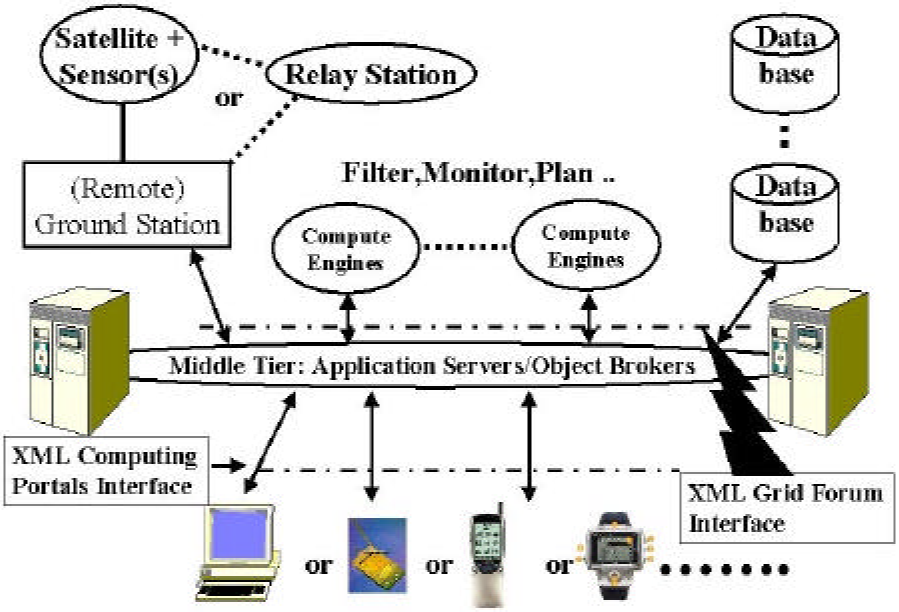


Figure 4: SMOP or Space Mission Operations Portal

Building GEM using the same approach and tools as other portals ensures the availability of these services. They will require customisation as (for example) there are many different visualisation packages and each requires non-trivial work to include in such a portal. Again, collaboration corresponds to sharing distributed objects, and as discussed in Section 3, this can currently only be automated for some objects. Many web pages can be shared using the generic techniques illustrated in Figure 5, page 24, but sharing (say) the control and output of a general simulation can require quite a lot of custom modifications.

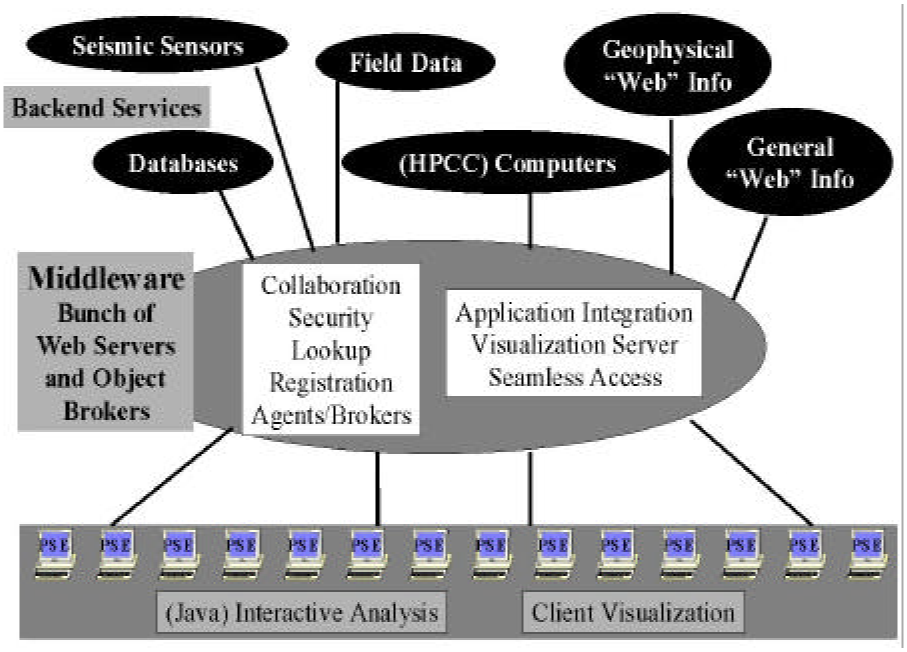


Figure 5: 3-tier architecture of a GEM computing portal

One needs to define the entities in the GEM environment as distributed objects, and for computer programs this implies a rather arcane process termed "wrapping the program as a distributed object". Operationally this implies allowing a middle-tier server (the CORBA object broker or Java application Server) to be able to run the program on one or more machines, specify the input files, and either specify output files or access them as streams of data in the fashion of UNIX pipes. The strategy is to define all relevant properties of computer programs in XML. These properties are used to generate either statically or dynamically the needed object wrappers. This approach requires the user to specify exactly what they know about the properties of their program, while the filter copes with the obscure syntax of each object model. Obviously this also allows one to support all object models – COM, CORBA, Java, by changing the filter and so one can adapt to changes in the commercial infrastructure used in the middle tier.

One must apply the XML object definition strategy to all entities in GEM; programs, instruments and other data sources and repositories. This gives the meta-data which defines macroscopically the object structure. However, equally usefully one need to look at the data stored in, produced by, or exchanged between these objects. This data is itself typically a stream of objects – each an array, a table or more complex data structure. One could choose to treat the data at some level as an unspecified (binary) "blob" with XML defining the overall structure but detailed input and output filters used for the data blobs. As an example, consider the approach that an electronic news organisation could take for their data. The text of news flashes would be defined in XML but the high volume multimedia data (JPEG images and MPEG movies) would be stored in binary fashion with XML used to specify <IMAGEOBJECT> or <MOVIEOBJECT> meta-data. Systematic use of XML allows use of a growing number of tools to search, manipulate, store persistently and render the information. It facilitates the linkage of general and specific tools/data sources/programs with clearly defined interfaces. This will help the distributed computing portal users to separately develop programs or generate data, which will be easily able to interoperate. XML standards will be defined hierarchically starting with distributed information systems, then general scientific computing and finally application specific object specifications. For example GEM would develop its own syntax for seismic data sensors but could build on general frameworks like the XSIL scientific data framework developed at Caltech. XSIL supports natural scientific data structures like arrays and the necessary multi-level storage specification. Another example is MathML which provides XML support for the display and formulation of mathematics. We can expect MathML to be supported by tools like web browsers and white boards in collaborative scientific notebooks and this will enhance theoretical collaboration within GEM. For instance, there will be modules that can be inserted into applications for parsing MathML or providing

II.4 HotPage Portal

II.4.1 Introduction

Another example of portal computing is the GridPort toolkit developed by Mary Thomas and co-workers at the San Diego Supercomputer Centre and the National Partnership for Advanced Computational Infrastucture (NPACI). User portals built on top of GridPort allow users to run codes, manipulate data and files, and otherwise use components of the computational Grid through a web interface. The web pages on the client only require simple HTML and JavaScript. Therefore, any browser can be used to view the pages, while on the server the services are built using Perl and CGI scripts. Behind the scenes GridPort uses technologies such as Globus and PKI to provide secure, interactive services (Visit the NPACI HotPage portal at http://hotpage.npaci.edu/).

HotPage is the Grid Computing Portal. HotPage enables researchers to find information about each of the resources in the NPACI computational Grid: technical documentation, operational status, load and current usage, queued jobs, etc. HotPage enables researchers who obtain HotPage accounts to use these resources from within HotPage to access and manipulate their files and data, and to submit, monitor, and delete jobs.

The Grid Portal Toolkit (GridPort) enables the rapid development of web portals on computational Grids. GridPort leverages both commodity technologies and metacomputing, or Grid, software (Globus, Network Weather Service, etc.) to provide secure methods for manipulating files and data, submitting jobs, and managing allocations on all Grid resources for which a user has privileges.

The NPACI HotPage is a user portal built using the GridPort Toolkit that provides views of a distributed set of HPC resources as either an integrated meta-system, or as individual machines. GridPort is being used to provide the foundation for a new *user portal*: The NPACI Genie Portal (GridPort Enabled Interactive Environment), formerly the HotPage. These web pages run on any web browser, regardless of system or geographical location and are supported by secure, encrypted login sessions where authenticated users can access their HPC system accounts and perform basic computational tasks. Users can directly access multiple compute resources by logging in once. They have secure access to all of their accounts, files, and data, and all interactions are point-and-click in a web browser. This enables 'web-based supercomputing,' which greatly simplifies the scientific process of running simulations on diverse resources.

In the GridPort model, we distinguish between 3 types of portals: the service portal, the user portal, and the application portal. The service portal is that part of the system that supports services such as authentication, account management, and job submission. Application portals are built to access the portal services via form actions calling CGI scripts. Thus, users can build their own portal without being experts in IT and Grid services.

There are both information services and interactive services provided by the HotPage user portal. The HotPage portal is designed to be a single point-of-access to compute resources. The layout of the page allows a user to view these resources from both the Grid and individual machine perspectives and to quickly determine system-wide information like status (machines up/down, machine load, nodes/cycles available, etc.). This design provides a simple, fast, and direct access to HPC systems and general information about them, with a focus on getting users working on NPACI systems quickly.

The information services provide a user-oriented interface to NPACI resources and services. They consist of on-line documentation, static information pages, and links to events within NPACI, including basic user information such as documentation, training, and news. As part of the *information services*, there are user tools that provide real-time information for each machine, including: operational status and utilisation of all resources; summaries of machine status, load, and batch queues; displays of currently executing and queued jobs; and graphical display of running applications mapped to nodes. The system for acquiring this data will be migrated to an LDAP/GIS model.

Interactive services are the secure transactions that provide users with direct access to the HPC compute resources and allows the user to perform many task that could be run on a Unix command line, including: command execution, including compiling and running programs; job submission and

- Communication methods - this will include descriptions of the mechanisms required for efficient implementation of, e.g. message passing and RPC, as well as characterisation and selection of available network connections:
  - communication models;
  - communication protocols;
  - Quality of Service (QoS);
  - fault tolerance;
  - services supporting collaboration and remote instrument control (such as secure, reliable group communication);
  - application support, APIs.

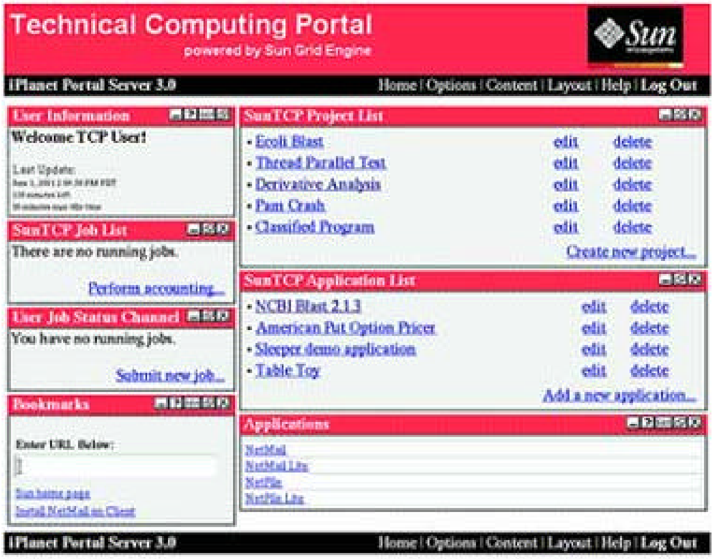
- Data management and integrity - the provision of uniform, efficient access to distributed or replicated data:
  - data access;
  - data migration and replication;
  - data storage;
  - data transfer;
  - meta-data management (data cataloguing and publishing - ability to automatically generate meta-data, and management of use conditions and access);
  - application support, APIs.

- Security - authentication, authorisation and accounting services for users, as well as confidentiality and integrity for data transfers:
  - authentication mechanisms;
  - authorisation mechanisms;
  - accounting;
  - credential delegation;
  - application support.

This part is divided into seven sections. The first section will describe the taxonomy of the Grid. In the subsequent sections different models will be described.

graphical user specification of mathematical formulae. One can also use MathML in high level tools allowing specification of basic differential equations that are translated into numerical code. This has been demonstrated in prototype problem solving environments like PDELab but have so far not had much practical application. Greater availability of standards like MathML should eventually allow more powerful interchangeable tools of this type. Finally we can mention a set of graphical XML standards such as X3D (3 dimensional objects) SVG and VML which are vector graphics standards, which can be expected to be important as basis of application specific plot and drawing systems.

Those who prefer to use off-the-shelf software will soon have another option: the Technical Computing Portal. This portal, which is currently under development, will integrate the iPlanet portal server with the Sun Grid Engine. Instead of selecting the computing queue of a specific system on the Internet, all the user will have to do is submit his job using the Technical Computing Portal. The Sun Grid Engine will then find the appropriate resource and monitor execution. A prototype of the Technical Computing Portal is shown in the illustration below.



Technical Computing Portal

powered by Sun Grid Engine

iPlanet Portal Server 3.0 | Home | Options | Content | Layout | Help | Log Out

User Information

Welcome TCP User!

Last Update:

SunTCP Job List

There are no running jobs.

Perform accounting...

User Job Status Channel

You have no running jobs.

Submit new job...

Bookmarks

Enter URL Below:

Sun home page

Install NetMail on Client

SunTCP Project List

• Ecoli Blast	edit	delete
• Thread Parallel Test	edit	delete
• Derivative Analysis	edit	delete
• Pam Crash	edit	delete
• Classified Program	edit	delete

Create new project...

SunTCP Application List

• NCBI Blast 2.1.3	edit	delete
• American Put Option Pricer	edit	delete
• Sleeper demo application	edit	delete
• Table Toy	edit	delete

Add a new application...

Applications

NetMail

NetMail Lite

NetFile

NetFile Lite

iPlanet Portal Server 3.0 | Home | Options | Content | Layout | Help | Log Out

Figure 6: a technical computing portal

deletion; file and directory acces s; account and allocation management; and file transfer between the local workstation and the HPC resources. All secure transactions, including login, are accessed through a separate web server running services based on the GridPort Toolkit.

The GridPort Toolkit (version 1.0) consists of two modules. The web portal services module runs on a commercial web server, and provides authenticated connections to the Grid that are used by client applications. The client interface tools provide a WWW interface that allows customised science portal development by end users who do not need to have any knowledge of the underlying portal infrastructure. A key feature of the architecture is that the client application and the Grid portal services can be run on separate web servers.

Because web interfaces are common, well understood, and pervasive, client portals based on the GridPort architecture are accessible wherever there is a web browser, no matter where the user is located.

On the web server, web pages are either stored as static HTML (with server-side includes), or built dynamically, using Perl scripts. User input information is obtained via form input elements, so CGI must be supported. Most of the back-end portal functionality is implemented in Perl5/CGI as well. In addition, Perl runs on most HPC systems, making it easier to develop a set of uniform, compatible, and portable scripts.

Both modules are based on commodity web technologies and existing Grid services and applications such as Communicator and Internet Explorer (4.0 or greater), HTML/JavaScript Perl/CGI, SSH, FTP, GSI, Globus, etc. The choice of which Grid technology to use is determined by a broker layer. Based on such simple technology, this user portal technology is easily ported to and used by other sites. In addition, the GridPort Toolkit does not require additional services to be run on the HPC Systems.

By using the GridPort Toolkit, an application programmer can extend the set of basic functions provided by the HotPage. Thus, users can also construct customised web pages and program directly to existing portal services, given a basic knowledge of HTML and/or Perl/CGI.

II.4.2 Contract specification methods

There is no information about the contract specification.

II.4.3 Resource management

Globus (described in more details in the Globus section).

Grid information services: Globus MDS. Informational services allow any user to look at the portal and get machine status (load, node usage, up/down, etc.)

Brokerage services: Users can perform basic computational tasks such as Unix commands, can submit, monitor, and cancel jobs in the batch queues. Template batch script generator available for all HPC systems. Future: SDSC Storage Resource Broker.

Advanced reservation: Not supported.

Scheduling mechanisms: None.

Co-scheduling: Globus co-allocation mechanisms (described in more details in the Globus section).

Application support:

- User toolkit so they can build their own application portal.
- User Portals - the PACI computational science community.
- Applications: LAPK is a group of MD's, Gamess is used by computational chemists.

II.4.4 Communication methods

http, https.

Communication models: Web based.

Communication protocols: http, https.

Quality of Service: None.

Fault tolerance: None.

Services supporting collaboration and remote instrument control: None.

Application support: User Portals - the PACI computational science community.

II.4.5 Data management and integrity

Data access: Web based.

Data migration: None.

Replication: None.

Data storage: Future plans: SRB (Storage Resource Broker).

Data transfer:

- upload/download from local to host to remote HPC system.
- maintain portal file spaces for users.
- tar/untar, compress/uncompress.

Meta-data management: None.

Application support: None.

II.4.6 Security

Encrypted login sessions via HTTPS/SSL and Globus GSI. Supports single login environment across all portals.

Authentication mechanism: Authenticated users can access all the HPC systems at NPACI, NCSA, and NASA/IPG.

Authorisation mechanisms: Information services allow any user to glance at portal and get machine status (load, node usage, up/down, etc.). Any user with an account on the systems connected into our network can run jobs on them, provided that their accounts are active.

Accounting:

- basic customisation features;
- job tracking (limited);
- incorporating HPC resource account management features.

Credential delegation: My_proxy mechanisms (described in more details in the Globus section).

II.5 Globus

II.5.1 Introduction

The Globus project is a joint effort of the Argonne National Laboratory and the University of Southern California Information Sciences Institute, that addresses Grid technology.

A central element of the Globus system is the Globus Metacomputing Toolkit, which defines the basic services and capabilities required to construct a computational Grid. The design of this toolkit was guided by the following basic principles:

- The toolkit comprises a set of components that implement basic services for security, resource location, resource management, communication, etc. The services currently defined by Globus are listed in Table 2. Computational Grids must support a wide variety of applications and programming models. Hence, rather than providing a uniform programming model, such as the object-oriented model defined by the Legion system, the Globus toolkit provides a bag of services from which developers of specific tools or applications can make selections which meet their needs.

Service	*Name*	*Description*
Resource management	GRAM	Resource allocation and process management
Resource co-allocation	DUROC	Resource co-allocation
Communication	Nexus	Unicast and multicast communication services
Security	GSI	Authentication and related security services
Information	MDS	Distributed access to structure and state information
Health and status	HBM	Monitoring of health and status of system components
Remote data access	GASS	Remote access to data via sequential and parallel interfaces
Executable management	GEM	Construction, caching, and location of executables

Table 2: core Globus services

- The toolkit distinguishes between local services, which are kept simple to facilitate deployment, and global services, which are constructed on top of local services and may be more complex. Computational Grids require that a wide range of services be supported on a highly heterogeneous mix of systems and that it be possible to define new services without changing the underlying infrastructure. An established architectural principle in such situations, as exemplified by the Internet Protocol suite, is to adopt a layered architecture with an “hourglass” shape. A simple, well-defined interface – the neck of the hourglass provides uniform access to diverse implementations of local services. Higher-level global services are then defined in terms of this interface. To participate in a Grid, a local site need provide only the services defined at the neck, and new global services can be added without local changes.

- Interfaces are defined so as to manage heterogeneity, rather than hiding it. These so-called translucent interfaces provide structured mechanisms by which tools and applications can discover and control aspects of the underlying system. Such translucency can have significant performance advantages because, if an implementation of a higher-level service can understand characteristics of the lower-level services on which the interface is layered, then the higher-level service can either control specific behaviours of the underlying service or adapt its own behaviour to that of the underlying service. Translucent interfaces do not imply complex interfaces.

- An information service is an integral component of the toolkit. Computational Grids are in a constant state of flux as utilisation and availability of resources change, computers and networks fail, old components are retired, new systems are added, and software and hardware on existing systems are updated and modified. It is rarely feasible for programmers to rely on standard or default configurations when building applications. Rather, applications must discover characteristics of their execution environment dynamically and then either configure aspects of system and application behaviour for efficient, robust execution or adapt behaviour during program execution. A fundamental requirement for discovery, configuration, and adaptation is an information-rich environment that provides pervasive and uniform access to information about the current state of the Grid and its underlying components. In the Globus toolkit, a component called the Metacomputing Directory Service, fulfils this role.

- The toolkit uses standards whenever possible for both interfaces and implementations. Globus developers envision computational Grids as supporting an important niche of applications that must coexist with more general-purpose distributed and networked computing applications such as CORBA, DCE, DCOM, and Web-based technologies. The Internet community and other groups are moving rapidly to develop official and de facto standards for interfaces, protocols, and services in many areas relevant to computational Grids. There is considerable value in adopting these standards whenever they do not interfere with other goals. Consequently, the Globus components are not, in general, meant to replace existing interfaces, but rather seek to augment them.

II.5.2 Application and Resource Description

The Globus Resource Specification Language (RSL) provides a common interchange language to describe resources. The various components of the Globus Resource Management architecture manipulate RSL strings to perform their management functions in co-operation with the other components in the system. The RSL provides the skeletal syntax used to compose complicated resource descriptions, and the various resource management components introduce specific <attribute, value> pairings into this common structure. Each attribute in a resource description serves as a parameter to control the behaviour of one or more components in the resource management system.

```
specification := request
request       := multirequest j conjunction j disjunction j parameter
multirequest  := + request-list
conjunction   := & request-list
disjunction   := | request-list
request-list  := ( request ) request-list j ( request )
parameter     := parameter-name op value
op            := = j > j < j >= j <= j !=
value         := ([a..Z][0..9][ ])+
```

Figure 7: BNF grammar describing the syntax of a RSL request

The syntax of an RSL specification, summarised in Figure 7, page 31, is based on the syntax for filter specifications in the Lightweight Directory Access Protocol (LDAP) and MDS. An RSL specification is constructed by combining simple parameter specifications and conditions with the operators &, to specify conjunction of parameter specifications, |, to express the disjunction of parameter specifications, or +, to combine two or more requests into a single compound request, or multirequest.

The set of parameter-name terminal symbols is extensible: resource brokers, co-allocators, and resource managers can each define a set of parameter names that they will recognise. For example, a resource broker that is specialised for tele-immersive applications might accept as input specification containing a frames-per-second parameter and generate as output a specification containing a Mflops-per-second parameter, to be passed to a broker that deals with computational resources. Resource managers, the system components that actually talk to local scheduling systems, recognise two types of parameter-name terminal symbols:

- MDS attribute names, used to express constraints on resources: e.g., memory>=64, network=atm. In this case, the parameter name refers to a field defined in the MDS entry for the resource being allocated. The truth of the parameter specification is determined by comparing the value provided with the specification with the current value associated with the corresponding field in the MDS. Arbitrary MDS fields can be specified by providing their full distinguished name.

- Scheduler parameters, used to communicate information regarding the job, such as count (number of nodes required), max time (maximum time required), executable, arguments, directory, and environment (environment variables). Schedule parameters are interpreted directly by the resource manager.

For example, the following specification:

```
&(executable=myprog)
(|(&(count=5)(memory>=64))(&(count=10)(memory>=32)))
```

requests 5 nodes with at least 64 MB memory, or 10 nodes with at least 32 MB. In this request, executable and count are scheduler attribute names, while memory is an MDS attribute name.

Our current RSL parser and resource manager disambiguate between these two parameter types on the basis of the parameter name. That is, the resource manager knows which fields it will accept as scheduler parameters and assumes all others are MDS attribute names. Name clashes can be resolved by using the complete distinguished name for the MDS field in question.

The ability to include constraints on MDS attribute values in RSL specifications is important. The state of resource managers is stored in MDS. Hence, resource specifications can refer to resource characteristics such as queue-length, expected wait time, number of processors available, etc. This technique provides a powerful mechanism for controlling how an RSL specification is interpreted.

The following example of a multirequest is derived from the example shown above.

```
+(&(count=80)(memory>=64M)(executable=sf_express)
(resourcemanager=ico16.mcs.anl.gov:8711))
(&(count=256)(network=atm)(executable=sf_express)
(resourcemanager=neptune.cacr.caltech.edu:755))
(&(count=300)(memory>=64M)(executable=sf_express)
(resourcemanager=modi4.ncsa.edu:4000))
```

This is a ground request: every component of the multirequest specifies a resource manager. A co-allocator can use the resource manager parameters specified in this request to determine to which resource manager each component of the multirequest should be submitted.

II.5.3 Contract specification methods

There is no information about the contract specification. Within the GrADS project some solution for contract specification for the Cactus application has been proposed, but no extensive information is available. A mechanism for contract monitoring has been introduced.

II.5.4 Resource management

The Globus approach to the Grid resource management problem is illustrated in Figure 8. In this architecture, an extensible resource specification language (RSL), discussed later, is used to communicate requests for resources between components: from applications to resource brokers, resource co-allocators and resource managers. At each stage in this process, information about resource requirements is coded in an RSL expression by the application or refined by one or more resource brokers and co-allocators. Information about resource availability and characteristics is obtained from an information service.

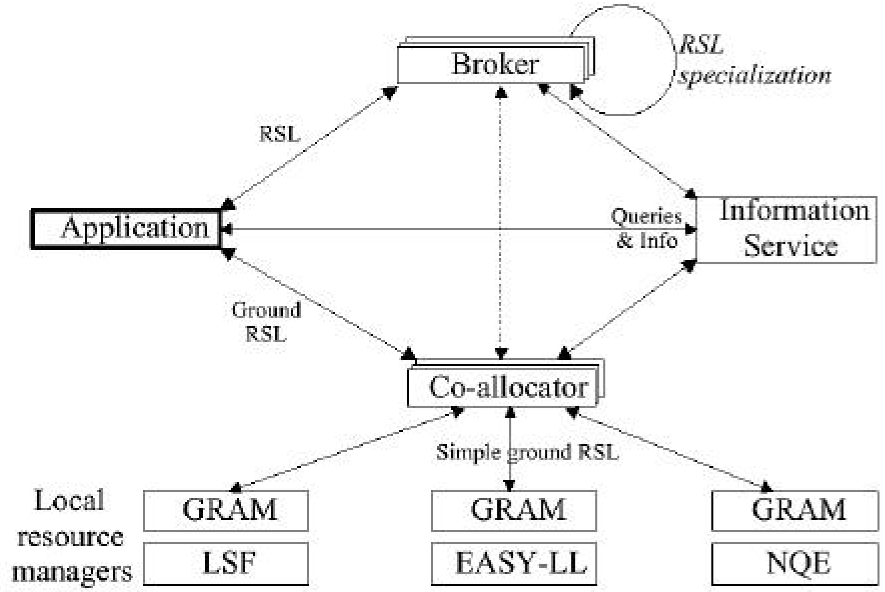


Figure 8: Globus resource management architecture

Resource brokers are responsible for taking high-level RSL specifications and transforming them into more concrete specifications through a process called specialisation. Multiple brokers may be involved in servicing a single request, with application-specific brokers translating application requirements into more concrete resource requirements, and different resource brokers being used to locate available resources that meet those requirements.

Transformations effected by resource brokers generate a specification in which the locations of the required resources are completely specified. Such a ground request can be passed to a co-allocator, which is responsible for co-ordinating the allocation and management of resources at multiple sites Resource co-allocators break down a multirequest – that is, a request involving resources at multiple sites – into its constituent elements, and pass each component to the appropriate resource manager. Each resource manager in the system is responsible for taking an RSL request and translating it into operations in the local, site-specific resource management system.

Grid information services: The information service is responsible for providing efficient and pervasive access to information about the current availability and capability of resources. This information is used to locate resources with particular characteristics, to identify the resource manager associated with a resource, to determine properties of that resource, and for numerous other purposes during the process of translating high-level resource specifications into requests to specific managers.

Globus uses Metacomputing Directory Service (MDS) as an information service. MDS uses the data representation and application programming interface (API) defined on the Lightweight Directory Access Protocol (LDAP) to meet requirements for uniformity, extensibility, and distributed maintenance. It defines a data model suitable for distributed computing applications, which is able to represent computers and networks of interest, and which also provides tools for populating this data model. LDAP defines a hierarchical, tree-structured name space called a directory information tree (DIT). Fields within the namespace are identified by a unique distinguished name (DN). LDAP supports both distribution and replication. Hence, the local service associated with MDS is just an LDAP server (or a gateway to another LDAP server, if multiple sites share a server), plus the utilities used to populate this server with up-to-date information about the structure and state of the resources within that site. The global MDS service is simply the ensemble of all these servers. An advantage of using MDS as the information service is that resource management information can be used by other tools.

Local resource management: The Globus Resource Allocation Manager (GRAM) is the lowest level of Globus resource management architecture. GRAM allows you to run jobs remotely, providing an API for submitting, monitoring, and terminating your job.

A job is submitted and the request is sent to the gatekeeper of the remote computer. The gatekeeper handles the request and creates a job manager for the job. The job manager starts and monitors the remote program, communicating state changes back to the user on the local machine. When the remote application terminates, normally or by failing, the job manager terminates as well.

GRAM is responsible for:

- Parsing and processing the Resource Specification Language (RSL) specifications that outline job requests. The request specifies resource selection, job process creation, and job control. This is accomplished by either denying the request or creating one or more processes (jobs) to satisfy the request.
- Enabling remote monitoring and managing of jobs already created.
- Updating Metacomputing Directory Service (MDS) with information regarding the availability of the resources it manages.

To run a job remotely, a GRAM gatekeeper (server) must be running on a remote computer, listening at a port; and the application needs to be compiled on that remote machine. The execution begins when a GRAM user application runs on the local machine, sending a job request to the remote computer. The executable, stdin and stdout, as well as the name and port of the remote computer, are specified as part of the job request. The job request is handled by the gatekeeper, which creates a job manager for the new job. The job manager handles the execution of the job, as well as any communication with the user.

As indicated above, a GRAM is intended to serve as the interface between a wide area metacomputing environment and an autonomous entity able to create processes, such as a parallel computer scheduler or a Condor pool. Notice that this means that a resource manager need not correspond to a single host or a specific computer, but rather to a service that acts on behalf of one or more computational resources.

A resource specification passed to a GRAM is assumed to be ground: that is, to be sufficiently concrete that the GRAM can identify local resources which will meet the specifications without the need for further interaction with the entity that ge nerated the request. A particular GRAM implementation may achieve this goal by scheduling resources itself or, more commonly, by mapping the resource specification into a request to some local resource allocation mechanisms. (To date, there are GRAM interfaces to six different schedulers or resource allocators: Condor, EASY, Fork, LoadLeveler, LSF, and NQE.) Hence, the GRAM API plays for resource management a similar role to that played by IP for communication: it can co-exist with local mechanisms, just as IP rides on top of Ethernet, FDDI, or ATM networking technology.

The GRAM implementations have the structure shown in Figure 9.

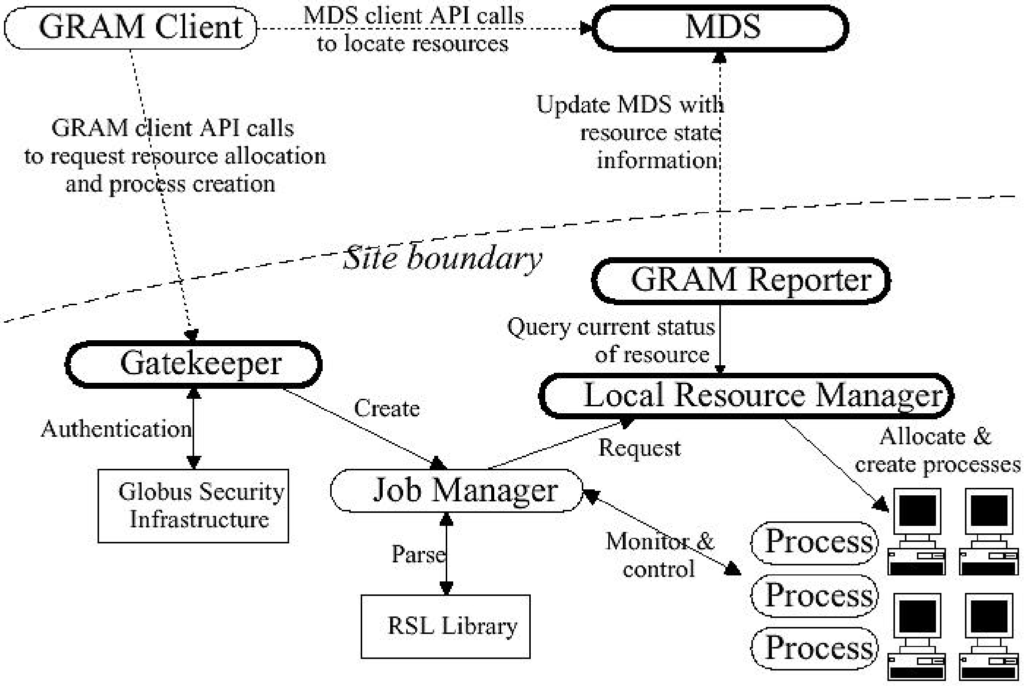


Figure 9: major components of the GRAM implementation

The principal components are the GRAM client library, the gatekeeper, the RSL parsing library, the job manager, and the GRAM reporter. The Globus security infrastructure (GSI) is used for authentication and for authorisation.

The GRAM client library is used by an application or a co-allocator acting on behalf of an application. It interacts with the GRAM gatekeeper at a remote site to perform mutual authentication and transfer a request, which comprises a resource specification, a callback (described below), and a few other components that are not relevant to the current discussion.

The gatekeeper is an extremely simple component that responds to a request by doing three things: performing mutual authentication of user and resource, determining a local user name for the remote user, and starting a job manager which executes as that local user and actually handles the request. The first two security-related tasks are performed by calls to the Globus security infrastructure (GSI), which handles issues of site autonomy and substrate heterogeneity in the security domain. In order to start the job manager, the gatekeeper must run as a privileged program. On Unix systems, this is achieved via suid or inetd. However, because the interface to the GSI is small and well defined, it is easy for organisations to review and approve the gatekeeper code. In fact, the gatekeeper code has successfully undergone security reviews at a number of large supercomputer centres. The mapping of remote user to locally recognised user name minimises the amount of code that must run as a privileged program; it also allows most authorisation issues to be delegated to the local system.

A job manager is responsible for creating the actual processes requested by the user. This task typically involves submitting a resource allocation request to the underlying resource management system, although if no such system exists on a particular resource, a simple fork may be performed. Once processes are created, the job manager is also responsible for monitoring the state of the created processes, notifying the callback contact of any state transitions, and implementing control operations such as process termination. A job manager terminates once the job for which it is responsible has terminated.

The GRAM reporter is responsible for writing into MDS various information about scheduler structure (e.g., whether the scheduler supports reservation and the number of queues) and state (e.g., total number of nodes, number of nodes currently available, currently active jobs, and expected wait time in a queue). An advantage of implementing the GRAM reporter as a distinct component is that MDS reports can continue even when no gatekeeper or job manager is running: for example, when the gatekeeper is run from inetd.

As noted above, GRAM implementations have been constructed for six local schedulers to date: Condor, LSF, NQE, Fork, EASY, and LoadLeveler. Much of the GRAM code is independent of the local scheduler and so only a relatively small amount of scheduler-specific code needed to be written in each case. In most cases, this code comprises shell scripts that make use of the local scheduler's user-level API. State transitions are mostly handled by polling, because this proved to be more reliable than monitoring job process by using mechanisms provided by the local schedulers.

Brokerage services: The term resource broker is used to denote an entity in Globus architecture that translates abstract resource specifications into more concrete specifications. This definition is broad enough to encompass a variety of behaviours, including application-level schedulers that encapsulate information about the types of resource required to meet a particular performance requirement, resource locators that maintain information about the availability of various types of resource, and (ultimately) traders that create markets for resources. In each case, the broker uses information maintained locally, obtained from MDS, or contained in the specification to tune the specification, thus creating a new specification that contain more detail. Requests can be passed to several brokers, effectively composing the behaviours of those brokers, until eventually the specification is fine-tuned to the point that it identifies a specific resource manager. This specification can then be passed to the appropriate GRAM or, in the case of a multirequest, to a resource co-allocator.

Such an architecture makes it straightforward to develop a variety of higher-level schedulers.

Quality of Service and Advanced reservation: The Globus Architecture for Reservation and Allocation (GARA) provides advance reservations and end-to-end management for quality of service on different types of resources, including networks, CPUs, and disks.

A GARA system comprises a number of resource managers that each implement reservation, ontrol, and monitoring operations for a specific resource. Resource managers can and have been implemented for a variety of resource types, hence the use of the term "resource manager" rather than the more specific "bandwidth broker" favoured in the networking literature.

Uniform interfaces allow applications to express QoS needs for different types of resources in similar ways, hence simplifying the development of end-to-end QoS management strategies.

Mechanisms provided by the Globus toolkit are used for secure authentication and authorisation of all requests to resource managers. An information service allows applications to discover resource properties such as current and future availability.

GARA extends the Globus resource management architecture in two major ways: it introduces the generic resource object which encompasses network flows, memory blocks, disk blocks, and other entities as well as processes. GARA also introduces the reservation as a first class entity in the resource management architecture. Various other architectural changes follow from these new concepts.

In GARA, the computation-specific allocation functions are reformulated in terms of general resource objects hence allowing different application components to be manipulated in common ways. A generic “create Object” operation is used to create a process, flow, disk object, memory object, according to the supplied arguments. Each “create Object” operation returns an object handle that can subsequently be used to monitor and control the object (e.g., to delete it). A system or application process can also request upcalls on specific events, such as reservation applied to object, object termination, or QoS contract violations. Upcalls allow the construction of adaptive systems.

GARA splits the task of creating a resource object into two phases: reservation and allocation. In the reservation phase, a reservation is created, which provides some confidence that a subsequent allocation request will succeed. However, no object is created at this time. Instead, a reservation handle is returned, that can be used to monitor and control the status of the reservation and that can also be passed to a subsequent “create Object” call in order to associate that object with the reservation. The introduction of a distinct reservation phase has two important ramifications. First, by splitting reservation from allocation, it enables us to perform advance reservation of resources, which can be critical to application success if a required resource is in high demand. Second, if reservation is cheaper than allocation (as is often the case for large parallel computers, for example), one can implement lighter-weight resource reservation strategies than if objects must be created in order to guarantee access to a resource. A reservation is created by a generic “create Reservation” operation, which interacts with local resource management elements to ensure that the requested quantity and quality of the resource will be available at the requested start time and will remain available for the desired duration. If the resource cannot make this assurance, the “create Reservation” operation fails. All “create Object” operations require a reservation in order to proceed. This reservation is normally created via a proceeding “create Reservation” all, but for some resources a default “best effort” reservation can be specified. Note that the GARA concept of reservation encompasses both immediate reservations, which are assumed to be followed immediately by an allocation, and advance reservations, which are created in the present to reserve resources for use in the future. Reservation and object creation operations are implemented by a renamed GRAM: the Globus Reservation and Allocation Manager. As illustrated in Figure 10, GARA introduces a new entity called a co-reservation. A co-reservation agent, like the co-allocation agent, is responsible for discovering a collection of resources that can satisfy application end-to-end QoS requirements (a resource set). However, rather than allocating those resources, it simply reserves them. Hence, a call to a co-reservation agent specifies QoS requirements and returns a set of reservation handles that can then be passed to a co-allocation agent.

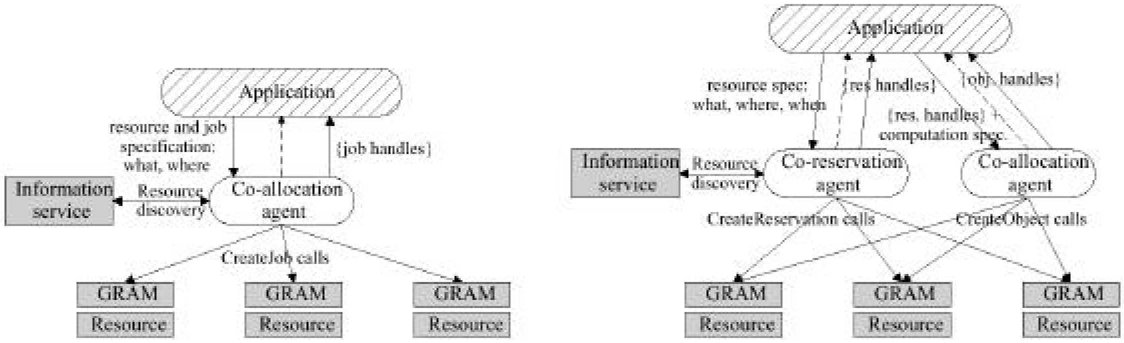


Figure 10: the Globus and GARA resource management architectures

In GARA, the co-allocation agent remains but now has the simpler task of allocating a resource set, given a reservation handle generated by a co-reservation agent. In practice it may need to either incorporate some aspects of co-reservation agent functionality, or interact with an external co-reservation agent, in order to recover from allocation failures that may occur.

In summary, GARA supports advance reservation directly. The introduction of generalised resource objects along with the standardised interface provide by GRAM addresses issues of heterogeneity in the resource set. Co-reservation and co-allocation agents are layered on top of GRAM and the standardised information services to enable the dynamic construction of collections of independently administered resources that satisfy application QoS requirements.

Co-reservation (and, to a lesser extent, co-allocation) agents play a critical role in GARA. They provide a bridge between the application and the available resources, constructing sets of resources that both match application QoS requirements and conform to the local practices and policies of resource providers.

Because GARA does not constrain agent design other than to define the GRAM and information service functions used to construct implementations, a wide range of agent architectures are possible. An agent can take the form of a library, linked with an application, that makes reservation decisions on behalf of that single application. Alternatively, an agent may be a global system oriented "broker" that provides reservation services to numerous users and applications. Functionality may be centralised or may be distributed over multiple agent instances or throughout a hierarchy of different agents. The latter organisation is useful, for example, if certain subsets of required resources are under the control of "local" reservation systems, such as bandwidth brokers. Finally, an agent can act autonomously, responding to a reservation request with either success or failure, or may proceed interactively, allowing a user or application to guide the construction of a resource set.

To clarify the role of co-reservation agents, we should return to the data analysis example of the introduction. Let us assume that while visualisation must occur at a specific location (the user computer), we can choose any one of several cached replicas of the data, and any one of several alternative analysis supercomputers. Reservation and object creation operations on these data stores, supercomputers, and network elements can be achieve by calls to appropriate GRAMs, although each system may of course implement these functions with quite different mechanisms.

The agent must now discover computational and bandwidth resources that can collectively provide the desired end-to-end QoS. Applications developed in the context of the current Globus system achieve this goal – for co-allocation – by using exhaustive search. In an advanced reservation environment, we can consider a range of future times, and so the number of candidate resources can be larger. Hence, efficient search heuristics will typically be required. For example, we can consider each potential data cache in turn, interrogating the information service for each to locate a supercomputer that can deliver the required computational power. At this point the agent may need to consider issues such as acceptable use and security policies, perhaps because data is proprietary. Then, the agent attempts to reserve both supercomputer nodes and network bandwidth between the supercomputer and the visualisation engine. If both reservations succeed, the agent can proceed to locate and reserve a network link between the supercomputer and the data cache.

This example illustrates an important aspect of end-to-end reservation, namely the importance that application-level criteria (e.g., end-to-end security requirements) can have for resource selection. The example also illustrates other issues that must be considered when developing agent strategies. For example, search procedures such as that just outlined can produce several alternative resource sets. If it does, we can choose between alternatives based on selection criteria such as first found and "best" found. If no suitable resources exist, the agent may either fail, or attempt to renegotiate with the user who might, for example, decide to proceed with fewer analysis than was originally desired, or with "best effort" rather than reserve bandwidth. In this example, reservation failure is handled by backtracking: we proceed to try alternative resources until either the request is successful or fails. In other situations, we may prefer to wait until all required resources are available. In this case, we need to be concerned about the possibility of deadlock, as other agents may attempt to acquire some of the same resources simultaneously. Deadlock can be avoided by using variants of well-known deadlock prevention and avoidance schemes, such as enforcing protocols for resource acquisition (e.g., processors sorted by IP address, followed by disks, followed by networks, etc) or timeout mechanisms.

The search and deadlock strategies just described are necessary because in this application there are dependencies among required resources. In other situations, simpler techniques can be used. For example, consider an application that simply requires N computational resources with a specified minimum network connectivity. If some number M from an attempted N reservations succeed (M <N), then we can either return to the information service to locate additional candidates or generate a callback to the application to determine whether it is possible to proceed with just M resources, a strategy used by the DUROC co-allocator when allocations fail.

Scheduling mechanisms: Globus resource management architecture does not solve the scheduling problems of Grid computing. However, the APIs allow the building of application level schedulers as well as the brokers for job scheduling.

Co-scheduling: Through the actions of one or more resource brokers, the requirements of an application are refined into a ground RSL expression. If the expression consists of a single resource request, it can be submitted directly to the manager that controls that resource. However, as discussed above, it is often the case that a metacomputing application requires that several resources – such as two or more computers and intervening networks – be allocated simultaneously. In these cases, a resource broker produces a multirequest and co-allocation is required. The challenge in responding to a co-allocation request is to allocate the requested resources in a distributed environment, across two or more resource managers, where global state, such as availability of a set of resources, is difficult to determine.

Within our resource management architecture, multirequests are handled by an entity called a resource co-allocator. In brief, the role of a co-allocator is to split a request into its constituent components, submit each component to the appropriate resource manager, and then provide a means for manipulating the resulting set of resources as a whole: for example, for monitoring job status or terminating the job. Within these general guidelines, a range of different co-allocation service can be constructed.

The currently developed co-allocator extends GRAM semantics to provide for simultaneous allocation of a collection of resources, enabling the distributed collection of processes to be treated as a unit.

DUROC – Dynamically-Updated Request Online Co-allocator – is a co-allocator of this type.

The task an intelligent co-allocation agent performs has two distinct (abstract) parts. First, the agent must process resource specifications to determine how a job might be distributed across the resources of which it is aware – the agent uses an abstract lower-level (or 'lowered') specification constructed in such a way that different portions of the specification are allocated to the different RMs that control access to those required resources.

Second, the agent must process the low-level resource specification as part of a job request to actually attempt resource allocation – the agent issues job requests to each of the pertinent RMs to schedule the job. The process of accessing a low-level resource specification in a job request in essence refines the request based on information available to the agent accessing the lower-level specification. By separating the tasks of refinement and allocation in the architecture, we can allow user intervention to adjust the refinement based on information or constraints beyond the heuristics used internally by a particular automated agent. A GUI specification-editor has been suggested as a meaningful mode of user (job requester) intervention.

```
spec1 : resource specification
spec2 : resource specification
lower (spec1)   -->   spec2

spec : resource specification
job : job contact information (or error status)
request (spec)   -->   job

lowering example:
lower ( (count=5) )   -->
(+ (& (count=3) (resourceManagerContact=RM1 ))
(& (count=2) (resourceManagerContact=RM2 )))
```

DUROC implements the allocation operation across multiple RMs in the Globus testbed and leaves decisions to use lower level specifications to the higher-level tools.

Once a resource specification has been refined the agent must attempt to allocate resources. In general the resources might managed by different RMs, and the co-allocator must atomically schedule the user's single abstract job or fail to schedule the job. Because the GRAM interface does not provide support for inter-manager atomicity, the user code must be augmented to implement a job-start barrier; as distributed components of the job become active, they must rendezvous with the allocating agent to be sure all components were successfully started prior to performing any non-restartable user operations.

```
main :
job_start_barrier ( )
. . .
user_operations ( )
```

There are three important points regarding the job-start barrier in the user's code. First, atomicity of job creation can only guaranteed after the barrier, so the user should not perform operations which cannot be reversed, e.g. certain persistent effects or input/output operations, until after the barrier. Second, the barrier call is used to implement guaranteed job cancellation within each RM; if the agent's job scheduling fails but some of the components have been scheduled through a manager that cannot cancel jobs it schedules, the agent will have to rendezvous with those components when they become active and signal them to abort. Third, the barrier call initialises the job-aggregation communication functions needed to make use of the co-allocated resources.

DUROC shares its Resource Specification Language (RSL) with GRAM. DUROC can perform allocations described by a 'lowered' resource specification.

The task of the lowering agent is to take a resource request of some form, be it a generalised GRAM request or user inputs to a GUI interface, and produce a lowered request so that DUROC can directly acquire the resources for the user.

The allocation semantics for DUROC requests are that each component of the top-level multi-request represents one GRAM request that DUROC should make as part of the distributed job DUROC is allocating. In order to make the request, DUROC must be able to determine what RM to contact.

Typically there will be additional terms in the conjunctions of the lowered request, and those terms will be passed on verbatim in each GRAM request. DUROC will extract each component of the lowered multi-request, remove the DUROC-specific components of the sub-request, and then forward that sub-request to the specified GRAM. Therefore any other attributes supported by GRAM are implicitly supported by DUROC. For example:

```
+(&(resourceManagerContact=RM1)(count=3)(executable=myprog.sparc))
 (&(resourceManagerContact=RM2)(count=2)(executable=myprog.rs6000))
```

In this request the executables and node counts are specified for each resource pool. While GRAM may in fact require fields such as these, DUROC treats them as it would any other fields not needed to do its job – it forwards them in the sub-requests and it is up the RMs to either successfully handle the request or return a failure-code back to DUROC (which will then return an appropriate code to the user).

Requests submitted to the DUROC API are decomposed into the individual GRAM requests and each request is submitted through the GRAM API. A DUROC request proceeds with each GRAM request in the job that succeeds. Runtime features available to the job processes include a start barrier and inter-process communications to help co-ordinate the job processes.

The start barrier allows the processes to synchronise before performing any non-restartable operations. In the absence of a start barrier, there is no way to guarantee that all job components are successfully created prior to executing user code. The communications library provides two simple mechanisms to send start-up and bootstrapping information between processes: an inter-subjob mechanism to communicate between "node 0" of each subjob, and an intra-subjob mechanism to communicate between all the nodes of a single subjob. A library of common bootstrapping operations is provided, using the public inter-subjob and intra-subjob communication interfaces.

Application support: The C and Java APIs which are available in Globus allow to build the Grid-aware applications. For example, Cactus applications make use of these APIs. The APIs also allow the development of some extended Grid services such as resource brokers, bandwidth brokers, resource estimators, etc.

II.5.5 Communication methods

The Globus communications module is based on the Nexus communication library. Nexus defines five basic abstractions: nodes, contexts, threads, communication links, and remote service requests (see Figure 11). The Nexus functions that manipulate these abstractions constitute the Globus communication interface. This interface is used extensively by other Globus modules and has also been used to construct various higher-level services, including parallel programming tools. The Active Messages and Fast Messages systems have similarities in goals and approach, but there are also significant differences.

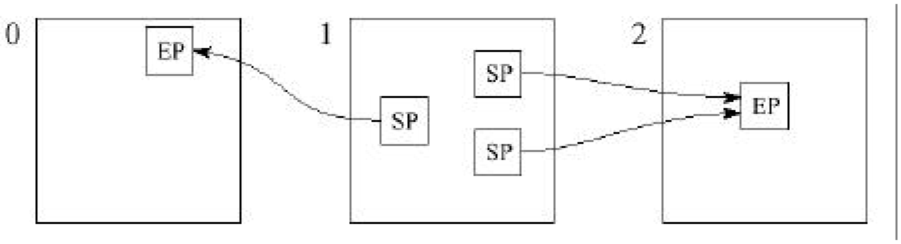


Figure 11: Nexus communication mechanisms (SP – startpoint, EP – endpoint)

Nexus programs bind communication startpoints and endpoints to form communication links. If multiple startpoints are bound to an endpoint, incoming communications are interleaved, in the same manner as messages sent to the same node in a message passing system. If a startpoint is bound to multiple endpoints, communication results in a multicast operation. A startpoint can be copied between processors, causing new communication links to be created that mirror the links associated with the original startpoint. This support for copying means that startpoints can be used as global names for objects. These names can be communicated and used anywhere in a distributed system. A communication link supports a single communication operation: an asynchronous remote service request (RSR). An RSR is applied to a startpoint by providing a procedure name and a data buffer. For each endpoint linked to the startpoint, the RSR transfers the data buffer to the address space in which the endpoint is located and remotely invokes the specified procedure, passing the endpoint and the data buffer as arguments. A local address can be associated with an endpoint, in which case startpoints associated with the endpoint can be thought of as "global pointers" to that address.

The Nexus interface and implementation support rule-based selection of the methods – such as protocol, compression method, and quality of service – used to perform communication. Dfferent communication methods can be associated with different communication links, with selection rules determining which method should be used when a new link is established. These mechanisms have been used to support multiple communication protocols and selective use of secure communication in heterogeneous environments.

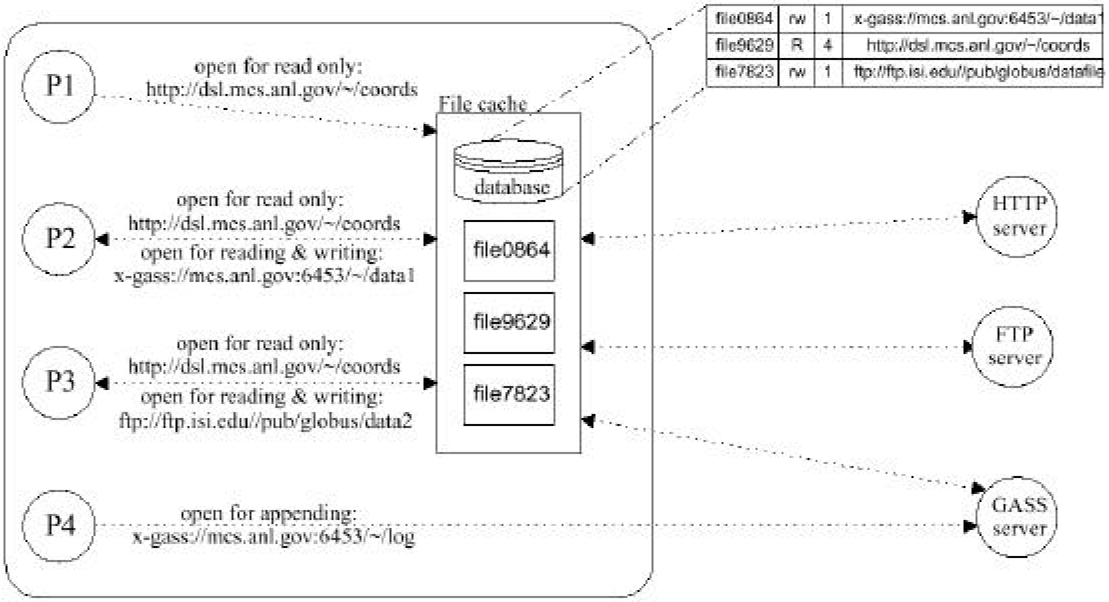
Fault tolerance: In addition to some fault tolerance integral to Nexus (communication library), Globus provides the Globus Heartbeat Monitor (HBM). The HBM performs periodic inquiry to detect process failures. The HBM does not provide system information, only process information. It additionally only provides an answer to the question "is process X still active?"

To use the HBM, the user process or heartbeat client (HBC) makes a function call to register with the HBM daemon running on the local machine. This registration notifies the daemon to actively probe the process, specifies the probing frequency, and specifies the data collector that receives probe data. In this manner a process' health can be remotely monitored.

Services supporting collaboration and remote instrument control: Globus allows for resource and data sharing. It's used in many collaborative environments as the basis of the security set-up as well as for data access. New on-going developments within different projects world-wide will allow these mechanisms to be made more transparent to a user.

II.5.6 Data management and integrity

Data access: Access to remote files is provided by the Global Access to Secondary Storage (GASS) subsystem. This system allows programs that use the C standard I/O library to open and subsequently read and write files located on remote computers, without requiring changes to the code used to perform the reading and writing. As illustrated in Figure 12, files opened for reading are copied to a local file cache when they are opened, hence permitting subsequent read operations to proceed without communication and also avoiding repeated fetches of the same file. Reference counting is used to determine when files can be deleted from the cache. Similarly, files opened for writing are created locally and copied to their destination only when they are closed. A similar copying strategy is used in UFO, but the Globus implementation does not rely on the Unix-specific proc file system. GASS also allows files to be opened for remote appending, in which case data is communicated to the remote file as soon as it is written; this mode is useful for log files, for example. In addition, GASS supports remote operations on caches and hence, for example, program-directed pre-staging and migration of data. HTTP, FTP, and specialised GASS servers are supported.



file0864	rw	1	x-gass://mcs.anl.gov:6453/~/data1
file9629	R	4	http://dsl.mcs.anl.gov/~/coords
file7823	rw	1	ftp://ftp.isi.edu//pub/globus/datafile

P1
open for read only:
http://dsl.mcs.anl.gov/~/coords
File cache
database
P2
open for read only:
http://dsl.mcs.anl.gov/~/coords
open for reading & writing:
x-gass://mcs.anl.gov:6453/~/data1
file0864
file9629
file7823
P3
open for read only:
http://dsl.mcs.anl.gov/~/coords
open for reading & writing:
ftp://ftp.isi.edu//pub/globus/data2
P4
open for appending:
x-gass://mcs.anl.gov:6453/~/log
HTTP server
FTP server
GASS server

Figure 12: the Global Access to Secondary Storage (GASS).

The Global Access to Secondary Storage (GASS) subsystem allows processes on local computers to read and write remote files. Copies of remote files opened for reading and/or writing are maintained in a local file cache. A simple database keeps track of the local file name, access mode, reference count, and remote file URL.

Data migration and replication: Globus Replica Management architecture is responsible for managing complete and partial copies of data sets. Replica management is an important issue for a number of scientific applications. For example, consider a data set that contains petabytes of experimental results for a particle physics ap plication. While the complete data set may exist in one or possibly several physical locations, it is likely that many universities, research laboratories or individual researchers will have insufficient storage to hold a complete copy. Instead, they will store copies of the most relevant portions of the data set on local storage for faster access. Services provided by a replica management system include:

- creating new copies of a complete, or partial data set;
- registering these new copies in a *Replica Catalog;*
- allowing users and applications to query the catalog to find all existing copies of a particular file or collection of files;
- selecting the "best'' replica for access based on storage and network performance predictions provided by a Grid information service;

The Globus replica management architecture is a layered architecture. At the lowest level is a *Replica Catalog* that allows users to register files as logical collections and provides mappings between logical names for files and collections and the storage system locations of one or more replicas of these objects. So far a *Replica Catalog API* in C as well as a command-line tool are implemented; these functions and commands perform low-level manipulation operations for the replica catalog, including creating, deleting and modifying catalog entries. After that a higher-level *Replica Management API* is defined which creates and deletes replicas on storage systems and invokes low-level commands to update the corresponding entries in the replica catalog.

The basic replica management services that are provided can be used by higher-level tools to *select among replicas* based on network or storage system performance or *automatically to create new replicas* at desirable locations. Some higher-level services will be implemented in the next generation of replica management infrastructure.

As mentioned above, the purpose of the replica catalog is to provide mappings between logical names for files or collections and one or more copies of the objects on physical storage systems. The catalog registers three types of *entry*: logical collections, locations and logical files.

A *logical collection* is a user-defined group of files. For a user it is convenient and intuitive to register and manipulate groups of files as a collection, rather than requiring that every file be registered and manipulated individually. Aggregating files should reduce both the number of entries in the catalog and the number of catalog manipulation operations required to manage replicas.

Location entries in the replica catalog contain all the information required for mapping a logical collection to a particular physical instance of that collection. The location entry may register information about the physical storage system, such as the hostname, port and protocol. In addition, it contains all information needed to construct a URL that can be used to access particular files in the collection on the corresponding storage system. Each location entry represents a complete or partial copy of a logical collection on a storage system. One location entry corresponds to exactly one physical storage system location. The location entry explicitly lists all files from the logical collection that are stored on the specified physical storage system.

Each logical collection may have an arbitrary number of associated location entries, each of which contains a (possibly overlapping) subset of the files in the collection. Using multiple location entries, users can easily register logical collections that span multiple physical storage systems. Despite the benefits of registering and manipulating collections of files using logical collection and location objects, users and applications may also want to characterise individual files. For this purpose, the replica catalog includes optional entries that describe individual *logical file*s. Logical files are entities with globally unique names that may have one or more physical instances. The catalog may optionally contain one logical file entry in the replica catalog for each logical file in a collection.

Data storage: The Storage Resource Broker, which actually is not a part of Globus (but has been developed to operate in the Globus environment) can be used to manage the data storage.

Data transfer: GridFTP is a universal Grid data transfer and access protocol that provides secure, efficient data movement in Grid environments. This protocol, which extends the standard FTP protocol, provides a superset of the features offered by the various Grid storage systems currently in use. Using GridFTP as a common data access protocol would be mutually advantageous to Grid storage providers and users. Storage providers would gain a broader user base, because their data would be available to any client, while storage users would gain access to a broader range of storage systems and data.

The FTP protocol was chosen for extension because it is the most commonly used protocol for data transfer on the Internet and the most likely candidate for meeting the Grid's needs.

The FTP protocol is an attractive choice for several reasons. First, FTP is a widely implemented and well-understood IETF standard protocol. As a result, there is a large base of code and expertise from which to build. Second, the FTP protocol provides a well-defined architecture for protocol extensions and supports dynamic discovery of the extensions supported by a particular implementation. Third, numerous groups have added extensions through the IETF, and some of these extensions will be particularly useful in the Grid. Finally, in addition to client/server transfers, the FTP protocol also supports transfers directly between two servers, mediated by a third party client (i.e. "third party transfer").

Features of GridFTP:

- GridFTP supports Grid Security Infrastructure (GSI) and Kerberos authentication, with user controlled setting of various levels of data integrity and/or confidentiality. GridFTP provides this capability by implementing the GSSAPI authentication mechanisms defined by RFC 2228, "FTP Security Extensions".

- To manage large data sets for distributed communities authenticated *third-party* control of data transfers between storage servers must be provided. A third-party operation allows a "third-party" user or application at one site to initiate, monitor and control a data transfer operation between two other "parties": the source and destination sites for the data transfer. Globus implementation adds GSSAPI security to the existing third-party transfer capability defined in the FTP standard. The "third-party" authenticates itself on a local machine, and GSSAPI operations authenticate the third party to the source and destination machines for the data transfer.

- On wide-area links, using multiple TCP streams in parallel (even between the same source and destination) can improve aggregate bandwidth over using a single TCP stream. GridFTP supports parallel data transfer through FTP command extensions and data channel extensions.

- Data may be striped or interleaved across multiple servers, as in a DPSS network disk cache or a striped file system. GridFTP includes extensions that initiate striped transfers, which use multiple TCP streams to transfer data that is partitioned among multiple servers. Striped transfers provide further bandwidth improvements over those achieved with parallel transfers. GridFTP protocol extensions have been defined that support striped data transfers

- Many applications would benefit from transferring portions of files rather than complete files. This is particularly important for applications like high-energy physics analysis that require access to relatively small subsets of massive, object-oriented physics database files. The standard FTP protocol requires applications to transfer entire files, or the remainder of a file starting at a particular offset. GridFTP introduces new FTP commands to support transfers of subsets or regions of a file.

- Using optimal settings for TCP buffer/window sizes can have a dramatic impact on data transfer performance. However, manually setting TCP buffer/window sizes is an error-prone process (particularly for non-experts) and is often simply not done. GridFTP extends the standard FTP command set and data channel protocol to support both manual setting and automatic negotiation of TCP buffer sizes for large files and for large sets of small files.

Reliable transfer is important for many applications that manage data. Fault recovery methods for handling transient network failures, server outages, etc. are needed. The FTP standard includes basic features for restarting failed transfers that are not widely implemented. The GridFTP protocol exploits these features and extends them to cover the new data channel protocol.

II.5.7 Security

The Globus Toolkit uses the Grid Security Infrastructure (GSI) for enabling secure authentication and communication over an open network. GSI provides a number of useful services for Grids, including mutual authentication and single sign-on.

The primary motivations behind the GSI are:

- The need for secure communication (authenticated and perhaps confidential) between elements of a computational Grid.
- The need to support security across organisational boundaries, thus prohibiting a centrally-managed security system.
- The need to support "single sign-on" for users of the Grid, including delegation of credentials for computations that involve multiple resources and/or sites.

GSI is based on public key encryption, X.509 certificates, and the Secure Sockets Layer (SSL) communication protocol. Extensions to these standards have been added for single sign-on and delegation. The Globus Toolkit's implementation of the GSI adheres to the Generic Security Service API (GSS-API), which is a standard API for security systems promoted by the Internet Engineering Task Force (IETF).

Authentication mechanism: A central concept in GSI authentication is the certificate. Every user and service on the Grid is identified by a certificate, which contains information vital to identifying and authenticating the user or service.

A GSI certificate includes four primary pieces of information:

- A subject name, which identifies the person or object that the certificate represents.
- The public key belonging to the subject.
- The identity of a Certificate Authority (CA) that has signed the certificate to certify that the public key and the identity both belong to the subject.

- The digital signature of the named CA.

Note that a third party (a CA) is used to certify the link between the public key and the subject in the certificate. In order to trust the certificate and its contents, the CA's certificate must be trusted. The link between the CA and its certificate must be established via some non-cryptographic means, or else the system is not trustworthy.

GSI certificates are encoded in the X.509 certificate format, a standard data format for certificates established by the Internet Engineering Task Force (IETF). These certificates can be shared with other public key-based software, including commercial web browsers from Microsoft and Netscape.

If two parties have certificates, and if both parties trust the CAs that signed each other's certificates, then the two parties can prove to each other that they are who they say they are. This is known as mutual authentication. The GSI uses the Secure Sockets Layer (SSL) for its mutual authentication protocol, which is described below. (SSL is also known by a new, IETF standard name: Transport Layer Security, or TLS.)

Before mutual authentication can occur, the parties involved must first trust the CAs that signed each other's certificates. In practice, this means that they must have copies of the CAs' certificates – which contain the CAs' public keys – and that they must trust that these certificates really belong to the CAs.

To mutually authenticate, the first person (A) establishes a connection to the second person (B). To start the authentication process, A gives B his certificate. The certificate tells B who A is claiming to be (the identity), what A's public key is, and which CA is being used to verify the certificate. B will first make sure that the certificate is valid by checking the CA's digital signature to make sure that the CA actually signed the certificate and that the certificate hasn't been tampered with. (This is where B must trust the CA that signed A's certificate.)

Once B has checked out A's certificate, B must make sure that A really is the person identified in the certificate. B generates a random message and sends it to A, asking A to encrypt it. A encrypts the message using his private key, and sends it back to B. B decrypts the message using A's public key. If this results in the original random message, then B knows that A is who he says he is.

Now that B trusts A's identity, the same operation must happen in reverse. B sends to A her certificate, A validates the certificate and sends a challenge message to be encrypted. B encrypts the message and sends it back to A, and A decrypts it and compares it with the original. If it matches, then A knows that B is who she says she is.

At this point, A and B have established a connection to each other and are certain that they know each others' identities.

By default, the GSI does not establish confidential (encrypted) communication between parties. Once mutual authentication is performed, the GSI gets out of the way so that communication can occur without the overhead of constant encryption and decryption.

The GSI can easily be used to establish a shared key for encryption if confidential communication is desired. Recently relaxed United States export laws now allow us to include encrypted communication as a standard optional feature of the GSI.

A related security feature is communication integrity. Integrity means that an eavesdropper may be able to read communication between two parties but is not able to modify the communication in any way. The GSI provides communication integrity by default. (It can be turned off if desired). Communication integrity introduces some overhead in communication, but not as large an overhead as encryption.

The core GSI software provided by the Globus Toolkit expects the user's private key to be stored in a file in the local computer's storage. To prevent other users of the computer from stealing the private key, the file that contains the key is encrypted via a password (also known as a pass phrase). To use the GSI, the user must enter the pass phrase required to decrypt the file containing their private key.

The use of cryptographic smartcards in conjunction with the GSI has also been prototyped. This allows users to store their private key on a smartcard rather than in a filesystem, making it still more difficult for others to gain access to the key.

The GSI provides a delegation capability: an extension of the standard SSL protocol which reduces the number of times the user must enter his pass phrase. If a Grid computation requires that several Grid resources be used (each requiring mutual authentication), or if there is a need to have agents (local or remote) requesting services on behalf of a user, the need to re-enter the user's pass phrase can be avoided by creating a proxy.

A proxy consists of a new certificate (with a new public key in it) and a new private key. The new certificate contains the owner's identity, modified slightly to indicate that it is a proxy. The new certificate is signed by the owner, rather than a CA. The certificate also includes a time notation after which the proxy should no longer be accepted by others. Proxies have limited lifetimes.

The proxy's private key must be kept secure, but because the proxy isn't valid for very long, it doesn't have to kept quite as secure as the owner's private key. It is thus possible to store the proxy's private key in a local storage system without being encrypted, as long as the permissions on the file are set to prevent anyone else from looking at it easily. Once a proxy is created and stored, the user can use the proxy certificate and private key for mutual authentication without entering a password.

When proxies are used, the mutual authentication process differs slightly. The remote party receives not only the proxy's certificate (signed by the owner), but also the owner's certificate. During mutual authentication, the owner's public key (obtained from her certificate) is used to validate the signature on the proxy certificate. The CA's public key is then used to validate the signature on the owner's certificate. This establishes a chain of trust from the CA to the proxy through the owner.

Authorisation mechanisms: For users to be authorised to use GSI-enable services they need to be entered into the GSI access control list. In Globus this information is stored in a file: *Grid-mapfile*. The purpose of this file is to map a GSI Credential to a local user's login name. The GSI administrator on the site can map the holder of any GSI credential to any local user name. It is up to the GSI administrator to verify that the GSI Identity is owned by and matches the local username.

The Grid-mapfile file is a plain text file, containing a quoted GSI Credential Name (the subject of an X509 certificate) and an unquoted local user name.

Each subject name in the Grid-mapfile must be listed only once. However, multiple identities may map to a shared local name. It is up to the GSI administrator to ensure that the Grid-mapfile entries do not violate any site security policies.

Accounting: There are no accounting mechanisms in Globus.

Credential delegation: Supported.

II.5.8 References

1. Home page: http://www.globus.org/
2. Globus Toolkit 1.1.3 System Administration Guide *December 2000*
3. Globus Tutorials
4. Globus papers: http://www.globus.org/research/papers.html

II.6 UNICORE (Uniform Interface To Computer Resources)

II.6.1 Introduction

The German Ministry for Education and Research approved and funded a project to develop a software infrastructure for seamless access to distributed supercomputer resources. The UNICOR project ran during the two and a half year period 01.08.1997 to 31.12.1999, and was a cooperation between the following partners ZAM - Forschungszentrum Jülich GmbH, DWD - Deutscher Wetterdienst, Offenbach, RUS - Rechenzentrum der Universität Stuttgart, Genias Software GmbH, Regensburg, Pallas GmbH, Brühl. The goal of UNICORE was to deliver a Web-based system that allows users to submit jobs to remote high performance computing resources without having to learn details of the target operating system, data storage conventions and techniques, or administrative policies and procedures at the target site. This mechanism should allow for job preparation, submission and monitoring. The UNICORE prototype developed runs in test-mode for the target systems Cray T3E, IBM SP2, Nec SX-4 and Fujitsu VPP 700. It also works with batch subsystems NQS and LoadLeveler.

The UNICORE Plus project is a continuation of the finished UNICORE project and it will run during the period 01.01.2000 and 31.12.2002. The goal of UNICORE Plus is to develop a Grid infrastructure. This has to be done with strong authentication in a uniform and easy to use way. Research areas in UNICORE Plus are resource modelling, application specific interfaces, data management, job control flow, and metacomputing. The project goals are to develop an improved, robust prototype which allows for multi-part, multi-site jobs with elaborate job flow functions and which support ap plication specific interfaces, efficient data transfer, resource modelling, and meta-computing at application level. The existing UNICORE prototype runs in test-mode at the partner sites for the target systems Cray T3E, T90, and J90, Fujitsu VPP 700, Hitachi SR 8000, IBM SP2, and Siemens hpcLine with the following resource managers: NQS, Load Leveler and CCS.

II.6.2 Application description

Because of constraints due to the http-based web architecture, the user can access the resources only in batch mode. An interactive mode will be developed within EUROGRID project and will be merged into a UNICORE system at the end of both projects. The current UNICORE does not support synchronous parallel applications among separate platforms. The recursive job model allows the specification of dependencies between the elements to reflect the necessary synchronisation. The job group is a frame with general information for the part of job, and it can contain other job groups, jobs, and dependencies.

The UNICORE Client interface consists of two components: JPA (the Job Preparation Agent) and JMC (the Job Monitor Component). A uniform user interface allows end-users to access Grid resources in a seamless way. Access to UNICORE is possible only for users using browsers with secure https protocols and those who have a valid X 509 - compliant UNICORE certificate. The user is authenticated by the Gateway when presenting his or her certificate. This certificate allows access to all resources with single UNICORE user id. Both the JPA and he JMC are accessible from the same user application.

The JPA is written as a Java Applet running within the user's Web browser. Based on experience gained during the UNICORE project, the new client is written as Java application. The new client application should have higher stability, portability, and performance. A UNICORE job contains a number of independent tasks. The interface allows the preparation of script tasks, compile-link-run tasks and job groups. The script task is used to submit existing job scripts via UNICORE. The compile-link-run task is used to prepare new applications. The job group is used to build sub-jobs for other target systems. Script task and compile link tasks are prepared using different input panels, which allow the user to specify his needs. The JPA knows from the resource pages which libraries and application packages are available, and could by translated by the NJS at the target system. The JPA helps the client to create the complex job consisting of the tasks that he or she may execute on multiple systems. The mechanism developed in the first UNICORE project allows writing a script, specifying data transfer between different job groups and specifying sub-jobs for other target systems. New tasks can be based on previously created tasks. This interface allows the user to specify temporal and data dependencies between tasks, and to specify dependencies between tasks.

When the job definition is syntactically correct, the client component (JPA) creates the AJO (the Abstract Job Object). This AJO can then be submitted to a UNICORE gateway. Additionally the client checks the definition of the job and prevents the submission of incorrect jobs. The JPA checks the consistency of the job and checks if the specified resource limits are correct. The AJO is sent to the Gateway. UNICORE Plus extends the static resource model to a flexible and extensible resource model, which allows the addition of new resource categories in a non-disruptive way, which supports dynamic resource information, such as system load. The interface will enable UNICORE to exploit resource brokers from the EUROGRID project.

The AJO (the Abstract Job Object) is the main component in the architecture. The AJO is implemented as a Java class library and is designed to specify the operations on target systems. The AJO architecture is composed of open interfaces and can easily be extended. Because of its object-oriented structure and syntax, it should support the specification of various hardware, software, and site-specific object rules. The AJO is used as the interface specification between the JPA and the NJS, which are being independently developed. The UNICORE job consists of jobs or tasks, which are to be translated into batch jobs before running on the target system. The client creates the actions, which are represented in a system independent way where it places specification of requests for computational and data resources. Before jobs are executed, these abstract descriptions are translated into system specific actions. The descriptions also describe the dependencies between the elements which reflect the necessary synchronisation. The synchronisation is made based on a directed acyclic dependency graph.

To support a broad range of application in the UNICORE Plus project the AJO is enhanced to handle extended control tasks. To let the user formulate extended control of tasks, the graphical interface will support loops, if-then-else constructs, case statements, and specialised error handling.

The JMC allows the user to monitor the progress of submitted jobs. The UNICORE 3.1 system allows refreshing, fetching output, display of the job log, and of the job dependencies and resources, and job deletion and abortion. The next version will be extended to include hold job and resume job features. The icons indicate the status of the job groups and tasks. The icon may indicate that a job has completed successfully, has failed, is executing, or is queued. For each component job the user may also terminate executing jobs and remove or hold queued jobs. As within the JPA, selecting an icon representing a job results in displaying the dependency graph of the job at the selected level, for script and compile-link-run jobs the standard output and error files will be displayed.

II.6.3 Resource description

Each UNICORE job as well as each job group contains general information for the job: the job name, the target system, the account id and the user's e-mail address. The target system can be selected from all available Vsites (virtual site). The user account id is currently not used but it can be when it becomes necessary for a UNICORE user to have different accounts. The information about available resources is kept in the Resource Pages and JPA can be asked to display it.

The UNICORE Plus project extends resource description with a flexible and extensible model. This model allows the addition of new resource categories in a non-disruptive way and which supports dynamic resource information, such as system load, which can be used as a decision criterion for target system selection. The new resource model will be based on the XML language.

II.6.4 Contract specification methods

Using the Job Preparation Agent the user specifies general resource description needed by the job. The resource input panel allows the specification of per job resource requirements such as CPU time, number of processors or CPUs, the amount of main memory, and the job disk space requirements. From the resource information received from the NJS system, the JPA knows about the maximum values for the selected target system and will not allow a user to specify a value higher than that.

Defining compile-link-run tasks, a user specifies the needed compilers, special libraries and compiler switches. Again the JPA using information received from NJS resource database knows about the available libraries, application packages ands their properties.

II.6.5 Resource management

Grid information services: There is no information about Grid information services.

Brokerage services: UNICORE Plus does not have brokerage services. UNICORE supports neither the brokering of user resource requests to the most available or most suitable platform, nor automatic load levelling among sites or platforms. The availability of resources at candidate sites is made visible to the user by the UNICORE interface. The user selects specific target platforms based on the availability – at job preparation time – of required resources such as computational capacity, data files, software licenses, and other site-dependent idiosyncrasies to which the user may be accustomed. If the target site offers multiple machines with common characteristics, the target platform can be selected as a class of required resources, allowing the most appropriate platforms to be selected by the NJS. The UNICORE Plus project will have an interface to exploit resource brokers from EUROGRID projects.

Advanced reservation: There is no information about advanced reservation.

Scheduling mechanisms: Because UNICORE connects independently run centres and recognises site-specific administrative policies and practices, the administrators of UNICORE site can maintain controlling of UNICORE scheduling policies.

The NJS (Network Job Supervisor) controls one Vsite (Virtual Site). The Vsite is a single system or single cluster of systems that share the same Userids and filespace. On each Vsite there is a running batch subsystem. This service fulfils the broad range of services. It main task is to analyse the AJO to extract local and remote jobs. The local jobs are translated into real batch jobs for the target system. It also maps the UNICORE user id to the local Userid for the target system and provides local resource information to the gateway and job status information and job output.

Because most of the batch subsystems do not support features like advanced reservation, the UNICORE Plus project is undertaking research in scheduling of MPP-applications to be run in parallel at different sites.

Co-scheduling: There is no information about co-scheduling.

Application support: The project supports applications in a variety of ways. UNICORE allows the creation of a completely new client for each potential application. A mechanism is being developed which will allow the generation of a graphical interface for each potential application. Building a completely new interface is probably too difficult for the average user. The simplest way for integrating an application should implement a mechanism to integrate existing application’s GUI within UNICORE. For selected application like Fluent and StarCD they will try to write a ‘wizard’ that will guide the user through the process of specifying the relevant parameters for a simulation. These interfaces will guide users through the configuration for their application in a step-by-step manner, and will use the UNICORE mechanisms to seamlessly submit and control the jobs. The interfaces are integrated into the UNICORE user client so that e.g. a NASTRAN job can be one job step in a larger UNICORE job. The finial project result will be a toolkit the easy integration of existing and new applications into UNICORE.

II.6.6 Communication methods

UNICORE does not support synchronous message passing, but the user can use the properties of the AJO architecture to define the independent tasks within the AJO. Each task can be executed simultaneously on different machines. The independent child AJOs could also be executed simultaneously at different sites.

Communication models: The architecture used in the UNICORE Plus project is based on the three-tier architecture developed in the UNICORE project. It consists of a UNICORE GUI used by the users, UNICORE Server placed in each UNICORE Site, and at the bottom level, the target cluster/execution systems tier. User and server tiers are Java applications and the system tier is implemented in the Perl language. The UNICORE client application enables the user to get access to UNICORE from any workstation or PC on the Internet. The client connects to the gateway for authentication before contacting to the execution systems.

Communication protocols: The user interface and the Gateway use the SSL protocol to communicate with each other over the public network. The UNICORE Protocol Layer is used by UNICORE to send the AJO to the Gateway, which hands it over to the NJS in the other UNICORE site.

Quality of Service: There is no information about Grid information services.

Fault tolerance: Within the UNICORE Plus project, techniques for distributed execution of application are being researched, including a fault tolerant, batch oriented startup procedure.

Services supporting collaboration and remote instrument control: There is no information about such services.

Application support: There is no information about application support.

II.6.7 Data management and integrity

Data management is one of the most important issues in the UNICORE Plus project. Current research includes the efficient transfer of information between UNICORE sites, especially those situations where large amounts of data need to be sent to a site for job processing.

Data access: Each AJO executes within a single, temporary UNICORE filespace (Uspace), set up by the NJS. Before the computational process is executed, the data is transferred to the computational system, and the process is never forced to block the computational resources. This mechanism guarantees that there will be no need to support asystem for accessing remote files at run time. The only files accessible to an AJO outside this temporary directory are files automatically mapped by the NJS (e.g. the MPI library, the FORTRAN compiler, etc.) and files named in an Import Task. When the AJO terminates, the Export Task exports the stdout, stderr, and log files and the files specified by the user. Finally the Uspace is deleted by the NJS.

In the UNICORE Plus project, access to data archive will be integrated with the system.

Data migration: There is no information about data migration.

Replication: There is no information about replication.

Data storage: One of the UNICORE Plus project goals is to integrate access to the data archives. This access will use specialised servers like HPSS or ADSM, to move data between different sites. The communication between these servers will be based on UNICORE security model.

Data transfer: Through the graphical interface (JPA) the user can prepare file transfer jobs. In UNICORE Plus the data transfer will be independent from UNICORE jobs but will use UNICORE security mechanisms.

Meta-data management: There is no information about meta data management.

Application support: There is no information about data management application support.

II.6.8 Security

The key objective of UNICORE is to co-exist with existing administration and security polices. UNICORE is based on X.509V3 certificates. Because the users id and group is mapped onto his or her certificate on each UNICORE site by the NJS, there is no need to use Grid-wide user identification. The graphical user interface offers the function to set up and maintain the user's security environment.

Authentication mechanism: UNICORE security is based on the Secure Socket Layer (SSL) protocol and X.509V3 type certificates. Additionally each user and each NJS has its own X.509 certificate that it uses to communicate with the Gateway via SSL protocol. The UNICORE Public Key Infrastructure (PKI) is based on the regulations defined by DFN-PCA (German Research Network – Policy Certification Authority). The UNICORE Certification Authority (CA) is localised at LRZ (Leibniz-Rechenzentrum, München) that is one of the collaborators. All partners' centres run a Registration Authority (RA). The UNICORE server can be protected by a firewall. The firewall can be placed either completely behind the site's firewall or partially hidden which means that the firewall is located between Gateway and NJS.

The SSL protocol is using for communication by the UNICORE GUI - the Job Preparation Agent and the Job Monitor Controller). For authentication by the gateway each user uses his or her UNICORE X509 certificate for identification. Also each NJS has it's own X.509 certificate which is used by the Gateway for authentication. The user interface application maintains the certificates and it needs to know about the Certification Authority which signs the user and the gateway certificates. A secure connection can be establishing based on certificates from both components. Because the AJO can send to the other UNICORE site, it must also include the user certificate for authorisation at the other site. This feature allows the recognition of such attempts by the AJO.

The Gateway and the NJS always communicate via sockets and this communication also uses the SSL protocol. The UNICORE Protocol Layer (UPL) is used by NJS to sends the AJO to the Gateway. The

Gateway can take advantage of site-specific authentication methods like DCE (Distributed Computing Environment) or SecureID cards.

Authorisation mechanisms: Both the user interface and the Gateway participate in the user authentication. Only a user in possession of valid X509-compilant UNICORE cert ificate can download the Web page containing the JPA. Authenticated user gets access to the sets of the GUI components. The user defines the AJO and after that sends it to the Gateway that just hands it over to the NJS. The NJS receives the AJO, tests it and accepts it for processing. At this point, an acknowledgment is returned to the JPA, and the user can exit, or use the JPA to construct another job. The NJS first checks whether the received AJO belongs to the user who signed it. Because the AJO is both hierarchical and recursive and it can consist of many other AJOs, the AJO can be sending either locally to another NJS or to a gateway at another Unicore site. After authentication the NJS analyses the AJO for parts to be sent to other sites. It uses it's own X.509 certificate to communicate to a Gateway at another site. Because the Gateway knows that the AJO was received from the NJS, it uses the user's public key instead of the one from the sending NJS. The user's public key is used by the Gateway to unpack the AJO. Finally the NJS does the translation into batch jobs for the target system and sends them to the UNICORE batch subsystem interface (UBSSI) sometimes this procedure is called Target System Interface (TSI).

Accounting: Because each UNICORE site has it's own local user database there are two levels that retain full control over resources. Only users that are registered in the login database may submit the jobs. The second allows the specification of system dependent regulation and limits like user or project quotas for a particular machine.

Each of the systems represented as the Vsite is governed by a TSI. The TSI communicates with the local batch system to maintain the job directory, submit jobs on behalf of the user. This interface also controls the job status.

There is no information about global accounting policy.

Credential delegation: The NJS maps the user's certificate to the user id at the execution system which may be a cluster of nodes governed by one resource manager or a single stand-alone system.

Application support: There is no information about application support.

II.6.9 Glossary of Acronyms

- AJO – the Abstract Job Object
- CA - Certification Authority
- DCE - Distributed Computing Environment
- DFN - German Research Network
- JMC - the Job Moni tor Component
- JPA - the Job Preparation Agent
- NJS – the Network Job Supervisor
- PCS - Policy Certification Authority
- PKI - Public Key Infrastructure PKI
- RA - Registration Authority
- SSL - Secure Socket Layer

- TSI - Target System Interface (UBSSI is the same)
- UBSSI - UNICORE batch subsystem interface (TSI is the same)
- UPL - UNICORE Protocol Layer
- Uspace – UNICORE space

II.6.10 References

1. Home page: http://www.unicore.de/

II.7 Legion

II.7.1 Introduction

Legion is a distributed computing environment that combines very large numbers of independently administered machines into a single, coherent environment. Like a traditional operating system, Legion is built on a diverse set of lower-level resources to provide convenient user abstractions, services, and policy enforcement mechanisms. The difference is that in Legion, the lower-level resources may consist of thousands of heterogeneous processors, storage systems, databases, legacy codes, and user objects, all distributed over wide-area networks spanning multiple administrative domains. Legion provides the means to pull these scattered components together into a single, object-based *metacomputer* that accommodates high degrees of flexibility and site autonomy.

The authors of Legion had ten main goals while implementing the Legion project:

- *Site autonomy*. Legion is not a monolithic system. Legion is composed of diverse resources owned and controlled by a variety of different organisations. These organisations would like to have control over their own resources, e.g., specifying how much resource can be used, when it can be used, and who can and cannot use the resource. Therefore, an appropriate model for Legion is that of a cellular automaton - where the collective behaviour arises out of the behaviour of the individual components.

- *Extensible core*. Legion is an open system and can be tailored to the future needs of users. Mechanism and policy can be realised via extensible, replaceable, components. This permits Legion to evolve over time and will allow users to construct their own mechanisms and policies to meet their specific needs.

- *Scalable architecture*. Because Legion can consist of thousands of hosts, it uses a scalable software architecture. This implies that there are no centralised structures, and the system is totally distributed.

- *Easy-to-use, seamless computational environment*. Legion masks the complexity of the hardware environment and the complexity of communication and synchronisation of parallel processing. Machine boundaries are invisible to users. Therefore as far as possible, compilers acting in concert with run-time facilities, must manage the environment for the user.

- *High performance via parallelism*. Legion supports easy-to-use parallel processing with large degrees of parallelism. This includes task and data parallelism and their combinations. Because of the nature of the interconnection network, Legion must be latency tolerant. Furthermore, Legion must be capable of managing hundreds or thousands of processors. This implies that the underlying computation model and programming paradigms must be scalable.

- *Single, persistent namespace*. One of the most significant obstacles to wide area parallel processing is the lack of a single name space for file and data access. The existing multitude of disjoint name spaces makes writing applications that span sites extremely difficult. Therefore Legion uses single, persistent name space.

- *Security for users and resource owners*. Because Legion does not replace existing host operating systems, it cannot significantly strengthen existing operating system protection and security mechanisms. Strengthening operating system security is not a goal of Legion. However, Legion ensures that existing mechanisms are not weakened.

- *Manage and exploit resource heterogeneity*. Clearly Legion supports interoperability between heterogeneous components. In addition, Legion is able to exploit diverse hardware and data resources. Some architectures are better than others at executing particular kinds of code, e.g., vectorisable codes. These affinities, and the costs of exploiting them, must be factored into scheduling decisions and policies.

- *Multiple language support and interoperability*. Legion applications can be written in a variety of languages, and heterogeneous source language application components must be integrated. There is also a support for legacy codes.

- *Fault tolerance*. In a system as large as Legion, it is certain that at any given instant several hosts, communication links, and disks will have failed. Thus, dealing with failure and dynamic reconfiguration is a necessity. Current parallel processing systems do not address fault-tolerance. As is the case in the management of heterogeneity there is an extensive literature on reliable distributed computing systems. Two different fault tolerance problems can be considered, fault tolerance in Legion itself, and application fault-tolerance.

Legion authors are also aware of three main constraints:

- *Host operating systems cannot be replaced*. This restriction is required for two reasons. First, organisations will not permit their machines to be used if their operating systems must be replaced. Operating system replacement would require them to rewrite many of their applications, re-train many of their users, and possibly make a particular system incompatible with other systems in their organisation. It is sufficient to layer a system on top of an existing host operating system.
- *Changes to the interconnection network cannot be legislated*. Legion assumes that the network resources, and the protocols in use, are a given, and therefore it must accommodate operating system heterogeneity, and live with the available system. That is not to say though, that better protocols cannot be layered over existing protocols. Neither can it be stated that the performance of a particular application on a particular network will be poor unless the protocol is changed. In addition to the purely technical issues, there are also political, sociological, and economic ones. These include encouraging the participation of resource-rich centres and avoiding the human tendency to free-load. Legion intends to discourage such practices by developing and employing accounting policies that encourage good community behaviour.
- *Neither is it required that Legion run as "root".* To protect their resources, most users will want to run Legion with the fewest possible privileges.

Legion is structured as a system of distributed objects. All of the entities within Legion are represented by independent, active objects that communicate using a uniform remote method invocation service. In many ways, Legion's fundamental object model is similar to CORBA's: object interfaces are described using an interface description language (IDL), and are compiled and linked to implementations in a given language (e.g., C++, Java, Fortran). This approach enables component interoperability between multiple programming languages and heterogeneous execution platforms. Objects provide a clean, natural approach to the problems of encapsulation and interoperability: because all of the elements in the system are objects they can communicate with one another regardless of location, heterogeneity, or implementation details. Objects are the building blocks for constructing a wide-area OS.

II.7.2 Application description and Resource description

The abstraction, control, and management of underlying hardware resources are among the most fundamental services provided by any operating system. Because a Legion system runs on top of the unmodified operating system of each host in the net, it does not need to manage very low-level resources - the local OS does that job. At Legion's level, the resource base consists instead of multiple heterogeneous processors and storage devices.

Both processors and storage resources are represented as objects, called host objects and vault objects. There are two primary benefits resulting from this object-based approach. First, each object defines a uniform interface to host and vault resources in a Legion system. Host objects provide a uniform interface to object (task) creation, and vault objects provide a uniform storage allocation interface, even though there may be many different implementations of each of these. Second, these objects naturally act as resource guardians and policy makers. For example, the host objects used to manage the processor resources at a given site are the points of access control for task creation at that site. If an organisation participating in Legion wishes to restrict job creation on local resources exclusively to local users, the host objects at the organisation's site can enforce this policy.

This object-based model for resource representation allows a tremendous degree of extensibility and site autonomy. Applications (acting as resource clients) need only be aware of the generic object interfaces for the resources they require. Resource providers can provide desired implementations of the resource objects.

If the administrators of a local site wish to enforce a specialised access control policy for their processing resources, they can extend the implementation of the basic host object provided by Legion to incorporate the desired policy. If the owner of a disk wants to use local Unix-based disk-usage accounting and quota tools, he can use a vault object implementation that allocates storage under the appropriate local Unix userid for each client. Of course, Legion provides reasonably configurable default implementations of the basic resource objects; resource providers do not need to write any code to make their resources available to Legion. However, as new local resource usage policies become desirable, Legion explicitly supports such natural evolution.

It is important to note that resource interfaces are not carved in stone. If new interfaces for underlying resources are required, new classes of resource objects can be created to extend or replace existing interfaces. For example, a number of the processing resources in several deployed Legion networks require access through a local queue management system such as Codine or LoadLeveler. On such hosts, an extended queue-aware version of the Legion host object is used.

A Host Object is a host's representative to Legion. It is responsible for executing objects on the host, reaping objects, and reporting object exceptions. Thus, the Host Object for a host is ultimately responsible for deciding which objects can run on the host it represents. Since Host Objects can be implemented by the users who offer their resources to Legion, and since the Legion security model is one in which security is built into the object by its implementor, Legion users can select that policy and mechanism which restricts access to their own hosts.

II.7.3 Contract specification methods

In Legion, scheduling is a process in which services are negotiated for between two autonomous agents. One agent acts on behalf of the application (consumer) and one on behalf of the resource or system (provider). These negotiating agents can either be the principals themselves (objects or programs), or Schedulers and intermediaries acting on their own behalf. Scheduling in Legion is never of a dictatorial nature; requests are made of resource guardians, who have final authority over what requests are honoured.

Legion provides simple, generic default Schedulers that offer the classic "90%" solution - they do an adequate job, but can easily be outperformed by Schedulers with special knowledge of the application. Application writers can take advantage of the resource management infrastructure, described below, to write application-specific or application-type-specific user-level Schedulers. Legions resource management model allows user-defined Schedulers to interact with the infrastructure. The components of the model are the basic resources (hosts and vaults), the information database (the Collection), the schedule implementor (the Enactor), and an execution Monitor. The steps in object placement are as follows illustrated in Figure 13:

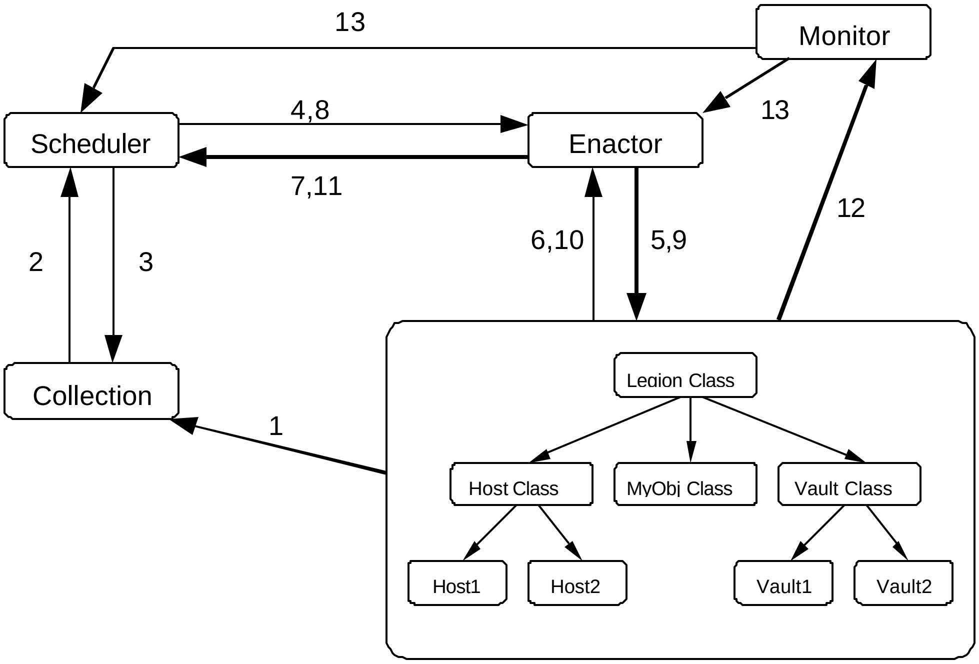


Figure 13: the Legion paradigm

(1) The Collection is populated with information describing the resources.

(2) The Scheduler queries the Collection, and based on the result and knowledge of the application, computes a mapping of objects to resources. This application-specific knowledge can either be implicit (in the case of an application-specific Scheduler), or can be acquired from the application's classes.

(3) This mapping is passed to the Enactor, which…

(4) … invokes methods on hosts and vaults to…

(5) … obtain reservations from the resources named in the mapping.

(6) After obtaining reservations, the Enactor consults with the Scheduler to confirm the schedule, and…

(7) … after receiving approval from the Scheduler,

(8) … attempts to instantiate the objects through member function calls on the appropriate class objects.

(9) The class objects report success/failure codes, and…

(10) … the Enactor returns the result to the Scheduler.

(11) If, during execution, a resource decides that the object needs to be migrated,…

(12) … it performs an outcall to a Monitor,…

… which notifies the Scheduler and Enactor that rescheduling should be performed.

II.7.4 Resource management

Grid information services: The Collection acts as a repository for information describing the state of the resources comprising the system. Each record is stored as a set of Legion object attributes. Collections provide methods to join (with an optional instalment of initial descriptive information) and update records, thus facilitating a push model for data. The security facilities of Legion authenticate the caller to be sure that it is allowed to update the data in the Collection. Collections may also pull data from resources. Users, or their agents, obtain information about resources by issuing queries to a Collection. A Collection query is string conforming to the grammar that allows typical operations (field matching, semantic comparisons, and boolean combinations of terms). Identifiers refer to attribute names within a particular record, and are of the form $AttributeName. For example, to find all hosts that run the IRIX operating system version 5.x, one could use the regular expression matching feature for strings, and query as follows:

```
match($host os name, \IRIX”) and match($host os name, \5n..*”)
```

In its current implementation, the Collection is a passive database of static information, queried by Schedulers. There are plans to extend Collections to support function injection, i.e., the ability for users to install code to dynamically compute new description information, and integrate it with the already existing description information for a resource. An important use of Collections is to structure resources within the Legion system. Having a few global Collections will reduce the scalability. Thus, Collections may receive data from, and send data to, other Collections. This allows the holding of a Collection for each administrative domain, and for Collections to be combined with other Collections.

Brokerage services: The Scheduler computes the mapping of objects to resources. At a minimum, the Scheduler knows how many instances of each class must be started. Application-specific Schedulers may implicitly have more extensive knowledge about the resource requirements of the individual objects, and any Scheduler may query the object classes to determine such information (e.g., the available implementations, or memory or communication requirements). The Scheduler obtains resource description information by querying the Collection, and then computes a mapping of object instances to resources. This mapping is passed on to the Enactor for implementation. It was not intended to directly develop more than a few widely-applicable Schedulers; that task has been left to experts in the field of designing scheduling algorithms. Schedules must be passed between Schedulers and Enactors. Each Schedule has at least one Master Schedule, and each Master Schedule may have a list of Variant Schedules associated with it. Both master and variant schedules contain a list of mappings, with each mapping having the type (Class LOID ! (Host LOID x vault LOID)). Each mapping indicates that an instance of the class should be started on the indicated (host, vault) pair. In the future, this mapping process may also select from among the available implementations of an object as well.

There are three important data types for interacting with the Enactor: the LegionScheduleFeedback type, the LegionScheduleList, and the LegionScheduleRequestList type.

A LegionScheduleList is simply a single schedule (e.g. a Master or Variant schedule). A LegionScheduleRequestList is the entire data structure. LegionScheduleFeedback is returned by the Enactor, and contains the original LegionScheduleRequestList and feedback information indicating whether the reservations were successfully made, and if so, which schedule succeeded.

Advanced reservation: There are three broad groups of functions: reservation management, object management, and information reporting. Reservation management functions make, check, and cancel reservation of resources. Object management functions start, kill, and deactivate objects.

The reservation functions are used by the Enactor to obtain a reservation token for each subpart of a schedule. When asked for a reservation, the Host is responsible for ensuring that the vault is reachable, that sufficient resources are available, and that its local placement policy permits instantiating the object.

In addition to the information reporting methods listed above, the Host also supports the attribute database included in all Legion objects. These information reporting methods for Host Objects allow the building of Collections using a pull model in which the Collection can query the host to determine its current state. All Legion objects include an extensible attribute database, the contents of which are determined by the type of the object. Host objects populate their attributes with information describing their current state, including architecture, operating system, load, available memory, etc. Future versions of host objects will export scheduling policy information so that user-level Schedulers can better determine whether particular hosts are good candidates for object placement.

The Host Object reassesses its local state periodically, and repopulates its attributes. If a push model is being used, it will then deposit information into its known Collection(s). The extensibility of the Legion object attribute databases allows the Host Object to export a rich set of information, well beyond the minimal “architecture, OS, and load average” information used by most current scheduling algorithms. For example, the Host could export information such as the amount charged per CPU cycle consumed, domains from which it refuses to accept object instantiation requests, or a description of its willingness to accept extra jobs based on the time of day. This kind of information can help Schedulers to make better choices at the outset, thus avoiding the computation of subtly unfeasible schedules.

The current implementation of Vault Objects does not allow dynamic states to the degree that Host Objects do. Vaults, therefore, only participate in the scheduling process at the start, when they verify that they are compatible with a host. They may, in the future, be differentiated by the amount of storage available, cost per byte, security policy, etc.

Scheduling mechanisms: Legion has a wide variety of resources. Resource classes include CPU cycles, disk and memory space, files, databases, communications channels and bandwidth. For many resources such as CPU’s, resource management is a scheduling problem. Scheduling policies and mechanisms must be developed for Legion that accommodate the heterogeneous and distributed nature of the machine. Because the general scheduling problem is NP-hard, the schedulers will be heuristic, and optimal schedules will not be realised. The objective of scheduling in Legion is to obtain reduced completion time. This is done is two ways: (1) exploit the amount of available CPU resources to increase opportunities for parallel execution and (2) exploit resource affinities within programs to choose the best machine for program *components*. Programs suitable for parallel execution in Legion may contain functional parallel and data parallel components. Scheduling functional parallel components provides an opportunity for exploiting resource affinities or satisfying implementation dependencies. Scheduling data parallel components also provides an opportunity for exploiting resource affinities such as communication topology affinity (e.g., a *1-D* or *2-D* application communication topology is well-suited to a mesh or torus), but the amount of available computational resources provides an even greater opportunity for data parallel components. Scheduling data parallel components is done to exploit communication affinities and the available computational resources.

Scheduling data parallel components in Legion is accomplished in three phases: *processor selection*, *load selection*, and *placement*. Processor selection determines a set of *candidate* processors from the available processors in Legion. Load selection chooses the number and type of processors to use from the candidate set and decomposes the data domain across the chosen processors. Placement maps tasks to processors such that communication time is reduced. Scheduling is performed at runtime when the available processors are known to the system. The scheduling method is a form of *co-scheduling* in that all selected processors are scheduled collectively. The method is also *static* in the sense that once a scheduling decision is made it is not changed. Completion time is determined by two factors: load balance and computation granularity. Computation granularity restricts the amount of parallelism that can be efficiently exploited. Load balance ensures that all processors will finish at the same time, a necessary condition for reduced completion time. The data domain is decomposed to achieve load balance.

An efficient method for automatically scheduling data parallel components across a LAN (i.e., department-wide Legion) has been developed. The method is a heuristic that chooses the best set of available processors and a decomposition of the data domain that produces the smallest completion time. The method uses information about the program components and the available resources, and a set of communication cost functions to make effective runtime scheduling decisions.

Co-scheduling: Co-scheduling in Legion is provided by the Enactor. A Scheduler first passes in the entire set of schedules to the make reservations() call, and waits for feedback. If all schedules fail, the Enactor may (but is not required to) report whether the failure was due to an inability to obtain resources, a malformed schedule, or other failure. If any schedule succeeds, the Scheduler can then use the enact schedule() call to request that the Enactor instantiates objects on the reserved resources, or the cancel reservations() method to release the resources.

The Enactor uses the master and variant schedules mentioned before. Each entry in the variant schedule is a single-object mapping, and replaces one entry in the master schedule. If all mappings in the master schedule succeed, then scheduling is complete. If not, then a Variant schedule is selected that contains a new entry for the failed mapping. This Variant may also have different mappings for other instances, which may have succeeded in the Master schedule. Implementing the Variant schedule entails making new reservations for items in the Variant schedule, and cancelling any corresponding reservations from the Master schedule. Default Schedulers and Enactor work together to structure the Variant schedules so as to avoid reservation thrashing (the cancelling and subsequent remaking of the same reservation).

Class objects implement a create_instance() method. This method has an optional argument containing an LOID and a reservation token. Use of the optional argument allows directed placement of objects, which is necessary to implement externally computed schedules. The Class object is still responsible for checking the placement for validity and conformance to local policy, but the Class does not have to go through the standard placement steps.

Application support: There is no information on application support (APIs).

II.7.5 Communication methods

Communication models: To enable inter-process communication, Legion supports a variation of remote method invocation designed to address the needs of wide-area applications. Wide-area systems communication can be costly, in terms of both latency and bandwidth. Applications in the wide-area operating system require effective tools for reducing inter-process communication, and for tolerating the high latencies involved. To address this issue, Legion supports a remote method invocation model known as macro-dataflow (MDF) in addition to (and built upon) a basic, low-level message-passing service. MDF is a synchronous remote method invocation protocol that enables multiple concurrent method invocations from a single client as well as the overlap of remote methods and local computation. Furthermore, MDF methods encode data-dependencies for remote method results. In MDF, a remote method caller need never receive the results of that method. If the results are needed only as parameters for other future method invocations, this fact is encoded in the method invocation protocol and the remote method implementation will forward the results directly to the objects that will handle the appropriate future invocations. Legion automates this protocol, enabling the client (typically through the use of a Legion-aware compiler such as MPLC) to specify complete program graphs of interdependent remote method invocations, and enabling objects to match incoming parameters into complete method invocations (including data dependencies).

Communication protocols: A method call from one Legion object to another can consist of multiple Legion messages. Because Legion supports dataflow-based method invocation, the various arguments of a method call may flow into the target as messages from several different objects. The messages themselves are "packetised" and transmitted using one of a number of underlying transport layers, including UDP/IP, TCP/IP, or platform-specific message passing services (e.g., user-level message passing over an IBM SP2 switch).

Communication among geographically distribut ed (and perhaps architecturally distinct) computers is accomplished by using a common suite of communications protocols. From the point of view of the application program, the most important of these are the protocols which implement the *network* and *transport* layers of the OSI reference model. The transport protocol traditionally provides end-to-end, reliable, sequential, non-duplicated delivery of data across an arbitrary subnet, while the network protocol accomplishes routing, route determination, fragmentation, and packet lifetime control. Two traditional protocols for these two tasks are the *Transmission Control Protocol (TCP)* and the *Internet Protocol (IP*). While TCP and IP are often considered to be a single unit, they are not; in fact, there are many transport protocols that run over IP and provide different types of end-to-end services.

For Legion, usage of the *Xpress Transfer Protocol (XTP)* as a transport protocol was proposed. XTP operates over IP and thus is usable in any computing device currently attached to the Internet. XTP was chosen instead of TCP because XTP provides all the functionality of TCP, plus additional services especially designed for high-throughput, low-latency communications. Examples of these additional services include: reliable datagrams (one-time messages that nevertheless must be successfully delivered); transactions (request/response queries); rich message priority mechanism (transmitters, receivers, and all interior routers are always operating on their most important messages); selective retransmission (when packets are lost, only the missing ones are retransmitted, unlike the go-back-n mechanism in TCP); multicast (every connection can be a multi-peer connection); and multicast group management (reliability is controlled by the application, and can vary from none to complete). In addition to file transfers and process-to-process communication, XTP also supports multimedia applications.

For instance, using a combination of XTP's modes, it is possible to establish distribution of a synchronised audio/video bitstream from a single transmitter to any number of receivers. For workstations equipped with the audio/video hardware, Legion can thus support a teleconferencing utility among co-operating end stations. While XTP has been implemented for various local area networks (e.g., Ethernet, token ring, FDDI) and for various operating systems (e.g., Unix, DOS, Windows, and some real-time kernels), global interoperability requires specifically operation over both the existing Internet and the burgeoning National Information Infrastructure. While the former has been largely accomplished, XTP was ported to operate over Asynchronous Transfer Mode (ATM) host adapters and ATM switches in anticipation of the latter. By supporting three prot ocol combinations underneath XTP (IP alone, ATM alone, and IP over ATM), it will be possible to support process-to-process messaging among applications located anywhere in the world.

Quality of Service: No support.

Fault tolerance: In a system as large a Legion, it is certain that at any given instant several hosts, communication links, and disks will have failed. Thus, dealing with failure and dynamic reconfiguration is a necessity. Two different fault tolerance problems can be considered, fault tolerance of Legion itself, and application fault-tolerance. There is a trade-off between performance and different levels of fault tolerance. It is not necessary or desirable to be able to withstand all forms of faults and continue execution as if nothing had happened. The limit of Legion concern is with fail-stop faults of hardware components, including both processors and the network.

Legion system components are fault tolerant to the extent necessary and possible without compromising the high-performance objectives. For example, the underlying message system guarantees the delivery of messages in the presence of lost, reordered, or duplicated packets, or less than complete network failure. Further, if a host fails, Legion reconfigures itself to remove that host from the current configuration, and reconfigures again to include the host when the host recovers. Similarly, Legion continues operation under network partition by treating the non-accessible hosts as dead. The difficulties often associated with partition merge, e.g., bringing divergent copies of a database to a consistent state, are avoided in Legion by ensuring that all Legion system owned databases do not require a globally consistent state, and by providing mechanisms for the lazy propagation of updates. Furthermore, Legion does not define the semantics of operation under partition and merge of user defined persistent objects. The philosophy on application fault tolerance is that an application should not pay for fault-tolerance it does not need. Therefore, applications are able to tune their fault tolerance requirements by specifying a level of fault tolerance, and a penalty they are willing to pay in terms of recovery time. Briefly the Legion implementation exploits the properties of objects and their realisation in the run-time. There are stateless and stateful objects and they are treated differently. Fault-tolerance for stateless objects can be provided by noting when a method begins execution and by treating the message system in a database-like fashion. When the invocation of a method begins the messages are read from the database without consuming them. Then, when the method completes, the arguments are consumed, and the results placed into the database in one atomic action. If the method fails during execution, either due to an internal failure, or due to hardware failure, the system reacts accordingly. In the case of an internal failure, e.g., memory fault, the user is notified and the arguments that caused the failure are available for use with a debugger. If a host has failed, then the method is restarted on another host using the same arguments. Stateful objects present a more difficult problem, and the search for the best way to specify and implement differing levels of fault-tolerance will be a research priority. The difficulty lies not with providing fault tolerance for a single object; that can be accomplished with checkpointing and transactions on the message system "database". The difficulty lies with the interaction of multiple objects and guaranteeing that they see a consistent view of the world.

Services supporting collaboration and remote instrument control: No support.

Application support: No support.

II.7.6 Data management and integrity

Data access: In Legion, all entities - files, processors, storage devices, networks, users, etc. - are represented by objects, so the object naming mechanism is of central importance. Legion objects are identified by a three-level naming scheme. At the lowest level each object is assigned an Object Address (OA), which contains a list of network addresses that can be used to pass messages to the object (an OA might contain an IP address and port number). But since Legion objects can migrate, OAs will vary over time. Furthermore, clients may not care about object locations. Therefore Legion defines an intermediate layer of location-independent names called Legion Object Identifiers, or LOIDs: unique, immutable identifiers that are assigned to objects on creation. Although higher level than OAs, LOIDs are binary, globally unique, variable-length identifiers, and do not constitute a convenient user-level naming scheme. To address this, the third level naming layer is a user-level, hierarchical directory service called context space, which allows arbitrary Unix-like string paths to be assigned to objects. As part of its naming mechanism Legion provides scalable replicated binding services that allow translation from higher-level names to lower-level names (i.e., context paths to LOIDs and LOIDs to OAs). Also, to reduce overall binding traffic, clients cache bindings in their own memory space.

The Legion naming mechanism effectively reduces the complexity of distributed application design by providing a single global name space for all entities within the system. A typical distributed environment supports separate name spaces for files, hosts, and processes, whereas Legion supports the same global name space for all of these entities and more. Furthermore, at the highest level (context space) this naming mechanism presents an extremely simple interface of Unix-style paths.

In traditional operating systems, persistent storage is typically managed in the form of a file system. Legion's use of persistent objects, coupled with the Legion global naming service, enables Legion to fully subsume the notion of a file system. Users are presented with familiar concepts of paths, directories, and universally accessible files, but Legion's "file system" is also populated with other arbitrary object types such as Host objects, Class Managers, and user application tasks. Legion's support of a generalised persistent object space in place of a traditional rigid file system provides the basis for an extensible file system service in which individual files are customised to better suit application requirements. For example, Legion file objects can be made to support application-specific access patterns. Consider a file logically containing a two-dimensional Grid of data items: in a traditional file interface, access to a single row or column of the Grid might require multiple file operations, but in Legion an extended file type can be used to represent the 2-D file object, providing additional methods allowing row and column access.

Data migration and replication: In the Legion implementation an object may be *active* or *inactive* at any given instant in time. An active object has an in-memory representation somewhere, i.e. it has an address space and a thread of control. An inactive object is not currently running anywhere, it cannot receive or process messages. Whether an object is active or inactive is irrelevant to users or other objects. The system is responsible for delivering messages to an object regardless of whether is it active or not. The OMS maintains a database of object information. For each object the OMS keeps track of its state (active or inactive), and if active, its address (where it is running), and the location on stable store where its persistent state is stored. In order for the OMS to activate and deactivate an object, the object's class must have the methods *capture_state*() and *restore_from_state*() defined. To deactivate an object the OMS will capture its state, save the state to stable store (disk), update the OMS database, and release the process resources held by the object. Similarly, to activate an object the OMS finds a suitable location (a scheduling decision), allocates process resources, begins execution of the appropriate concrete implementation, reads the state from stable store, invokes the *restore_from_state*() method, and update the OMS database. The capture/restore state mechanism can be used not only to multiplex objects onto limited process resources, it can also be used to provide processes migration by saving the state on one processor and restoring it on another, and for simple fault-tolerance by periodically saving the state of an object, and restoring the object to it's last checkpoint.

II.7.7 Security

Security is an essential part of the Legion design. In a metacomputing environment, the security problem can be divided into two main concerns:

- Protecting the metacomputer's high-level resources, services, and users from each other and from possibly corrupted underlying resources.
- Preserving the security policies of the underlying resources that form the foundation of the metacomputer and minimising their vulnerability to attacks from the metacomputer level.

For example, restricting who is able to configure a metacomputer-wide scheduling service would fall into the first category, and its solution requires metacomputer-specific definitions of identity, authorisation, and access control. Meanwhile, enforcing a policy that permits only those metacomputer users who have local accounts to run jobs on a given host falls into the second category, and it might require a means to map between local identities and metacomputer identities. To satisfy users and administrators, a full security solution must address and reconcile both of these security concerns. Users must have confidence that the data and computations they create within the metacomputer are adequately protected. Administrators need assurances that by adding their resources to a metacomputer (and thus making those resources more accessible and valuable to users), they are not also introducing unreasonable security vulnerabilities into their systems. Attempting to incorporate security as an add-on late in the implementation process has been problematic in a number of first-generation metacomputing systems such as PVM, MPI, and Mentat. To avoid this pitfall, the Legion group has addressed security issues since the earliest design phases. This metacomputing security model has three interrelated design goals:

- *Flexibility.* The framework must be adaptable to many different security policies and allow multiple policies to coexist.
- *Autonomy.* Organisations and users within a metacomputing environment should be able to select and enforce their desired security policies independently.
- *Breadth.* The metacomputer's architectural framework must enable a rich set of security policy features. These goals are strongly driven by a view that a fundamental capability of a metacomputer should be its ability to scale over and across multiple trust domains. A Legion "system" is really a federation of meta- and lower-level resources from multiple domains, each with its own separately evaluated and enforced security policies. As such, there is no central kernel or trusted code base that can monitor and control all interactions between users and resources. Nor is there the concept of a super-user. No one person or entity controls all of the resources in a Legion system.

If it is to satisfy a broad range of security needs, the Legion architecture must allow the implementation of a number of different security features. These include:

- *Isolation.* Components in the metacomputer should be able to insulate themselves from security breaches in other parts of the system. This feature is particularly important in large-scale systems, where it must be generally assumed that at least some of the underlying hosts have been compromised or may even be malicious.
- *Access control.* Resources typically require access control mechanisms that embody authentication and authorisation policies.
- *Identity.* The ability to assert and confirm identity is essential for access control, non-repudiation, and other basic functions.
- *Detection and recovery.* A metacomputing environment should support mechanisms for detecting intrusion and misuse of resources, and for recovering after a security breach.
- *Communication privacy and integrity.* Communication over the networks that bind the metacomputer together may need to be encrypted or protected if the networks cannot themselves be trusted.
- *Standards.* Existing security standards such as Kerberos, ssh, DCE, etc., may need to be integrated into the metacomputing environment to satisfy local administrative policy and to handle legacy software.

The Legion Security model is based on three principles:

- as in the Hippocratic Oath, *do no harm* !
- *caveat empto*r, let the buyer beware.
- *small is beautiful.*

Legion's first responsibility is to minimise the possibility that it will provide an avenue via which an intruder can do mischief to a remote system. The remote system is, by the second principle, responsible for ensuring that it is running a valid copy of Legion — but subject to that, Legion should not permit its corruption.

The second principle means that in the final analysis users are responsible for their own security. Legion provides a model and mechanism that make it feasible, conceptually simple, and inexpensive in the default case, but in the end the user has the ultimate responsibility to determine what policy is to be enforced and how vigorous that enforcement will be. This also models the “real world”; the strongest door with the strongest lock is useless if the user leaves it open.

The third principle simply means, given that one cannot absolutely, unconditionally depend on Legion to enforce security, there is no reason to invest it with elaborate mechanisms. On the contrary, at least intuitively, the simpler the model and the less it does, the lower the probability that a corrupted version can do harm.

As noted earlier, Legion is an object-oriented system. Thus, the unit of protection is the object, and the “rights” to the object are to invoke its member functions (each member function is associated with a distinct right).

This is not a new idea; it dates to at least the Hydra system in the mid-1970’s and is also in some proposed Corba models. Note, however, that it subsumes more common notions such as protection at the level of file systems. In Legion, files are merely one type of user-defined object that happen to have methods read/write/seek/etc. Directories are just another type of object with methods such as lookup/enter/delete/etc. There is no reason why there must be only one type of file or one type of directory and, indeed, these need not be distinguished concepts defined by, or even known to Legion. The basic concepts of the Legion Security Model are minimal; there are just four:

- every object provides certain member functions (that may be defaulted to NIL); the basic are “MayI,” “Iam,” and “Delegate.”.
- user-defined objects can play two security-related roles — those of the “responsible agent”, RA, and the “security agent”, SA. To play these roles they must provide additional member functions known to Legion; the one described here is “pass”.
- every invocation of a member function is performed in an environment consisting of a triple of (unique) object names — those of the operative responsible agent, security agent, and “calling agent”, CA.
- there is a small set of rules for actions that Legion will take, primarily at member function invocation. These rules are defined informally here.

The general approach is that Legion will invoke the known member functions (MayI, etc.), thus giving objects the responsibility of defining and ensuring the policy.

In Legion, access is the ability to call a method on an object. The object may represent a file, a Legion service, a device, or any other resource. Access control is not centralised in any one part of the Legion system. Each object is responsible for enforcing its own access control policy. It *may* collaborate with other objects in making an access decision, and indeed, this allows an administrator to control policy for multiple objects from one point. The Legion architecture does not require this, however.

The general model for access control is that each method call received at an object passes through a *MayI* layer before being serviced. MayI is specified as an event in the configurable Legion protocol stack. MayI decides whether to grant access according to whatever policy it implements. If access is denied, the object will respond with an appropriate security exception, which the caller can handle any way it sees fit. MayI can be implemented in multiple ways. The trivial MayI layer could just allow all access. The default implementation provides a more sophisticated MayI that implements access control lists and credential checking. In this MayI, access control lists can be specified for each method in an object. There are two lists for each method, an *allow* and a *deny.* The entries in the lists are the LOIDs of callers that are granted or denied the right to call the particular method. A deny entry supersedes an allow. Default allow and deny lists can be specified to cover methods that don’t have their own entries.

The LOIDs in the allow and deny lists may specify particular users, the object’s Class Manager, or the object itself. The lists can also include a special token that represents any LOID at all. The LOIDs of objects used to represent groups can also be contained in the lists. Group Objects simply represent a list of member LOIDs, providing methods for querying or modifying membership. Any user in the system can create their own group, listing whichever LOIDs they wish as members, and modifying membership dynamically over time. When a group object LOID is found in an access control list, all of the contained members are logically added to the list. For performance, the results of the membership lookup are cached, but with a short timeout so that group membership changes will be reflected relatively quickly. Groups provide one means for centralising access control policy.

When a method call is received, the credentials it carries are checked by MayI and compared against the access control lists. For example, in the case of a delegated credential, the caller must have included proof of his identity in the call so that MayI can confirm that the credential applies. Multiple credentials can be carried in a call; checking continues until one provides access. Note that credentials provide an alternative way to define groups. If the group owner alone is on the access control list for a method, then he can give delegated credentials to all the members of the group, allowing them to call the method as well.

The default library MayI is configured when an object starts up. The configuration information is passed to it by its Class Manager, which in turn may have inherited the information, or part of it, from the user. For example, the user may have a default access control list for object-mandatory methods that all objects created on his behalf will inherit, while the Class Manager for those objects may specify additional access control lists specific to the particular kinds of objects they manage.

The form of access control provided by the default MayI is sufficient for some kinds of objects, such as file objects, but not for others. For example, Class Managers support a "deactivate" method that allows the caller to bring down an object managed by that Class Manager. Multiple clients of a single Class Manager Object may all need to call this method, but each should be allowed to deactivate only the objects they created. The default MayI doesn't have this ownership information.

To solve this particular case, an additional MayI event handler is added to the Class Manager implementation that can check the arguments of the deactivate call against an internally maintained table of who created which object instance. The configurable, event-based protocol stack makes it easy to replace or supplement the default MayI with extra functionality such as this. The default MayI itself is relatively simple to modify if, for example, new forms of credentials or different kinds of access control lists must be supported. With the Legion security architecture, these types of changes can be made on a local basis without affecting other parts of a Legion system.

Authentication mechanism: One aspect of the previous approach which may be unacceptable at some sites is the use of Legion authentication mechanisms to control access to a host. For example, a site may require that only users with local accounts may access the system, and that those users must be authenticated by a locally adopted authentication system such as Kerberos. Once authentication succeeds, though, normal Legion objects can be created. The Kerberos will be used as a sample authentication mechanism. The Kerberos authentication protocol is fundamentally based on clients obtaining tickets for the use of services. Tickets are unforgeable tokens obtained from a distribution server through a protocol that involves the actual authentication of the user through password entry. To avoid requiring the repeated entry of passwords, a special Ticket Granting Ticket (TGT) is obtained by clients. This TGT is a credential that can then be used for a limited time to obtain further tickets that are required to access individual services. Clients can obtain specially marked TGTs that can be forwarded to proxies for use within a limited time period.

The use of these forwarded TGTs is the basis for employing Kerberos as an authentication mechanism within Legion. The TGT is sent in the implicit parameters of an object creation request to the eventual Host Object. The TGT is equivalent to a bearer credential, and it is treated as such by not being sent in the clear, etc. The Host Object uses the TGT it receives to request a new TGT from the local Kerberos server. If authentication fails, it will not get a TGT. The new TGT grants access to a Kerberized version of the PCD. The Host proceeds to make an object creation request to the Kerberized PCD, supplying the local TGT and the name of the user's local account along with the standard job request information (e.g., the executable path, the path of the object's persistent state directory, etc.). The PCD performs the normal actions of changing the ownership of the directory for the object's state and spawning the object.

The TGTs expire relatively rapidly, and the Host discards them immediately after use. However, to support the continued management of local objects in the absence of the object owners, the PCD will perform certain limited actions on behalf the Host without a TGT. It can stop (i.e., kill) a managed object, it can restart the object, and it can switch the ownership of the object's state directory back and forth between the Vault ownership and user account ownership. The PCD ensures that only accounts of users authenticated via Kerberos are used

An important restriction covers object restarts. The PCD will only restart the same objects under the same accounts that it did when TGTs were presented. It will not restart an object that is already running (thereby creating multiple copies), and it keeps track of its children to prevent this. These conditions prohibit the Host, or anybody contacting the Host, from leveraging off a previously authenticated use of the PCD. Consider the example of a user object running on a Host. If this object becomes deactivated, and then is needed to service a method called by a user other than its owner, it will need to be reactivated without a TGT. The PCD will reactivate the given object only if it had been previously started on the Host using a valid TGT. The PCD will not start additional objects of the same class (i.e., additional processes running the same executable), nor will it start the given object under a different user-id than was originally selected. The effect is the same as if the object had never been deactivated, but had instead continued to execute. This retains the level of access control required by site administrators: processes can only be effectively started by authorised users, and then only through the locally mandated authentication mechanism. For the purposes of long-lived objects, it is possible to simply extend this to support the temporary suspension of objects created by authenticated users, and the subsequent reactivation of these processes by other clients. The services provided by Kerberos (e.g., obtaining forwardable user credentials) are available in other systems, such as the Secure Sockets Layer (SSL). In fact, both Kerberos and SSL can be called through a generic interface: the Generic Secure Service Application Program Interface (GSSAPI). By using GSSAPI, straightforward extensions to other systems such as SSL are possible.

Authorisation mechanisms: Identity is fundamental to higher-level security services such as access control. In Legion, identity can be based most naturally on LOIDs, since all entities of interest (including users) are represented as Legion objects. As a default Legion security practice, one of the LOID fields contains security information including an RSA public key. By including the public key in an object's LOID, other objects can easily encrypt communications to that object, or verify messages signed by it. Objects can just extract the key from the LOID, rather than looking it up in some separate database. By making the key an integral part of an object's name, some kinds of public key tampering have been eliminated. An attacker cannot substitute a new key in a known object's id, because if any part of the LOID is altered, including the key, a new LOID is created that will not be recognised by Class Manager Objects during the binding process, and so on. One drawback though, is that there is no mechanism for revoking an object's key and issuing a new one, as this step implies a complete change of the object's name.

An object normally gets its LOID from its Class Manager when it is created. The Class Manager assigns a new instance number to the object and creates its keys. The resulting LOID and keys are communicated over an encrypted channel to the Host Object on the machine where the object will run. Once the object is up, the Host Object passes its LOID and keys to it over a temporary socket connection. With a LOID in its possession, the object can now begin communicating with other objects using normal Legion mechanisms.

Certain objects are not created by Host Objects and get their LOIDs in different ways. Command-line Legion programs create their LOIDs and keys themselves. The instance number is chosen at random, and the new LOID is registered with a special command-line Class Manager for the domain. The private key is never transmitted. Another special case is the core system objects that are necessary to bootstrap a Legion domain. These have their LOIDs and keys generated by a special domain initialisation program. Users also have LOIDs. Users create their own LOIDs, which are then registered with a user Class Manager and entered in appropriate system groups and access control lists by the respective administrators. When an object such as a command-line program calls another object on behalf of the user, the user's LOID and associated credentials provide the basis for authentication and authorisation.

The ownership of a LOID resides in the user's unique knowledge of the private key hat is paired with their LOID. The private key is kept encrypted on disk, on a smart card, or in some other safe place. Although LOIDs serve as ids in Legion, they are not easily manipulated by people. The same service objects in different Legion systems will have different LOIDs, making it hard to write utility programs based on raw LOIDs. To solve these problems, Legion provides a directory service called *context space* that maps string names to LOIDs. A context contains string entries which may be linked to any kind of Legion object. Objects in a context may be files, hosts, users, etc., as well as other contexts. Context space is similar to a Unix file system; the contexts resemble directories where all the entries are soft links. Contexts make it much easier to identify services and objects in a Legion system. However, this convenience also introduces some risks. Once objects rely on contexts to look up services, the focus for an attacker becomes the contexts themselves. If an attacker compromises a context, he can replace the LOIDs of valid objects with LOIDs of his own. There is nothing new about this type of vulnerability (an attacker who gains root access on a Unix system can easily replace systems programs, for example), but it points out that LOIDs and their integral public keys do not protect against spoofing.

Accounting: In Legion access to the set of resources managed by the system is possible through an interface such as an X-Windows interface. In such an environment a rational user will purchase just the interface, and leave the purchase of expensive resources to other users. It would not be long before no new equipment was purchased, resulting in an impoverished system. To avoid this classic "tragedy of the commons", mechanism and policy are required to encourage good community behaviour. In the case of Legion this takes the form of an accounting system which monitors resource contribution and consumption. Accounting for resource consumption is accomplished by associating each instantiated user object with an owner, and monitoring user object activity. The owner-id is the Unix UID of the shell which launched the application. Resource consumption is monitored on an object-by-object basis. The run-time libraries used by objects are instrumented to collect information such as the amount of CPU used, the number of messages sent and received, the total volume of data moved in and out of the object, and the number of method invocations performed. On object termination this information is forwarded to the instantiation manager. The instantiation manager in turn places the resource data into a host specific accounting database. Periodically the accounting data is collected, merged into a single database, and resource reports are generated. Resource contribution is monitored similarly. Actual resource contribution is derived from the resource consumption database; if resources were used on a host by an object then they were contributed by that host. Offered resources are determined by maintaining thermostat logs. Both resource contribution and consumption are scaled using the resource scale file. The scale file consists of scale factors for each host type, the time of day, and whether the resource was consumed, offered, or contributed. This is used to account for the fact that, for example, a Sparc IPC CPU second at 1:00 AM is not worth nearly as much as an SGI CPU second at noon. The scale factors can be thought of as prices. Using the accounting files and the resource scale file a resource balance for each user can be calculated. The resource balance is the users amount of surplus or deficit. A system-wide surplus, available to system administrators, can be generated by maintaining a spread between the price for resources consumed and resources offered. The mechanism lets us know who is using and who is contributing resource and in what quantities. Without a policy on resource consumption, collecting the information is an exercise in programming only. Policy is still being worked out with resource holders.

Credential delegation: For a resource, the essential step in deciding whether to grant an access request is to determine the identity of the caller. A user who communicates directly with the target object, can establish their identity relatively easily with an authentication protocol, which typically involves performing an operation that only someone in possession of the users private key can do. In a distributed object system, however, the user typically accesses resources indirectly, and objects need to be able to perform actions on their behalf (for example, a user does not invoke the services of a Host Object directly, but instead relies on a Class Manager to use Host services). Though intermediate objects could in principle be given the user's private key, the risk involved is too great. Given the user's private key, an object can do anything the user can. That's more privilege than is usually necessary.

To avoid the need for sharing the private key, resources can call back to the user or their trusted proxy when they receive access requests in the user's name. This step puts control back in the user's hands. There are several drawbacks to this approach, though. First, the fine-grain control afforded by authorisation callbacks may be mostly illusory. It can be very difficult to craft policies for a user proxy (or even the real user in person!) that are much more than "grant all requests"—too much contextual and semantic information is generally missing from the request. Beyond this barrier, callbacks are expensive and do not scale well. Though there is nothing in the Legion architecture that precludes using callbacks for particular objects or resources (and in some cases they may indeed be appropriate), calling back for authorisation is not a universal solution. In Legion, after all, every object represents a resource of some type, and a callback on every method call would be a crippling performance hit.

The intermediate solution between these approaches is to issue *credentials* to objects. A credential is a list of rights granted by the credential's maker, presumably the user. They can be passed through call chains. When an object requests a resource, it presents the credential to gain access. The resource checks the rights in the credential, and who the maker is, and uses that information in deciding to grant access.

There are two main types of credentials in Legion: *delegated credentials* and *bearer credentials*. A delegated credential specifies exactly who is granted the listed rights, whereas simple possession of a bearer credential grants the rights listed within it. A Legion credential specifies the period the credential is valid, who is allowed to use the credential, the rights—which methods may be called on which specific objects or class of objects. If missing, fields default to "all." The credential also includes the identity of its maker, who digitally signs the complete credential.

A sample delegated credential is "*[Object A may call object B's method M as Alice during the period T] signed Alice*." To use this credential, A must authenticate to B when it makes its request. There is no need to worry about protecting the credential from theft, because only A can use it. Moreover, the specification of the target object, the method to be called, and the timeout closely limit how this credential can be used. Greater specificity lowers the risk of giving away rights that can be misused by other parties.

It is not always possible to specify a credential so narrowly. Call chains can be long, and the identity of the final object making a resource request may be unknown. If the call chain branches out, several different objects from different classes may need to make calls on the user's behalf. The specificity of credentials can be loosen to handle these cases, but risk increases at the same time. The credential "*[The bearer has all of Alice's rights forever] signed Alic*e" is very convenient to give to objects, as there is no danger of accidentally restricting any actions, but should the credential be stolen, Alice is in trouble.

In Legion, tools or commands directly executed by the user create the credentials they need to carry out their actions. The credentials are made as specific as possible. For example, if the user executes the command to create an object instance, that command will create a credential that authorises the specified Class Manager to create an object instance on a Host Object. Of course, the Host Object that is contacted might reject the object creation request, but this rejection would be because the user was not authorised to use that resource, not because the Class Manager lacked the authorisation to act on the user's behalf.

If a long-lived bearer credential is known to be stolen, the recovery strategy is to create a new LOID for the user. The new LOID has a different instance number but reuses the user's official X.509 certificate for its security field. The resource providers must then modify the appropriate system groups and replace the user's old LOID with the new one. The user's old objects will need new credentials to access resources. The flaw, though, is in having long-lived bearer credentials to begin with.

II.7.8 References

[1] Andrew S. Grimshaw, William A. Wulf, James C. French, Alfred C. Weaver, Paul F. Reynolds Jr. A Synopsis of the Legion Project, Technical Report No. CS-94-20 June, 1994.

[2] Andrew S. Grimshaw, William A. Wulf, James C. French, Alfred C. Weaver, Paul F. Reynolds Jr., Legion: The Next Logical Step Toward a Nationwide Virtual Computer, Technical Report No. CS-94-21, June, 1994.

[3] Andrew S. Grimshaw, Anh Nguyen-Tuong, William A. Wulf, Campus-Wide Computing: Early Results Using Legion at the University of Virginia.

[4] Wm A. Wulf, Chenxi Wang, Darrell Kienzle, A New Model of Security for Distributed Systems, Computer Science Technical Report CS-95-34, University of Virginia, August 10, 1995.

[5] Mike Lewis, Andrew Grimshaw, The Core Legion Object Model , Department of Computer Science University of Virginia August 1995.

[6] Adam Ferrari, Frederick Knabe, Marty Humphrey, Steve Chapin, and Andrew Grimshaw, A Flexible Security System for Metacomputing Environments, Department of Computer Science University of Virginia, Technical Report CS-98-36 December 1, 1998.

[7] Andrew Grimshaw, Adam Ferrari, Fritz Knabe, Marty Humphrey, Legion: An Operating System for Wide-Area Computing.

[8] Marty Humphrey, Frederick Knabe, Adam Ferrari, Andrew Grishaw, Accountability and Control of Process Creation in the Legion Metasystem.

[9] Brian S. White Andrew S. Grimshaw Anh Nguyen-Tuong, Grid-Based File Access: The Legion I/O Model.

[10] Andrew Grishaw, William Wulf, Legion - A View From 50000 Feets.

[11] Steve J. Chapin, Dimitrios Katramatos, John Karpovich, and Andrew Grimshaw, Resource Management in Legion.

All these papers are accessible at http://legion.virginia.edu/papers.html

II.8 Condor

II.8.1 Introduction

Years ago, the engineers and scientists relied on large centralised mainframe and big supercomputers to do computational work. A large number of individuals and groups would have to pool their financial resources to afford such a machine. Users would then have to wait for their turn on the mainframe, and would only have a certain amount of time allotted to them. While this environment was inconvenient for the users, it was very efficient, since the mainframe was busy nearly all the time.

As computers became smaller, faster, and cheaper, users started to move away from centralised mainframes and started purchasing personal desktop workstations and PCs. An individual or small group could afford a computing resource that was available whenever they wanted it. It was usually far slower than the large centralised machine, but since they had exclusive access, it was worth it. Now, instead of one giant computer for a large institution, there might be hundreds or thousands of personal computers. This is an environment of distributed ownership, where individuals throughout an organisation own their own resources. The total computational power of the institution as a whole might rise dramatically as the result of such a change, but because of distributed ownership individuals could not capitalise on the institutional growth of computing power. And while distributed ownership is more convenient for the users, it is also much less efficient. Many personal desktop machines sit idle for very long periods of time while their owners are busy doing other things.

Condor is a software system that creates a High Throughput Computing (HTC) environment by effectively harnessing wasted CPU cycles in a cluster of any number of workstations of possibly different architectures and operating systems, that are connected by a network. Although Condor can manage a dedicated cluster of workstations, a key appeal of Condor is its ability to effectively harness non-dedicated, pre-existing sets of machines that are distributively owned (i.e. not maintained or administered by a central authority) such as machines sitting on people's desks in offices or labs.

In other words, from the user's point of view, Condor is a specialised batch system for managing compute-intensive jobs. Condor provides a queuing mechanism, scheduling policy, priority scheme, and resource classifications. Users submit their compute jobs to Condor, Condor puts the jobs in a queue, runs them, and then informs the user as to the result. Unlike traditional batch systems, Condor is also designed to effectively utilise non-dedicated machines to run jobs. By being told to only run compute jobs on machines which are currently not being used (no keyboard activity, no load average, no active telnet users, etc), Condor can effectively harness otherwise idle machines throughout a pool of machines

Condor has several unique capabilities and features at its disposal which are geared towards effectively utilising non-dedicated resources that are not owned or managed by a centralised resource. These include:

- **Checkpoint and Migration.** Users of Condor may be assured that their jobs will eventually complete even in an opportunistic computing environment. If a user submits a job to Condor which runs on another's workstation, but the job is not finished when the workstation owner returns, the job can be checkpointed. The job continues by migrating to another machine. It makes progress toward its completion by the checkpoint and migration feature. Condor's "periodic checkpoint feature" periodically checkpoints a job even in lieu of migration in order to safeguard the accumulated computation time on a job from being lost in the event of a system failure such as the machine being shutdown or a crash.

- **Remote System Calls.** Despite running jobs on remote machines, the Condor standard universe execution mode preserves the local execution environment via remote system calls. Users do not have to worry about making data files available to remote workstations or even obtaining a login account on remote workstations before Condor executes their programs there. The program behaves under Condor as if it were running as the user that submitted the job on the workstation where it was originally submitted, no matter on which machine it really ends up executing on.

- **No Changes Necessary to User's Source Code.** No special programming is required to use Condor. Condor is able to run normal non-interactive UNIX or NT programs. The checkpoint and migration of programs by Condor is transparent and automatic, as is the use of remote system calls. If these facilities are desired, the user only re-links the program. The code is not compiled or changed.

- **Sensitive to the Desires of Workstation Owners.** The owner of a workstation has complete priority over the workstation, by default. A workstation owner is generally happy to let others compute on the workstation while it is idle, but wants the workstation back promptly upon returning. The owner does not want to take special action to regain control. Condor handles this automatically.

- **Resource management with ClassAds**. The ClassAd mechanism in Condor provides an extremely flexible, expressive framework for matchmaking resource requests with resource offers. One result is that users can easily request practically any resource, both in terms of job requirements and job desires. For example, a user can require that a job run on a machine with 64 Mbytes of RAM, but state a preference for 128 Mbytes if available. Likewise, a workstation can state a preference in a resource offer to run jobs from a specified set of users, and it can require that there be no interactive workstation activity detectable between 9 am and 5 pm before starting a job. Job requirements/preferences and resource availability constraints can be described in terms of powerful expressions, resulting in Condor's adaptation to nearly any desired policy.

Instead of running a CPU-intensive job in the background on their own workstation, users submit their job to Condor. Condor will then find an available machine (i.e. machine which is currently idle) on the network and begin running the job on that machine. When Condor detects that a machine running a Condor job would no longer be available (perhaps because the owner of the machine came back from lunch and started typing on the keyboard), Condor checkpoints the job and then migrates it over the network to a different machine which would otherwise be idle. Condor then continues the execution of the job on the new machine from precisely where it left off. If no machine on the network is currently available, then the job is stored in a queue on disk until a machine becomes available. An important issue is that execution of the checkpointed job can be continued only on a machine that is binary compatible with the machine on which the job was checkpointed.

Users may also need to run their single job for *N* different times (e.g. 100 times probably on 100 different data sets). In this case, Condor can be a real time saver. With one command, all *N* runs are submitted to Condor. Depending upon the number of machines in the organisation's Condor pool, there could be dozens or even hundreds of otherwise idle machines running the job at any given moment.

A Condor pool is comprised of at least a single machine which serves as the "Central Manager" (CM), and an arbitrary number of other machines that have joined the pool. Conceptually, the pool is a collection of resources (machines) and resource requests (jobs). The role of Condor is to match waiting requests with available resources. Every part of Condor sends periodic updates to the Central Manager, the centralised repository of information about the state of the pool. Periodically, the Central Manager assesses the current state of the pool and tries to match pending requests with the appropriate resources.

Each resource has an owner, the user who works at the machine. This person has absolute power over their own resource and Condor goes out of its way to minimise the impact on this owner caused by Condor. It is up to the resource owner to define a policy for when Condor requests will serviced and when they will be denied.

Each resource request has an owner as well: the user who submitted the job. These people want Condor to provide as many CPU cycles as possible for their work. Often the interests of the resource owners are in conflict with the interests of the resource requesters.

Every machine in a Condor pool can play a variety of roles. Most machines play more than one role simultaneously. Certain roles can only be performed by a single machine in a pool. Table 3 lists the various roles and briefly describes each and the resources required on the machine playing the given role.

Role	*Description and requirements*
Central Manager	There can be only one Central Manager for a Condor pool. The machine is the collector of information, and the negotiator between resources and resource requests. These two halves of the Central Manager's responsibility are performed by separate daemons, so it would be possible to have different machines providing those two services. However, normally they both live on the same machine. This machine plays a very important part in the Condor pool and should be reliable. If this machine crashes, no further matchmaking can be performed within the Condor system (although all current matches remain in effect until they are broken by either party involved in the match). Therefore, a machine that is likely to be online all the time, or at least will be rebooted quickly if something goes wrong, should be chosen as the Central Manager. The Central Manager will ideally have a good network connection to all the machines in your pool, since they all send updates over the network to it. All queries go to the Central Manager.
Execute	Any machine in a pool (including the Central Manager) can be configured for whether or not it should execute Condor jobs. Obviously, some of your machines will have to serve this function or your pool will not be very useful. Being an execute machine does not require many resources at all. About the only resource that might matter is disk space, since if the remote job dumps core, that file is first dumped to the local disk of the execute machine before being sent back to the submit machine for the owner of the job. However, if there is not much disk space, Condor will simply limit the size of the core file that a remote job will drop. In general the more resources a machine has (swap space, real memory, CPU speed, etc.) the larger the resource requests it can serve. However, if there are requests that don't require many resources, any machine in your pool could serve them.
Submit	Any machine in a pool (including the Central Manager) can be configured for whether or not it should allow Condor jobs to be submitted. The resource requirements for a submit machine are actually much greater than the resource requirements for an execute machine. First of all, every job that you submit that is currently running on a remote machine generates another process on your submit machine. So, if you have lots of jobs running, you will need a fair amount of swap space and/or real memory. In addition all the checkpoint files from your jobs are stored on the local disk of the machine you submit from. Therefore, if your jobs have a large memory image and you submit a lot of them, you will need a lot of disk space to hold these files. This disk space requirement can be somewhat alleviated with a checkpoint server (described below), however the binaries of the jobs you submit are still stored on the submit machine.
Checkpoint Server (optional)	One machine in your pool can be configured as a checkpoint server. This is optional, and is not part of the standard Condor binary distribution. The checkpoint server is a centralised machine that stores all the checkpoint files for the jobs submitted in your pool. This machine should have lots of disk space and a good network connection to the rest of your pool, as the traffic can be quite heavy

Table 3: roles of different machines in a Condor pool

Every Condor job involves three machines. One is the submitting machine, where the job is submitted from. The second is the Central Manager, which finds an idle machine for that job. The third is the executing machine, the computer that the job actually runs on. In reality, a single machine can perform two or even all three of these roles. In such cases, the submitting machine and the executing machine might actually be the same piece of hardware, but all the mechanisms described here will continue to function as if they were separate machines. The executing machine is often many different computers at different times during the course of the job's life. However, at any given moment, there will either be a single execution machine, or the job will be in the job queue, waiting for an available computer.

Every machine in the pool has certain properties: its architecture, operating system, amount of memory, the speed of its CPU, amount of free swap and disk space, and other characteristics. Similarly, every job has certain requirements and preferences. A job must run on a machine with the same architecture and operating system it was compiled for. Beyond that, jobs might have requirements as to how much memory they need to run efficiently, how much swap space they will need, etc. Preferences are characteristics the job owner would like the executing machine to have but which are not absolutely necessary. If no machines that match the preferences are available, the job will still function on another machine. The owner of a job specifies the requirements and preferences of the job when it is submitted. The properties of the computing resources are reported to the Central Manager by the condor_startd daemon on each machine in the pool. The task of the condor_negotiator daemon is not only to find idle machines, but machines with properties that match the requirements of the jobs, and if possible, the job preferences.

When a match is made between a job and a machine, the Condor daemons on each machine are sent a message by the Central Manager. The condor_schedd daemon on the submitting machine starts up condor_shadow daemon. This acts as the connection to the submitting machine for the remote job, the shadow of the remote job on the local submitting machine. The condor_startd on the executing machine also creates another daemon, the condor_starter. The condor_starter actually starts the Condor job, which involves transferring the binary from the submitting machine. (See Figure 14, page 76). The condor_starter is also responsible for monitoring the job, maintaining statistics about it, making sure there is space for the checkpoint file, and sending the checkpoint file back to the submitting machine (or the checkpoint server, if one exists). In the event that a machine is reclaimed by its owner, it is the condor_starter that vacates the job from that machine.

A job's code does not have to be modified in any way to take advantage of the benefits that Condor offers, though it must be linked with the Condor libraries. Once re-linked, jobs gain two crucial abilities: they can checkpoint and perform remote system calls. These jobs are called "standard" Condor jobs. Condor also provides a mechanism to run binaries that have not been re-linked, which are called "vanilla" jobs. Vanilla jobs do not gain any of these benefits, but they are still scheduled by Condor to run on idle machines. Since vanilla jobs are not linked with the Condor library, they are not capable of performing remote system calls. Because of this, they can not access remote filesystems. For a vanilla job to properly function, it must run on a machine with a local filesystem that contains all the input files it will need, and where it can write its output. Normally, this would only be the submit machine, and any machines that had a shared filesystem with it via some sort of network filesystem like NFS or AFS. Moreover, the job must run on a machine where the user has the same UID as on the submit machine, so that it can access those files properly. In a distributively owned computing environment, these are clearly not necessarily properties of every machine in the pool, though they are hopefully properties of some of them. Condor defines two attributes of every machine in the pool, the UID domain and the filesystem domain. When a vanilla job is submitted, the UID and filesystem domain of the submit machine are added to the job's requirements. The negotiator will only match this job with a machine with the same UID and filesystem domain, ensuring that local filesystem access on the execution machine will be equivalent to filesystem access on the submit machine.

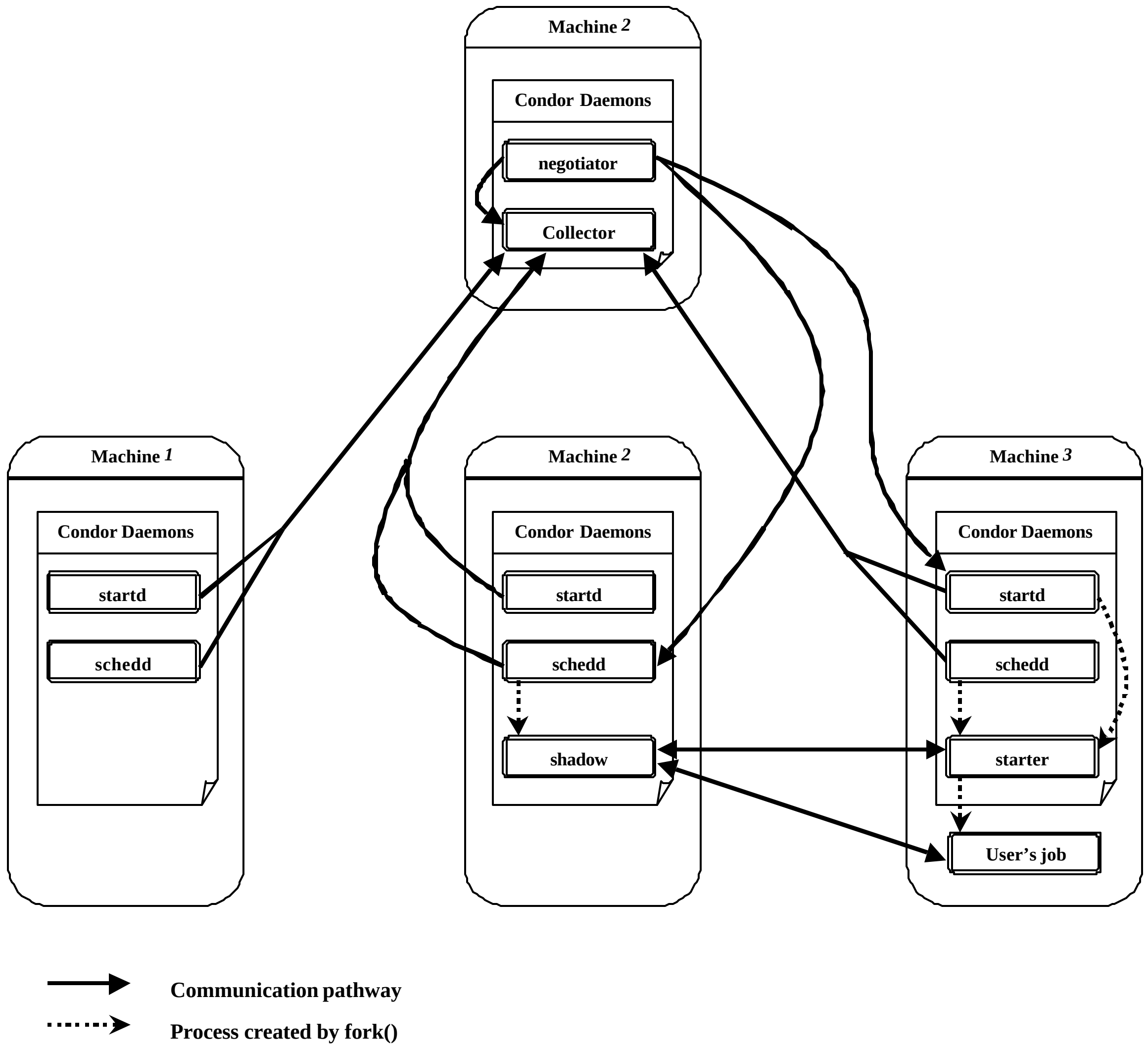


Figure 14: architecture of a Condor pool: job submitted on Machine*2* running on Machine *3*

A universe in Condor defines an execution environment. The latest version of Condor supports five different universes for user jobs:

- **Standard**
- **Vanilla**
- **PVM**
- **MPI**
- **Globus**

The Universe attribute is specified in the submit description file. If the universe is not specified, then it will default to standard.

- **Standard Universe:** In the standard universe, Condor provides checkpointing and remote system calls. These features make a job more reliable and allow it uniform access to resources from anywhere in the pool. To prepare a program as a standard universe job, it must be re-linked with condor_compile. Most programs can be prepared as a standard universe job, but there are a few restrictions.

  To convert your program into a standard universe job, you must use condor_compile to re-link it with the Condor libraries. Simply put condor_compile in front of your usual link command. You do not need to modify the program's source code, but you do need access to its un-linked object files. A commercial program that is packaged as a single executable file cannot be converted into a standard universe job. For example, if you normally link your job by executing:

  ```
  % cc main.o tools.o -o program
  ```

  Then you can re-link your job for Condor with:

  ```
  % condor_compile cc main.o tools.o -o program
  ```

  There are a few restrictions on standard universe jobs:

  - Multi-process jobs are not allowed. This includes system calls such as fork(), exec(), and system().
  - Interprocess communication is not allowed. This includes pipes, semaphores, and shared memory.
  - Network communication must be brief. A job may make network connections using system calls such as socket(), but a network connection left open for long periods will delay checkpointing and migration.
  - Sending or receiving the SIGUSR2 or SIGTSTP signals is not allowed. Condor reserves these signals for its own use. Sending or receiving all other signals is allowed.
  - Alarms, timers, and sleeping are not allowed. This includes system calls such as alarm(), getitimer(), and sleep().
  - Multiple kernel-level threads are not allowed. However, multiple user-level threads are allowed.
  - Memory mapped files are not allowed. This includes system calls such as mmap() and munmap().
  - File locks are allowed, but not retained between checkpoints.
  - All files must be opened read-only or write-only. A file opened for both reading and writing will cause trouble if a job must be rolled back to an old checkpoint image. For compatibility reasons, a file opened for both reading and writing will result in a warning but not an error.
  - A fair amount of disk space must be available on the submitting machine for storing a job's checkpoint images. A checkpoint image is approximately equal to the virtual memory consumed by a job while it runs. If disk space is short, a special checkpoint server can be designated for storing all the checkpoint images for a pool.
  - On Digital Unix (OSF/1), HP-UX, and Linux, your job must be statically linked. Dynamic linking is allowed on all other platforms.

- **Vanilla Universe:** The vanilla universe in Condor is intended for programs which cannot be successfully re-linked. Shell scripts are another case where the vanilla universe is useful.

Unfortunately, jobs run under the vanilla universe cannot checkpoint or use remote system calls. This has unfortunate consequences for a job that is partially completed when the remote machine running a job must be returned to its owner. Condor has only two choices. It can suspend the job, hoping to complete it at a later time, or it can give up and restart the job from the beginning on another machine in the pool.

- **PVM Universe:** The PVM universe allows programs written to the Parallel Virtual Machine interface to be used within the opportunistic Condor environment.

- **MPI Universe:** The MPI universe allows programs written to the MPICH interface to be used within the opportunistic Condor environment.

- **Globus Universe:** The Globus universe in Condor is intended to provide the standard Condor interface to users who wish to start Globus system jobs from Condor. Each job queued in the job submission file is translated into a Globus RSL string and used as the arguments to the globusrun program.

II.8.2 Application description and Resource description

Classified Advertisements (ClassAds)

Classified Advertisements are used for describing jobs, workstations, and other resources. They are exchanged by Condor processes to schedule jobs. They are logged to files for statistical and debugging purposes. They are used to enquire about current state of the system.

A ClassAd is a mapping from attribute names to expressions. In the simplest cases, the expressions are simple constants (integer, floating point, or string). A ClassAd is thus a form of property list. Attribute expressions can also be more complicated. There is a protocol for evaluating an attribute expression of a ClassAd versus another Ad. For example, the expression “other.size > 3” in one Ad evaluates to “true” if the other Ad has an attribute named size and the value of that attribute is (or evaluates to) an integer greater than three. Two ClassAds match if each ad has an attribute requirements that evaluates to “true” in the context of the other ad. ClassAd matching is used by the Condor Central Manager to determine the compatibility of jobs and workstations where they may be run.

Condor’s ClassAds are analogous to the classified advertising section of the newspaper. Sellers advertise specifics about what they have to sell, hoping to attract a buyer. Buyers may advertise specifics about what they wish to purchase. Both buyers and sellers may have constraints that need to be satisfied. For instance, perhaps the buyer is not willing to spend more than X dollars, and the seller requires to receive a minimum of Y dollars. Furthermore, both want to rank requests from the other in such a fashion that is to their advantage. Certainly a seller would rank a buyer offering $50 dollars for a service higher than a different buyer offering $25 for the same service. In Condor, users submitting jobs can be thought of as buyers of compute resources and machine owners are the sellers.

All machines in the Condor pool advertise their attributes, such as available RAM memory, CPU type and speed, virtual memory size, current load average, and many other static and dynamic properties, in a machine ClassAd (Resource Offer Ad). The machine ClassAd also advertises under what conditions it is willing to run a Condor job and what type of job it would prefer. These policy attributes can reflect the individual terms and preferences by which all the different owners have graciously allowed their machine to be part of the Condor pool. For example, a machine may advertise that it is only willing to run jobs at night and when there is nobody typing at the keyboard. In addition, machines may advertise a preference (rank) for running jobs submitted by either their owners or co-workers whenever possible. Likewise, when submitting a job, you can specify many different job attributes, including whatever requirements and preferences for whatever type of machine you’d like this job to use. For instance, perhaps you’re looking for the fastest floating-point machine available, i.e. you want to rank matches based upon floating-point performance. Or perhaps you only care that the machine has a minimum of 128 Meg of RAM. Or perhaps you’ll just take any machine you can get! These job attributes and requirements are bundled up into a job ClassAd (Resource Request Ad) and published to Condor. Resource Request Ad defines both the required and desired set of resources to run the job. Similarly, a Resource Offer Ad can define requirements and preferences.

II.8.3 Contract specification methods

Each resource has an owner, the user who works at the machine. Owners have absolute power over their own resources and Condor goes out of its way to minimise the impact on this owner caused by Condor. It is up to the resource owner to define a policy for when Condor requests will be serviced and when they will be denied.

Resources (in particular, machines) are only allocated to Condor jobs when they are idle. The individual owners of the machines specify to Condor what their definition of "idle" is. Usually, this involves some combination of the load on the CPU, and the time that the keyboard and mouse have been idle. If a machine is running a Condor job and its owner returns, Condor must make the machine available to the user again. Condor performs a checkpoint of the job and puts it back in the job queue to be matched with another idle machine whenever one is available.

II.8.4 Resource management

Grid information services: A Condor pool is comprised of at least a single machine which serves as the "Central Manager" (CM), and an arbitrary number of other machines that have joined the pool. The "Central Manager" keeps track of all the resources and jobs in the pool. All of the condor_schedds and condor_startds (special Condor daemons, mentioned in section A) in the entire pool report their information to a daemon running on the CM called the "condor_collector". The collector maintains a global view, and can be queried for information about the status of the pool. Another daemon on the CM, the "condor_negotiator", periodically takes information from the collector to find idle machines and match them with waiting jobs. This process is called a "negotiation cycle" and usually happens every five minutes.

Brokerage services: Match-making resource owners with resource consumers is the cornerstone of a successful HTC environment. Unlike many other compute cluster resource management systems which attach properties to the job queues themselves (resulting in user confusion over which queue to use as well as administrative hassle in constantly adding and editing queue properties to satisfy user demands), Condor implements a clean design called ClassAds (described in section B). Condor acts as a broker (or in other words "as a matchmaker") by continuously reading through all of the job ClassAds (Resource Request Ads) and all of the machine ClassAds (Resource Offer Ads) and matching and ranking job ads with machine ads, while making certain that all requirements in both ads are satisfied. During this match-making process, Condor also takes several layers of priority values into consideration: the priority the user assigned to the Resource Request ad, the priority of the user which submitted the ad, and desire of machines in the pool to accept certain types of ads over others.

Advanced reservation: None.

Scheduling mechanisms:

- *Priorities and Pre-emption:* Condor enforces that each user gets his fair share of machines according to user priority both when allocating machines which become available and by priority pre-emption of currently allocated machines. For instance, if a low priority user is utilising all available machines and suddenly a higher priority user submits jobs, Condor will immediately checkpoint and vacate jobs belonging to the lower priority user. This will free up machines that Condor will then give over to the higher priority user. Condor will not starve the lower priority user; it will pre-empt only enough jobs so that the higher priority user's fair share can be realised (based upon the ratio between user priorities).

To prevent thrashing of the system due to priority pre-emption, the Condor site administrator can define the conditions that must be met for a pre-emption to occur, by specifying PREEMPTION_REQUIREMENTS expression in Condor's configuration. It is usually defined to deny pre-emption if a current running job has been running for a relatively short period of time. This effectively limits the number of pre-emptions per resource per time interval. For instance, the default expression that ships with Condor is configured to only pre-empt lower priority jobs that have run for at least one hour. So in the previous example, in the worse case it could take up to a maximum of one hour until the higher priority user receives his fair share of machines.

- *User Priorities in the Condor System:* Condor uses priorities to determine machine allocation for jobs. The numerical priority value assigned to a user is inversely related to the goodness of the priority, so a user with lower numerical priority gets more resources than a user with higher numerical priority (e.g. a user with a priority of 10 will get twice as many machines as a user with a priority of 20)

  User priorities are keyed on "username@domain". The domain name to use, if any, is configured by the Condor site administrator. Thus, user priority and therefore resource allocation is not impacted by which machine the user submits from or even if the user submits jobs from multiple machines.

  There are two priority values assigned to Condor users:

  - **Real User Priority (RUP)**, which measures resource usage of the user. A user's RUP measures the resource usage of the user through time. Every user begins with a RUP of one half (0.5), and at steady state, the RUP of a user equilibrates to the number of resources used by that user. Therefore, if a specific user continuously uses exactly ten resources for a long period of time, the RUP of that user stabilises at ten. However, if the user decreases the number of resources used, the RUP gets better and if the number of machines a user currently has is greater than the user's priority, then the user's priority will worsen by numerically increasing over time. The long-term result is fair-share access across all users. The speed at which Condor adjusts the priorities can be set in Condor's configuration, by the macro PRIORITY_HALFLIFE, a time period defined in seconds. The default is one day. Intuitively, if the PRIORITY_HALFLIFE in a pool is set to 86400 (one day), and if a user whose RUP was 10 removes all his jobs, the user's RUP would be 5 one day later, 2.5 two days later, and so on.

    This definition ensures that the RUP can be calculated over non-uniform time intervals. The time interval varies due to events internal to the system, but Condor guarantees that unless the Central Manager machine is down, no matches will be unaccounted for due to this variance.

  - **Effective User Priority (EUP)**, which determines the number of resources the user can get. The effective user priority (EUP) of a user is used to determine how many resources that user may receive. The EUP is linearly related to the RUP by a priority factor which may be defined on a per-user basis. Unless otherwise configured, the priority factor for all users is 1.0, and so the EUP is the same as the RUP. However, if desired, the priority factors of specific users (such as remote submitters) can be increased so that others are served preferentially. The number of resources that a user may receive is inversely related to the ratio between the EUPs of submitting users. Therefore user A with EUP=5 will receive twice as many resources as user B with EUP=10 and four times as many resources as user C with EUP=20. However, if A does not use the full number of allocated resources, the available resources are repartitioned and distributed among remaining users according to the inverse ratio rule.

Condor supplies mechanisms to directly support two policies in which EUP may be useful:

- *Nice users.* A job may be submitted with the parameter nice_user set to TRUE in the submit command file. A nice user job gets its RUP boosted by the NICE_USER_PRIO_FACTOR priority factor specified in the configuration file, leading to a (usually very large) EUP. This corresponds to a low priority for resources. These jobs are therefore equivalent to Unix background jobs, which use resources not used by other Condor users.
- *Remote Users.* The flocking feature of Condor allows the condor_schedd to submit to more than one pool. In addition, the submit-only feature allows a user to run a condor_schedd that is submitting jobs into another pool. In such situations, submitters from other domains can submit to the local pool. It is often desirable to have Condor treat local users preferentially over these remote users. If configured, Condor will boost the RUPs of remote users by REMOTE_PRIO_FACTOR specified in the configuration file, thereby lowering their priority for resources.

Co-scheduling: None.

Application support: None.

II.8.5 Communication methods

Communication models: None.

Communication protocols: None.

Quality of Service: None.

Fault tolerance : None.

Services supporting collaboration and remote instrument control: None.

Application support:

- *Condor-PVM:* Condor has a PVM submit Universe which allows the user to submit PVM jobs to the Condor pool. Condor-PVM is an optional Condor module. Condor-PVM provides a framework to run in Condor's opportunistic environment parallel applications written with PVM. This means that a user does not need a set of dedicated machines (unlike pure PVM applications) to run PVM applications. Condor can be used to dynamically construct PVM virtual machines out of non-dedicated desktop machines on a network which would have otherwise been idle. In Condor-PVM, Condor acts as the resource manager for the PVM daemon. Whenever the user's PVM program asks for nodes (machines), the request is re-mapped to Condor. Condor then finds a machine in the Condor pool via the usual mechanisms, and adds it to the PVM virtual machine. If a machine needs to leave the pool, your PVM program is notified of that as well via the normal PVM mechanisms.

- *The Master-Worker Paradigm:* One of the most common parallel programming paradigms is the master-worker (or pool of tasks) arrangement. In a master-worker program model, one node acts as the controlling master for the parallel application and sends pieces of work out to worker nodes. The worker node does some computation, and sends the result back to the master node. The master has a pool of work that needs to be done, and simply assigns the next piece of work out to the next worker that becomes available.

  Not all parallel programming paradigms lend themselves to an opportunistic environment. In such an environment, any of the nodes could be pre-empted and therefore disappear at any moment. The master-worker model, on the other hand, is a model that can work well. The idea is the master needs to keep track of which piece of work it sends to each worker. If the master node is then informed that a worker has disappeared, it puts the piece of work it assigned to that worker back into the pool of tasks, and sends it out again to the next available worker. If the master notices that the number of workers has dropped below an acceptable level, it could request for more workers (via pvm_addhosts()) or it could request a replacement node every single time it is notified that a worker has gone away. The point is that in this paradigm, the number of workers is not important (although more is better!) and changes in the size of the virtual machine can be handled naturally.

  Condor-PVM is designed to run PVM applications which follow the master-worker paradigm. Condor runs the master application on the machine where the job was submitted and will not pre-empt it. Workers are pulled in from the Condor pool as they become available.

- *Compatibility between Condor-PVM and regular PVM:* Condor-PVM does not define a new API. Programs can simply use the existing resource management PVM calls such as pvm_addhosts() and pvm_notify(). Because of this, some master-worker PVM applications are ready to run under Condor-PVM with no changes at all. Regardless of using Condor-PVM or not, it is good master-worker design to handle the case of a worker node disappearing, and therefore many programmers have already constructed their master program with all the necessary logic for fault-tolerance purposes.

  In fact, regular PVM and Condor-PVM are binary compatible with each other. The same binary which runs under regular PVM will run under Condor, and vice-versa. There is no need to re-link for Condor-PVM. This permits easy application development (develop your PVM application interactively with the regular PVM console, XPVM, etc) as well as binary sharing between Condor and some dedicated MPP systems.

- *Runtime differences between Condor-PVM and regular PVM:*
  - **Concept of machine class**. Under Condor-PVM, machines of different architectures belong to different machine classes. Machine classes are numbered 0, 1, ... , etc. A machine class can be specified by the user in the submit-description file when the job is submitted to Condor.
  - **PVM function: "pvm_addhosts()".** When the application needs to add a host machine, it should call pvm_addhosts() with the first argument as a string that specifies the machine class. For example, to specify class 0, a pointer to string "0" should be used as the first argument. Condor will find a machine that satisfies the requirements of class 0 and adds it to the PVM virtual machine. Furthermore, pvm_addhosts() no longer blocks under Condor. It will return immediately, before the hosts are actually added to the virtual machine. After all, in a non-dedicated environment the amount of time it takes until a machine becomes available is not bound. The user should simply call pvm_notify() before calling pvm_addhosts(), so that when a host is added later, the user will be notified via Condor in the usual PVM fashion (via a PvmHostAdd notification message).

  - **PVM function: "pvm_notify()"**. Under Condor, two additional possible notification requests have been added, PvmHostSuspend and PvmHostResume, to the function pvm_notify(). When a host is suspended (or resumed) by Condor, if the user has called pvm_notify() with that host tid and with PvmHostSuspend (or PvmHostResume) as arguments, then the application will receive a notification for the corresponding event.
  - **PVM function: "pvm_spawn()"**. If the flag in pvm_spawn() is PvmTaskArch, then the string specifying the desired architecture class should be used. Typically, if a user is using only one class of machine in a virtual machine, "0" is specified as the desired architecture. Furthermore, under Condor only one PVM task can be spawned per node, since Condor's typical setup at most sites will suspend or vacate a job if the load on its machine is higher than a specified threshold.

- *MPI jobs in Condor:* In addition to PVM, Condor also supports the execution of parallel jobs that utilise MPI. The latest Condor implementation supports the following features:

  - There are no alterations to the MPICH implementation. You can directly use the version from Argonne National Labs.
  - You do not have to re-compile or re-link your MPICH job. Just compile it using the regular mpicc. Note that you have to be using the ch_p4 subsystem provided by Argonne.
  - The communication speed of the MPI nodes is not affected by running it under Condor.

However, there are some limitations:

  - The MPI job must be compiled with MPICH, Argonne National Labs' implementation of MPI. Specifically, you must use the "ch_p4" device for MPICH. Your version of MPICH must not be compiled with the path to `rsh` hard-coded into the library (This could possibly occur as a result of running configure as `./configure -rsh=/path/to/your/rsh`). Condor provides a special version of `rsh` that it uses to start jobs.
  - You must make sure that your MPICH jobs will be running on machines that will not vacate the job before the job terminates naturally. (This is a limitation of MPICH and the MPI specification.) Unlike PVM, the current MPICH implementation does not support dynamic resource management. That is, processes in the virtual machine may NOT join or leave the computation at any time. If you start an MPI job with *n* nodes, none of these *n* nodes can be preempted by other Condor jobs or the machine's owner.
  - Currently, there is no sophisticated scheduling algorithm for MPI jobs. If you set things up properly, there should not be much of a problem. However, if there are several users trying to run MPI jobs on the same machines, it may be the case that no jobs will run at all and Condor's scheduling will deadlock. Writing a good scheduler for this environment is high on the priority list for Condor version 6.5.
  - "Special" condor_shadow and scondor_starter - You have to use new, special versions of these daemons to run MPI jobs.
  - Shared File System - The machines where you want your MPI job to run must have a shared file system. There is no remote I/O for our MPI support like there is for our Standard Universe jobs.

II.8.6 Data management and integrity

Data access: Condor is designed to run on a pool of machines that are distributively owned. Individuals with computers on their desks can join a Condor pool without an account on any of the other machines in the pool. By agreeing to allow others to use their machines when they are idle, these people gain access to all the idle machines in the pool when they have computations to perform. However, since they have no accounts on the other machines, they can not access the filesystem of the machines where their jobs run. While it is possible that a program could represent its output with a single integer, the return value, this is not generally sufficient. Therefore, Condor jobs must have a way to access a filesystem where they can write output files and read input. Condor uses Remote System Calls to provide access to the filesystem on the machine where a job was submitted. All of the system calls that the job makes as it runs are actually executed on the submitting machine. In this way, any files or directories that the owner of the job can access on his machine can be accessed by the job as it runs, even when running on a machine where the user has no account.

While running on the executing machine, nearly every system call a job performs is caught by Condor. This is done by linking the job against the Condor library, not the standard C library. The Condor library contains function stubs for all the system calls, much like the C library contains function wrappers for the system calls. These stubs send a message back to the shadow, asking it to perform the requested system call. The shadow executes the system call on the submitting machine, takes the result and sends it back to the execution machine. The system call stub inside the Condor executable gets the result, and returns control back to the job (see Figure 15). From the job's point of view, it made a system call, waited for the system to give it an answer, and continued computation. The job has no idea that the system that performed the call was actually the submit machine, instead of the machine where it is running. In this way, all I/O the job performs is done on the submitting machine, not the executing machine. This is the key to Condor's power in overcoming the problems of distributed ownership. Condor users only have access to the filesystem on the machine that jobs are submitted from. Jobs can not access the filesystem on the machine where they execute because any system calls that are made to access the filesystem are simply sent back to the submitting machine and executed there.

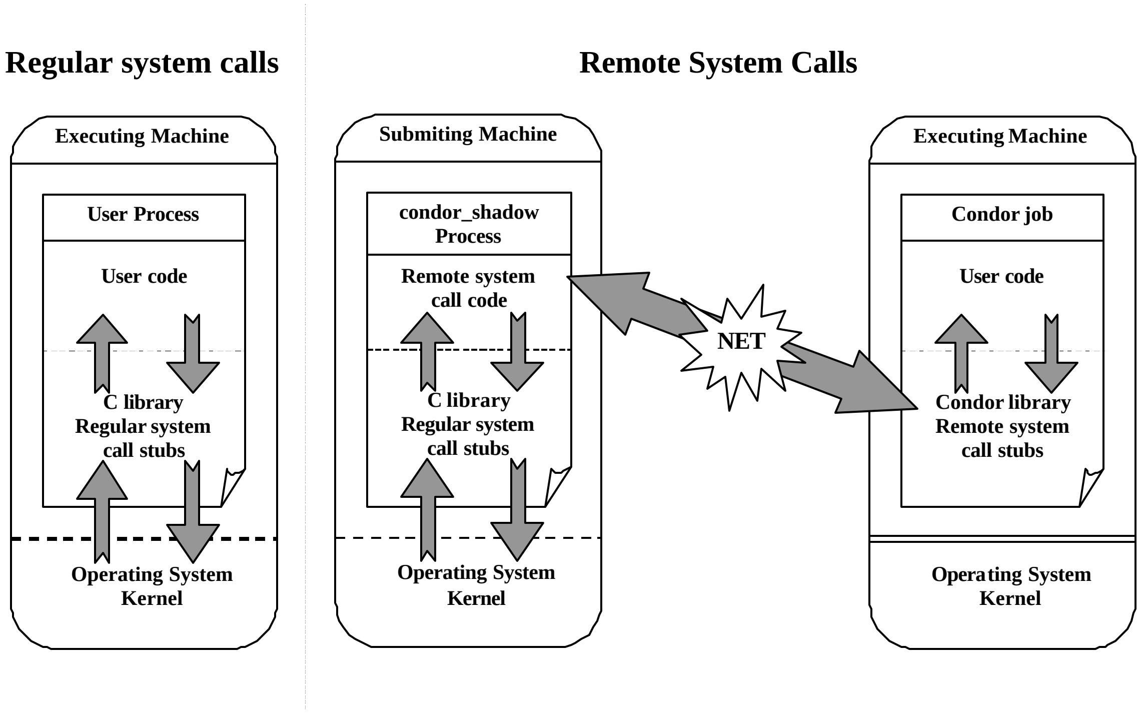


Figure 15: remote system calls versus regular system calls

II.8.7 Data migration

Checkpointing is a key feature which enables job migration without wasting computational results made by the job until the moment when it must release the machine which it is currently occupying.

Checkpointing means saving all the work a job has done up until a given point (i.e. until the moment in time when checkpoint is made) to enable job migration during its execution, from one machine to another, without losing all the results that have been reached so far. In other words checkpointing always takes place when a job must migrate (i.e. the machine on which the job was being executed is no longer available and there is another machine on the network on which the job can continue its execution).

Checkpointing is accomplished by saving all the information about the state of the job to file. This includes all the registers currently in use, a complete memory image, and information about all open file descriptors. This file, called a "checkpoint file", is written to disk. The file can be quite large, since it holds a complete image of the process's virtual memory address space. Normally, the checkpoint file is returned to the machine the job was submitted from. A "checkpoint server" can be installed at a Condor pool, which is a single machine where all checkpoints are stored. An administrator can set up a machine with lots of disk space to be a checkpoint server and then individual machines in the pool do not need any additional disk space to hold the checkpoints of jobs they submit.

Usually, a checkpoint is performed when the machine running a Condor job must be made available to its owner again. Condor performs a checkpoint of the job and puts it back in the job queue to be matched with another idle machine whenever one is available.

Moreover, usually, when performing long-running computations, if a machine crashes, or must be rebooted for an administrative task, all the work that has been done is lost. The job must be restarted from scratch which can mean days, weeks, or even months of computation wasted. With checkpointing, Condor ensures that positive progress is always made on jobs, and that you only loose the computation that has been performed since the last checkpoint. Condor can be configured to periodically checkpoint, you can issue a command to asynchronously checkpoint a job on any given machine, or you can even call a function within your code to perform a checkpoint as it runs.

When a job with a checkpoint file is matched with a machine, Condor starts the job from the state it was in at the time of checkpoint. In this way, the job migrates from one machine to another. As far as the job knows, nothing ever happened, since the entire process is automatic. Checkpointing and process migration even work in a distributively owned computing environment. The job can start up on a machine where its owner has an account, the owner can start to use the machine again, the job will checkpoint, can move to a machine where the owner has no account, and continue to run as though nothing happened. Some long running Condor jobs can end up running on dozens of different machines over their life.

Replication: NONE.

Data storage: NONE.

Data transfer: NONE.

Meta-data management: NONE.

Application support: NONE.

II.8.8 Security

Authentication mechanism: Condor can use the same authentication technology as that used for secure connections in web browsers, i.e., SSL authentication with X.509 certificates. Versions of Condor which include this technology support the GSS (Generic Security Services) authentication method. The primary difference between SSL and GSS is that GSS is a security API which uses the underlying mechanisms of SSL to accomplish such tasks as user authentication, key exchange, and secure communication. The implementation of GSS used in Condor is part of the Globus software.

To use GSS authentication in Condor, the pool administrator(s) must also act as a Certification Authority (CA), as well as maintaining an authorisation list.

The U.S. export restrictions in effect precludes the Condor team from making this capability available to the general public, and so it can only be distributed on a case-by-case basis.

Authorisation mechanisms: Condor's IP/Host-based security allows you to control what machines can join your Condor pool, what machines can find out information about your pool, and what machines within your pool can perform administrative commands. By default, Condor is configured to allow anyone to view or join your pool.

Inside the Condor daemons or tools that use the so called "DaemonCore" ("DaemonCore" is a library that is shared among most of the Condor daemons which provides common functionality), most things are accomplished by sending commands to another Condor daemon. These commands are just an integer to specify which command, followed by any optional information that the protocol requires at that point (such as a ClassAd, capability string, etc). When the daemons start up, they register which commands they are willing to accept, what to do with them when they arrive, and what access level is required for that command. When a command comes in, Condor sees what access level is required, and then checks the IP address of the machine that sent the command and makes sure it passes the various allow/deny settings in the config file for that access level. If permission is granted, the command continues, otherwise, the command is aborted.

Access levels can be set globally to affect all the machines in a pool or locally to affect only a specific machine. The settings for a given machine determine what other hosts can send commands to that machine.

Here is the list of the various access levels that commands within Condor can be registered with:

- **READ**

  Basically, a machine listed with READ permission cannot join a Condor pool - it can only view information about the pool. Machines with READ access can read information from Condor. For example, they can view the status of the pool, see the job queue(s) or view user permissions. READ access does not allow for anything to be changed or jobs to be submitted.

- **WRITE**

  Machines with WRITE access can write information to Condor. Basically it means that it can join a given pool by advertising ClassAd updates to the Central Manager and can communicate with other machines in the pool to submit jobs. In addition, any machine with WRITE access can request the condor_startd daemon to perform a periodic checkpoint on any job it is currently executing (after a periodic checkpoint, the job will continue to execute and the machine will still be claimed by whatever condor_schedd had claimed it). This allows users on the machines where they submitted their jobs to periodically checkpoint their jobs, even if they do not have an account on the remote execute machine.

  Notice that for a machine to join a condor pool, it must have both WRITE and READ permission! WRITE permission is not enough.

- **ADMINISTRATOR**

  Machines with ADMINISTRATOR access have special Condor administrator rights to the pool. This includes things like changing user priorities, turning Condor on and off, asking Condor to reconfigure or restart itself, etc. Typically only a few machine would have ADMINISTRATOR permission – e.g. workstations where the Condor administrators or sysadmins work, or Condor Central Manager.

  IMPORT ANT: In Condor, access levels are host-wide. This means that if you grant some access level to a given machine, any user on that machine will have corresponding rights (including users who can run Condor jobs on that machine). Therefore, you should grant ADMINISTRATOR access carefully.

- **OWNER**

  This level of access is required for commands that the owner of a machine (any local user) should be able to use, in addition to the Condor administrators. For example the condor_vacate command that causes the condor_startd to vacate any running condor job is registered with OWNER permission, so that anyone can issue condor_vacate to the local machine they are logged into.

- **NEGOTIATOR**

  This access level means that the specified command must come from the Central Manager of Condor pool. The commands that have this access level are the ones that tell the condor_schedd to begin negotiating and that tell an available condor_startd that it has been matched to a condor_schedd with jobs to run.

- **CONFIG**

  This access level is required to modify a daemon's configuration. Hosts with this level of access will be able to change any configuration parameters, except those specified in the condor_config.root configuration file. Therefore, this level of host-wide access should only be granted with extreme caution. By default, CONFIG access is denied from all hosts.

For READ, WRITE, ADMINISTRATOR and OWNER access level you can set ALLOW and/or DENY list. If you have ALLOW list, it means "only listed machines are allowed". If you have DENY list, it means "deny these machines". If you have both an ALLOW and a DENY list, it means allow the machines listed in ALLOW except for the machines listed in DENY. If there is no ALLOW list, it means "allow any one" and if there is no DENY list, it means "deny nobody".

Accounting: NONE.

Credential delegation: NONE.

Application support: NONE.

II.8.9 References

1. http://www.cs.wisc.edu/condor/ Condor Team. The Condor High Throughput Computing Environment.
2. Deploying a High Throughput Computing Cluster, *Jim Basney, Miron Livny,* Computer Sciences Department University of Wisconsin-Madison.
3. Improving Goodput by Co-scheduling CPU and Network Capacity, *Jim Basney, Miron Livny*, Computer Sciences Department University of Wisconsin-Madison.
4. Mechanisms for High Throughput Computing, *M.Livny, J. Basney, R. Raman, T. Tannenbaum*, Department of Computer Sciences, University of Wisconsin-Madison.
5. Matchmaking: Distributed Resource Management for High Throughput Computing, *Rajesh Raman, Miron Livny, Marvin Solomon*, University of Wisconsin-Madison (*Proceedings of the Seventh IEEE International Symposium on High Performance Distributed Computing,* July 28-31, 1998).
6. http://www.cs.wisc.edu\condor\overview\index.html *Condor Team,* Overview of the Condor High Throughput Computing System.
7. http://www.cs.wisc.edu\condor\manual\v6.0\index.html *Condor Team,* Condor Version 6.1 manual.
8. http://www.cs.wisc.edu\condor\mw\index.html*Condor Team,* the MW Homepage.

II.9 Load Sharing Facility (LSF)

II.9.1 Introduction

The Load Sharing Facility (LSF) is a suite of application resource management products that schedule, monitor, and analyse the workload for a network of computers. LSF allows control and management of computing resources effectively and efficiently. It comes from Platform Computing Corporation as a commercial product. It assumes the networked queuing model of processing.

LSF consists of a set of products that provide workload management services across the whole cluster, an API that allows access to such services at the procedure level, and a suite of tools or utilities that end users can use to access such services at the command or GUI level.

The software is available on most of the current architectures and operating systems. It also supports single and multiprocessor jobs in shared or distributed environments. The following systems are supported: AIX, Linux, IRIX, Unicos, HP-OS, Solaris, MS WinNT, Tru64.

Platform Computing's LSF for DRM is the key enabling technology in some of the world's largest superclusters, where hundreds or thousands of computers work in concert on some of the most advanced computing projects ever devised. Today, LSF supports everything from Linux to supercomputing operating systems and from inexpensive x86-based workstations to powerful Cray and NEC supercomputers.

LSF allows the creation of compute farms or clusters from networks of workstations and servers. By leveraging all of these resources, organisations can run simulations faster, perform more accurate structural analyses, and maximise the efficiency and productivity of their computing environments. The LSF suite of products also monitors and manages all available resources to keep track of such necessary functions as system capability, access and priority of systems, and efficient workload balance to ensure systems do not get overloaded and jobs are processed as quickly and reliably as possible.

II.9.2 Application description

The user using the JobSheduler feature can provides a brief description of their application. Basically, this is a description of the behaviour of particular jobs rather than their requirements. The job dependency can be mentioned during the submission stage of the user job. This allows the creation of a flow control mechanism for several dependent jobs.

The LSF JobScheduler manages network-wide events, and uses events to drive the scheduling of jobs. The LSF JobScheduler responds to the following types of events:

- time events – points of time that can be used to trigger the scheduling of jobs;
- job events – the starting and completion of other jobs;
- job group events – job group status conditions which can be used to trigger the execution of jobs that depend on them;
- file events – changes in the files residing in accessible file systems, such as the arrival (creation) of a specific file – site-specific occurrences such as a tape mount – defined by the LSF;
- JobScheduler administrators for system exception events – conditions which indicate a problem with the processing of a job

LSF JobScheduler events are used to trigger jobs. As such, events are defined when jobs that are to be triggered by the events are created. There is a difference between a condition and an event. An event is a condition that has been associated with a job as one of its scheduling conditions. A condition that is not associated with a job is not monitored by LSF JobScheduler, and is not considered an event.

Users can specify one or more conditions that will trigger the job's execution. JobScheduler will automatically register events that monitor the specified conditions.

II.9.3 Resource description

LSF provides some mechanisms for resource description. LSF provides a powerful means to describe a heterogeneous cluster in terms of resources. One of the most important decisions LSF makes when scheduling a job is to map a job's resource requirements to resources available on individual hosts.

A computer may be thought of as a collection of resources used to execute programs. In LSF, resources are handled by naming them and tracking information relevant to them. LSF does its scheduling according to an application's resource requirements and resources available on individual hosts. LSF classifies resources in different ways. There are several types of resources. Load indices measure dynamic resource availability, such as a host's CPU load or available swap space. Static resources represent unchanging information, such as the number of CPUs a host has, the host type, and the maximum available swap space. Resources can also be described in terms of where they are located. For example, a shared resource is a resource that is associated with the entire cluster or a subset of hosts within the cluster.

Resources can be classified as follows:

- by Values Numerical Resources - Resources that take numerical values, such as all the load indices, number of processors on a host, or host CPU factor. String Resources - Resources that take string values, such as host type, host model, and host status. Boolean Resources - Resources that denote the availability of specific features.

- by the Way Values Change Dynamic Resources - Resources that change their values dynamically: host status and all the load indices. Static Resources - Resources that do not change their values: all resources except for load indices or host status.

- by Definitions Custom Resources - Resources defined by user sites: external load indices and resources defined by the administrator in the configuration file. Built-In Resources - Resources that are always defined in LSF, such as load indices, number of CPUs, or total swap space.

- by Availability General Resources - Resources that are available on all hosts, such as all the load indices, number of processors on a host, total swap space, or host status. Special Resources - Resources that are only associated with some hosts, such as some shared resources or most Boolean resources.

- by Scope Host-Based Resources - Resources that are not shared among hosts, but are tied to individual hosts, such as swap space, CPU, or memory. An application must run on a particular host to access the resources. Using up memory on one host does not affect the available memory on another host. Shared Resources - Resources that are not associated with individual hosts in the same way, but are "owned" by the entire cluster or a subset of hosts within the cluster, such as floating licenses or shared file systems. An application can access such a resource from any host, which is configured to share it, but doing so affects its value as seen by other hosts. LSF does not contain any built-in shared resources. The LSF administrator must configure all shared resources.

II.9.4 Contract specification methods

The ‘bsub’ command submits a job for batch execution and assigns it a unique numerical job ID. It runs the job on a host that satisfies all requirements of the job, when all conditions on the job, host, queue, and cluster are satisfied. If LSF cannot run all jobs immediately, LSF scheduling policies determine the order of dispatch. Jobs are started and suspended according to the current system load. Users reserve resources and set a time duration for them. This is done with the -R “res_req” option where res_req is a string of maximum length 512 bytes.

A resource requirement string describes the resources a job needs. LSF uses resource requirements to select hosts for remote execution and job execution. A resource requirement string is divided into four sections:

- A selection section. The selection section specifies the criteria for selecting hosts from the system.
- An ordering section. The ordering section indicates how the hosts that meet the selection criteria should be sorted.
- A resource usage section. The resource usage section specifies the expected resource consumption of the task.
- A job spanning section. The job spanning section indicates if a (parallel) batch job should span across multiple hosts.

Depending on the command, one or more of these sections may apply. The syntax of the resource requirement string is as follows:

```
select[selection_string] order[order_string]
rusage[usage_string] span[span_string]
```

The selection string specifies the characteristics a host must have to match the resource requirement. It is a logical expression built from a set of resource names. The selection string is evaluated for each host; if the result is non-zero, then that host is selected.

The order string allows the selected hosts to be sorted according to the values of resources.

The usage string defines the expected resource usage of the task. It is used to specify resource reservations for jobs, or for mapping tasks onto hosts and adjusting the load when running interactive jobs. There is a possibility of defining the duration of the reservation and decay as well. The decay value indicates how the reserved amount should decrease over the duration.

The span string specifies the locality of a parallel job. In LSF version 4.1 only the following two cases are supported:

```
span[hosts = 1]
```

This indicates that all the processors allocated to this job must be on the same host.

```
span[ptile = n]
```

This indicates that only n processors on each host should be allocated to the job regardless of how many processors the host possesses.

If span is omitted, LSF will allocate the required processors for the job from the available set of processors.

II.9.5 Resource management

Grid information services: The Load Information Manager (LIM) is responsible for the cluster information service. LIM is a key server that forms the basis for the concept of an LSF cluster. LIM belongs to LSF Base and provides a single system image called a cluster.

The LIM runs on every server host, and provides cluster configuration information, load information, and host selection services. The LIMs on all hosts co-ordinate in collecting and transmitting load and resource information. Load information is transmitted between LIMs in the form of vectors of load indices.

All LIMs in a cluster must share the same cluster configuration, statically predefined by an administrator, in order to work correctly. To run LIM on a host, the host must be configured as a 'server' host in a file.

A master LIM is elected for each LSF cluster. The master LIM receives load information from all slave LIMs and provides services to all hosts. The slave LIMs periodically check their own load conditions and send a load vector to the master if significant changes in load condition are observed. The minimum load information exchange interval is 15 seconds. Slave LIMs also monitor the status of the master LIM and elect a new master if the original one becomes unavailable.

The load indices monitored at a site can be extended by directing the LIM to invoke and communicate with an External Load Information Manager (ELIM). The ELIM is responsible for collecting load indices not managed by the LIM. These indices are passed on to the LIM by ELIM through a well-defined protocol.

Brokerage services: All work connected with load gathering or job dispatching is centralised and should be maintained by an administrator of the entire cluster configuration. Every time, when a new host is considered to be added to the existing cluster, the manual update of the file entries should be executed.

The LSF daemons' service could be understood as a poor substitute for a brokerage service. There are some other daemons responsible for job maintenance. MBD (Master Batch Daemon) receives job requests from LSF clients and servers and applies scheduling policies to dispatch the jobs to LSF servers in the cluster. MBD is responsible for the overall state of all jobs in the batch system. MBD keeps a file of all transactions performed on jobs throughout their lifecycle. MBD manages queues and schedules jobs on all hosts in the LSF cluster. Each cluster has one MBD, which runs on the master host. MBDs and LIMs can communicate with each other within an entire Multicluster environment.

PIM (Process Information Manager) runs on each LSF server, and is responsible for monitoring all jobs and monitoring every process created for all jobs running on the server. PIM periodically walks the process tree, and accumulates memory and CPU use data. PIM provides run time resource use for all LSF jobs.

Advanced reservation: An advanced time-resource reservation scheme is not supported under LSF. As a poor solution, a calendar mechanism is available. However, it may delay the dispatch of users' jobs. When the required moment comes, the LSF mechanism starts the dispatching procedure, but it doesn't guarantee its placement.

LSF has no cluster-global load index. Indices are maintained for each machine independently. It influences the choice of the remote target cluster by checking the remote queue-level threshold number of the remote pending jobs.

Scheduling mechanisms: The job must be submitted to a queue. The criteria LSF uses for selecting a suitable queue are as follows:

- *Users access restriction.* Queues that do not allow this user to submit jobs are not considered.
- *Host restriction.* If the job explicitly specifies a list of hosts on which the job can be run, then the selected queue must be configured to send jobs to all hosts in the list.
- *Queue status.* Closed queues are not considered.
- *Exclusive execution restriction.* If the job requires exclusive execution, then queues that are not configured to accept exclusive jobs are not considered.
- *Job's requested resources.* These must be within the resource limits of the selected queue.

If multiple queues satisfy the above requirements, then the first queue listed in the candidate queues (predefined by the cluster administrator) that satisfies the requirements is selected. The host election is based on a number of conditions determining whether a host is eligible:

- Host dispatch windows.
- Resource requirements of the job.
- Resource requirements of the queue.
- Host list of the queue.
- Host load levels.
- Job slot limits of the host.

When a job is submitted to LSF, many factors control when and where the job starts to run:

- Active time window of the queue or hosts.
- Resource requirements of the job.
- Availability of eligible hosts.
- Various job slot limits.
- Job dependency conditions.
- Fair share constraints.
- Load conditions.

Jobs are not necessarily dispatched in order of submission. Each queue has a priority number. LSF Batch tries to start jobs from the highest priority queue first. Queue priority is set by an LSF Administrator when the queue is defined. By default, LSF considers jobs for dispatch in the following order:

- For each queue, from highest to lowest priority.
- For each job in the queue, according to FCFS order.
- If any host is eligible to run this job, start the job on the best eligible host, and mark that host ineligible to run any other job until JOB_ACCEPT_INTERVAL dispatch turns have passed.

Jobs can be dispatched out of turn if pre-execution conditions are not met, specific hosts or resources are busy or unavailable, or a user has reached the user job slot limit.

If a fair share scheduling policy has been specified for the queue, or if host partitions have been configured, jobs are dispatched in accordance with these policies instead. To solve diverse problems, LSF allows multiple scheduling policies in the same cluster. LSF has several queuing scheduling policies such as exclusive, pre-emptive, fair share, and hierarchical fair share.

Co-scheduling: LSF supports multiprocessor job execution. The LSF base paradigm is a job slot. Thus, the parallel or distributed jobs are handled using multiple job slots. One time slot could be understood as a space that could be occupied by the one (or several) process (forked sub-processes). LSF doesn't care about the type of multiprocess job: parallel, distributed or simply as a concurrent one-processor set of subprocesses. Everything depends on the job slot set interpretation. Parallel or distributed jobs are given the required number of job slots by the scheduler, and there is no difference between local or remote treating of multiple job slots.

The multiprocessor jobs use job slots in the same way as sequential serial jobs. A user must emphasise the number of required processors (-n option) exactly, or as a minimum and maximum number. Referring to the type of processing (distributed, parallel SMP), the user should care about job slots' spanning over multiple hosts.

Parallel jobs need multiple job slots to be available before they can be dispatched. If the cluster is always busy, a large parallel job could be pending indefinitely. Processor reservation solves this problem by reserving job slots, as they become available, until there are enough reserved job slots to run the big parallel job.

If processor reservation is enabled, and a parallel job cannot be dispatched because there are not enough job slots to satisfy its minimum processor requirements, the job slots that are currently available will be reserved and accumulated.

By default, a reserved job slot is unavailable to any other job. To avoid deadlock situations, in which the system is reserving job slots for multiple parallel jobs and none of them can acquire sufficient resources to start, a parallel job will give up all its reserved job slots if it has not accumulated enough to start within a specified number of dispatch turns.

By default, a reserved job slot cannot be used by another job. To make better use of resources and improve the performance of LSF, backfill scheduling is available. Backfill scheduling allows other jobs to use the reserved job slots, as long as the other jobs will not delay the start of the big parallel job. Back-filling, together with processor reservation, allows large parallel jobs to run while not under-utilising resources. In a busy cluster, processor reservation helps to schedule big parallel jobs sooner. However, by default, reserved processors remain idle until the big job starts. This degrades the performance of LSF because the reserved resources are idle while jobs are waiting in the queue. For backfill scheduling, LSF assumes that a job will run until its run limit expires. Backfill scheduling works most efficiently when all the jobs in the cluster have a run limit.

Since jobs with a shorter run limit have more chance of being scheduled as backfill jobs, users who specify appropriate run limits in a backfill queue will be rewarded by improved turnaround time. Once the big parallel job has reserved sufficient job slots, LSF calculates the start time of the big job, based on the run limits of the jobs currently running in the reserved slots. LSF cannot backfill if the big job is waiting for a job that has no run limit defined.

If LSF can backfill the idle job slots, only jobs with run limits that expire before the start time of the big job will be allowed to use the reserved job slots. LSF cannot backfill with a job that has no run limit. Limitations of back-filling include:

- A job will not have an estimated start time immediately after LSF master daemon is reconfigured.
- You can never pre-empt jobs in a backfill queue (a job in a backfill queue might be running in a reserved job slot, and starting a new job in that slot might delay the start of the big parallel job)

- A backfill queue cannot be pre-empted.
- A pre-emptive queue whose priority is higher than the backfill queue cannot pre-empt the jobs in backfill queue.

A backfill job borrows a job slot that is already taken by another job. The backfill job will not run at the same time as the job that reserved the job slot first. Backfilling can take place even if the job slot limits for a host or processor has been reached. However, backfilling cannot take place if the job slot limits for users or queues have been reached.

Application support: Extensive API is provided both for users and administrators.

II.9.6 Communication methods

Communication models: LSF assumes client-server model.

Communication protocols: LSF uses standard Internet protocols for communication.

Quality of Service: There is no QoS supported under LSF.

Fault tolerance: LSF is designed to continue operating even if some of the hosts in the cluster are unavailable. One host in the cluster acts as the master, but if the master host becomes unavailable another host takes over. LSF services are available as long as there is one available host in the cluster.

LSF can tolerate the failure of any host or group of hosts in the cluster. When a host crashes, all jobs running on that host are lost. No other pending or running jobs are affected. Important jobs can be submitted to LSF with an option to automatically restart if the job is lost because of a host failure.

The LSF master host is chosen dynamically. If the current master host becomes unavailable, another host takes over automatically. The master host selection is based on the order in which hosts are listed in the configuration file. If the first host in the file is available, that host acts as the master. If the first host is unavailable, the second host takes over, and so on. LSF might be unavailable for a few minutes while hosts are waiting to be contacted by the new master.

Running jobs are managed by a slave batch daemon (SBD) on each batch server host. When the new master batch daemon (MBD) starts up it polls the SBDs on each host and finds the current status of its jobs. If SBD fails but the host is still running, jobs running on the host are not lost. When SBD is restarted it regains control of all jobs running on the host.

If the cluster is partitioned by a network failure, a master LIM takes over on each side of the partition. Interactive load-sharing remains available, as long as each host still has access to the LSF executables.

Fault tolerance in LSF depends on the event log file, lsb.events, which is kept on the primary file server. Every event in the system is logged in this file, including all job submissions and job and host state changes. If the master host becomes unavailable, a new master is chosen by the LIMs. SBD on the new master starts a new MBD. The new MBD reads the lsb.events file to recover the state of the system.

For sites not wanting to rely solely on a central file server for recovery information, LSF can be configured to maintain a duplicate event log by keeping a replica of lsb.events. The replica is stored on the file server, and used if the primary copy is unavailable. When using LSF's duplicate event log function, the primary event log is stored on the first master host, and re-synchronised with the replicated copy when the host recovers.

If the network is partitioned, only one of the partitions can access lsb.events, so batch services are only available on one side of the partition. A lock file is used to guarantee that only one MBD is running in the cluster.

If an LSF server host fails, jobs running on that host are lost. No other jobs are affected. Jobs can be submitted so that they are automatically rerun from the beginning, or restarted from a checkpoint on another host if they are lost because of a host failure.

If all of the hosts in a cluster go down, all running jobs are lost. When a host comes back up and takes over as master, it reads the lsb.events file to get the state of all batch jobs. Jobs that were running when the systems went down are assumed to have exited, and email is sent to the submitting user. Pending jobs remain in their queues, and are scheduled, as hosts become available.

To provide job fault tolerance, checkpoints are taken at regular intervals (periodically) during the job's execution. If the job stops running for some reason, it can be restarted from its last checkpoint and not waste the efforts to get it to its current stage. Similarly, if the host on which a job is running fails, the job can resume execution from the last checkpoint. Jobs can be migrated when a host fails or when a host becomes unavailable due to load.

Services supporting collaboration and remote instrument control: There is no support for distributed collaboration and remote instrument control.

Application support: Extensive API is provided both for users and administrators.

II.9.7 Data management and integrity

Data access: LSF is designed for networks where all hosts have shared file systems, and files have the same names on all hosts. LSF supports the Network File System (NFS), the Andrew File System (AFS), and DCE's Distributed File System (DFS). NFS file systems can be mounted permanently or on demand using 'automount'.

LSF includes support for copying user data to the execution host before running a job, and for copying results back after the job executes. In networks where the file systems are not shared, this can be used to give remote jobs access to local data.

By default, LSF assumes uniform user accounts throughout the cluster. This means that a job will be executed on any host with exactly the same user ID and user login name.

The LSF JobScheduler has a mechanism to allow user account mapping across dissimilar name spaces. Account mapping can be done at the individual user level. Individual users of the LSF cluster can set up their own account mapping by setting up a .lsfhosts file in their home directories. The LSF administrator can disable user account mapping.

Data migration: Migration, in the LSF nomenclature, is the process of moving a checkpointable or re-runnable job from one host to another host. Checkpointing enables a migrating job to make progress by restarting it from its last checkpoint. Re-runnable non-checkpointable jobs are restarted from the beginning. LSF provides the ability to manually migrate jobs from the command line and automatically through configuration. When a job is migrated, LSF performs the following actions:

- Stops the job if it is running.
- Checkpoints the job if it is checkpointable.
- Kills the job on the current host.
- Restarts or reruns the job on the next available host, bypassing all pending jobs.

To migrate a checkpointable job to another host, both hosts must:

- be binary compatible;
- run the same version of the operating system. Unpredictable results may occur if both hosts are not running exactly the same version of the OS;
- have access to the executable;
- have access to all open files (LSF must locate them with an absolute path name);
- have access to the checkpoint file.

Replication: No data replication mechanisms exist.

Data storage: There is no storage internally supported by the LSF software.

Data transfer: Data transfer is provided by the rcp based application. The user or administrator should enable access to the user's input files. LSF is usually used in networks with shared file space. When shared file space is not available, the bsub -f command should be used to instruct LSF to copy needed files to the execution host before running the job, and to copy result files back to the submission host after the job completes.

When LSF runs a job, it attempts to run the job in the directory where the bsub command was invoked. If the execution directory is under the user's home directory, SBD looks for the path relative to the user's home directory. This handles some common configurations, such as cross-mounting user home directories with the /net automount option.

If the directory is not available on the execution host, the job is run in /tmp. Any files created by the batch job, including the standard output and error files created by the -o and -e options to bsub, are left on the execution host.

LSF provides support for moving user data from the submission host to the execution host before executing a batch job, and from the execution host back to the submitting host after the job completes. The file operations are specified with the -f option to bsub.

LSF uses the lsrcp command to transfer files. lsrcp contacts RES on the remote host to perform file transfer. If RES is not available, the UNIX rcp command is used.

The main limitations of lsrcp are as follows. Because LSF client hosts do not run RES, jobs that are submitted from client hosts should only specify bsub -f if rcp is allowed. Permissions for rcp must be set up if account mapping is used. File transfer using lscrp is not supported in the following contexts:

- If LSF account mapping is used; lsrcp fails when running under a different user account.
- LSF client hosts do not run RES, so lsrcp cannot contact RES on the submission host.

Meta data management: There is no meta data management in LSF.

Application support: Limited API is provided both for users and administrators.

II.9.8 Security

To enable LSF users to execute commands remotely, one of the following methods should be chosen during installation process:

- External authentication (eauth)
- Privileged ports (setuid)
- Identification daemon (identd)

Authentication mechanisms: Optionally, it is possible for users to choose to write their own eauth executable that uses some site-specific authentication method such as Kerberos4 or DCE client authentication using the GSSAPI. Examples of these can be found in the examples/krb and examples/dce directories of LSF directory.

User authentication data is passed to eauth -s via its standard input. The standard input stream to eauth has the following format:

```
uid gid user_name client_addr client_port user_auth_data_len  user_auth_data
```

where:

`uid` is the user ID in ASCII of the client user

`gid` is the group ID in ASCII of the client user

`user_name` is the user name of the client user

`client_addr` is the host address of the client host in ASCII dot notation

`client_port` is the port number from where the client request is made

`user_auth_data_len` is the length of the external authentication data in ASCII passed from the client

`user_auth_data` is the external user authentication data passed from the client

By default, eauth uses an internal key to encrypt authentication data. To use an external key to improve security, the LSF_EAUTH_KEY parameter should be set. The default eauth program is installed without setuid permission. When the LSF_EAUTH_KEY parameter used, eauth must be setuid.

Privileged Ports Authentication (setuid) is the mechanism most UNIX remote utilities use. The LSF commands must be installed as setuid programs and owned by root. If a load-sharing program is owned by root and has the setuid bit set, the LSF API functions use a privileged port to communicate with LSF servers, and the servers accept the user ID supplied by the caller. This is the same user authentication mechanism as used by the UNIX rlogin and rsh commands.

When a setuid application calls the LSLIB initialisation routine, a number of privileged ports are allocated for remote connections to LSF servers. The effective user ID then reverts to the real user ID. As a consequence of this, the number of remote connections is limited.

An LSF utility reuses the connection to RES for all remote task executions on that host, so the number of privileged ports is only a limitation on the number of remote hosts that can be used by a single application, not on the number of remote tasks. If a file server does not permit setuid permission, the LSF administrator should install the LSF binaries on a file system that does allow setuid. Privileged ports authentication is not available on Windows NT.

LSF also supports authentication using the RFC 931 or RFC 1413 identification protocols. Under these protocols, user commands do not need to be installed as setuid programs owned by root. The Identd daemon must be installed instead. This daemon is available in the public domain.

The RFC 1413 and RFC 931 protocols use an identification daemon running on each client host. RFC 1413 is a more recent standard than RFC 931. LSF is compatible with both. Using an identification daemon incurs more overheads, but removes the need for LSF applications to allocate privileged ports. When the site cannot install programs owned by root with the setuid bit set, or if software developers create new load-sharing applications using LSLIB, the identification daemon should be installed. Implementations of RFC 931 or RFC 1413 such as pidentd or authd are available in the public domain. Identification daemon authentication is not available on Windows NT.

LSF uses the LSF_AUTH environmental parameter to determine which type of authentication to use. External user authentication is used if LSF_AUTH is defined to be eauth. This is the default. In this case, LSF will run the external executable eauth in the LSF_SERVERDIR directory to perform the authentication.

If a load-sharing application is not setuid to root, library functions use a non-privileged port. If the LSF_AUTH parameter is not set in the configuration file lsf.conf, the connection is rejected. If LSF_AUTH is defined to be ident, then RES on the remote host (or MBD in the case of a bsub command), contacts the identification daemon on the local host to verify the user ID. The identification daemon looks directly into the kernel to make sure the network port number being used is attached to a program being run by the specified user.

LSF allows both the setuid and identification daemon methods to be in effect simultaneously. If the effective user ID of a load-sharing application is root, then a privileged port number is used in contacting RES. RES always accepts requests from a privileged port on a known host even if LSF_AUTH is defined to be ident. If the effective user ID of the application is not root, and the LSF_AUTH parameter is defined to be ident, then a normal port number is used and RES tries to contact the identification daemon to verify the user's identity.

All authentication methods supported by LSF depend on the security of the root account on all hosts in the cluster. If a user can get access to the root account, they can subvert any of the authentication methods. LSF only accepts job execution requests that originate from hosts within the LSF cluster, so the identification daemon can be trusted.

The system environment variable LSF_ENVDIR is used by LSF to obtain the location of configuration file lsf.conf, which points to the LSF configuration files. Any user who can modify system environment variables can modify LSF_ENVDIR to point to their own configuration.

Authorisation mechanisms: Authorisation mechanisms are applied to the users' entry points to the LSF cluster. Usually, this is a standard operating system mechanism, e.g. password, or some other external feature supported by e.g. Kerberos certificates.

Accounting: The LSF Analyzer processes historical workload data from the LSF system to produce graphical and tabular reports about a cluster. This extra, chargeable module is available from Platform Computing. Workload data includes information on:

- Batch jobs
- System metrics
- Resource usage

LSF Analyzer provides system administrators and managers with the information to make the informed scheduling and capacity planning decisions required to fully utilise the power delivered by LSF. LSF Analyzer can also be used to perform chargeback accounting. The LSF Analyzer contains the following components:

- *Data Collection:* Provides the infrastructure to collect and manage LSF cluster data. Includes data collection agents, data storage and management, and data export to ASCII format.
- *Reports:* A report generation and viewing tool producing LSF-specific reports so that LSF system data can be analysed immediately. You have the option of generating reports on Windows NT, and of viewing reports on Windows 95, Windows 98, Windows NT, or with a web browser. Reports can also be viewed over the web with LSF Analyzer Reports WebClient.
- *Reports Designer:* A tool to create your own custom reports. Easy data selection, WYSIWYG report layout, and sophisticated formatting tools to design and create high-quality reports in a variety of formats are all available.

LSF Analyzer provides tools to perform cluster-wide workload analysis and to display data in a variety of formats.

- Analyse:
  - The number of jobs processed by the system.
  - Job resource usage (memory usage, CPU usage, and swap usage).
  - Load indices and shared resources.
  - Throughput.
  - Turnaround time.
  - Wait time.
  - Run time.
  - Pending reasons.

- Identify trends for:
  - LSF system hosts.
  - Users.
  - Queues.
  - Applications.
  - Projects.
  - Pending reasons.
  - Resource bottlenecks.

LSF Analyser supports charge-back report creation based on license and resource usage, classified by project and user. LSF Analyser generates reports directly in the desired format or will export data to:

- HTML.
- Standard spreadsheet and data analysis tool formats such as Microsoft Excel, Lotus 1-2-3, command and tab-separated values.
- RTF.
- Microsoft Word for Windows.
- PostScript.

LSF administrator can automate data analysis with:

- LSF-specific reports to immediately generate reports and create custom reports more easily Batch scheduling of report processing.
- Customise Data Viewing.

The information retrieved with LSF Analyzer can prove valuable for upgrade planning, performance tuning, and management reporting. The presentation of data by LSF Analyzer is highly customisable. Data can be sorted by:

- Time period.
- Host.
- Host group.
- Host model.
- Host type.
- Queue.
- User.
- User group.
- Project.
- Job name for application types.
- Pending reason.
- Load index.
- Job status.

The appearance of the final report is also customisable:

- Tables, bar charts and graphs can be created and shared with reports across the intranet.
- Specify, how often, and when reports are generated, and on which hosts.
- The format in which the reports are configured and generated may be exported.

Credential delegation: The credential delegation is not supported under LSF.

Application support: Extensive API is provided both for users and administrators.

II.9.9 References

1. LSF Administrators Guide 4.0, Platform Computing, Canada,2000 2. S. Zhou, X. Zheng, J. Wang, P. Delisle, UTOPIA: A Load Sharing Facility for Large, Heterogeneous Distributed Computer Systems, University of Toronto, 1993.
2. http://www.platform.com

II.10 Internet Computing: Entropia and Seti@Home

II.10.1 Introduction

Over the past few years, there have been significant changes in the way we think about computers. We are no longer bound by large and cumbersome applications on the desktop. On the other hand - science, although almost totally supported by public funds, has traditionally been carried out in laboratories and observatories not open to the general public. Today, the public is uniquely involved in at least one real scientific project (SETI). Imaginative and innovative use of Internet and GRID computing facilities can give researchers wide access to a different kind of scientific resource and laboratory. We could connect to remote hosts and execute our applications, we could even remote control, steer and browse the results of our research and easily exchange information with other people or scientists. Moreover, we are witnessing the biggest attempt ever to harness the combined computing power of hundreds of thousands of desktops, laptops, and other small computers, a partnership of radio astronomers, space scientists, and scientific laboratories is proposing to distribute special screen savers to ordinary computer users around the world. Individuals will download a screensaver programs that will not only provide the usual attractive graphics when their computer is idle, but will also perform sophisticated analysis and solve many of the world's great problems using the host computer.

As a part of the global High Performance Internet Computing Grid, the recycled idle resources of peoples PCs can collectively join with millions of others on an Internet adventure of discovery to:

- help research new treatments and possible cures for cancer, AIDS, Alzheimer's and other diseases;
- explore new horizons in human knowledge and make new discoveries;
- deepen our ability to predict changes in complex systems;
- protect the environment for future generations;
- searches for intelligent life out in the universe;
- and many more...

The goal of many High Performance Internet Computing projects is to do good science and to do it in a way that engages and excites the general public. Today, there is so much data that it would take a huge supercomputer to analyse it all. But sending little packets of data to lots of small personal computers accomplishes the same thing. Ultimately, the scientific community can only profit from the better public understanding of the scientific enterprise. Already, several companies have been formed to explore commercial applications of such techniques to (amongst others) movie special effects and economic projections.

Popular current Internet computing projects include:

- SETI@home: www.setiathome.ssl.berkeley.edu
- Folding@home: www.stanford.edu/group/pandegroup/Cosm
- Compute-against-Cancer: www.parabon.com/cac.jsp
- Fight AIDS@home: www.fightaidsathome.org
- Great Internet Mersenne Prime Search: www.mersenne.org
- Casino 21: Climate simulation: www.climate-dynamics.rl.ac.uk

There are also many Internet computing companies such as:

- Entropia: www.entropia.com
- United Devices: www.uniteddevices.com
- Parabon: www.parabon.com
- Popular Power: www.popularpower.com

Of the many companies and projects related to Internet Computing, the two most famous and successful for scientific research are probably *Entropia* and SETI@home. *Entropia* combines personal computing resources into an engine for positive world change, and in so doing creates an entire series of new industries based upon on-demand scalable computing services built from the untapped resources of the Internet itself. By creating a powerful service infrastructure and brokering the demand and supply of many thousands of Internet-connected computers, *Entropia* supports groundbreaking research and paves the way for software integration and for software vendors to enable and enhance the capabilities of their products and services. Founded in 1997, *Entropia* applies the collective power of its members' computers to help solve crucial problems, explore new frontiers of knowledge, and to open the door to new avenues of electronic commerce. Today, *Entropia* is a service company that aggregates the idle computing resources of PCs all over the world to sell a portion of that capacity to commercial customers, and gives the rest away to world-renowned non-profit research organisations.

Starting in the late 1950s, researchers have been performing progressively more intensive investigations in many fields, but these have been limited by the technologies available at the time. As radio frequency technologies have become more efficient and computers have become faster, the techniques have become more powerful and more sensitive. *SETI@home,* is probably Capacity Computings most famous project. *SETI@home* is managed by a group of researchers at the Space Sciences Laboratory of the University of California, Berkeley. The project is the first attempt to use large-scale distributed computing to perform an intensive search for artificial radio signals. SETI, or the Search for Extraterrestrial Intelligence, is a scientific effort to determine if there is intelligent life out in the universe. There are many methods that SETI scientific teams use to search for extraterrestrial intelligence. Many of these search the billions of radio frequencies that flood the universe, looking for another civilisation that might be transmitting a radio signal. Other SETI teams search by looking for signals in pulses of light emanating from the stars. *SETI@home* is one of the SETI projects that searches for extraterrestrial life. *SETI@home* allows anyone with a computer and an Internet connection to take part in the search.

II.10.2 Resource and application description

Amongst the most important issues in Internet Computing are the methods and mechanisms that allow users to describe their computing applications and resources in general terms. Resource description combines a variety of web-based meta-data activities including specification of technical details, attributes and other parameters. Such descriptions provide a uniform and simultaneous model for accessing heterogeneous and available resources. However, there may be many different applications in a given environment. Different applications and user requirements need similar mechanisms to describe resource requirements. Ultimately we need to be able to describe a complex layered multi-component architecture, and to be able to dissociate users and services from particular resources within that architecture.

If we think about existing application and resources description in High Performance Internet Computing terms, there is little difference between these mechanisms and those described in the previous paragraph. However, before using such mechanisms in the real world, we should be aware of the basic constraints which will affect the way in which the *Entropia* and *SETI@home* projects are implemented. The idea behind Capacity Computing is to take advantage of the unused processing cycles of personal computers. The way this would work is as follows: an interested computer owner will download free software from appropriate web pages. Then, when their computer is idle, this software will download a set of data for analysis. The results of this analysis are ultimately sent back to the server and combined with the processed data from the many thousands of other participants. Below we can see the general architecture that is being used the *Entropia* and *SETI@home* projects.

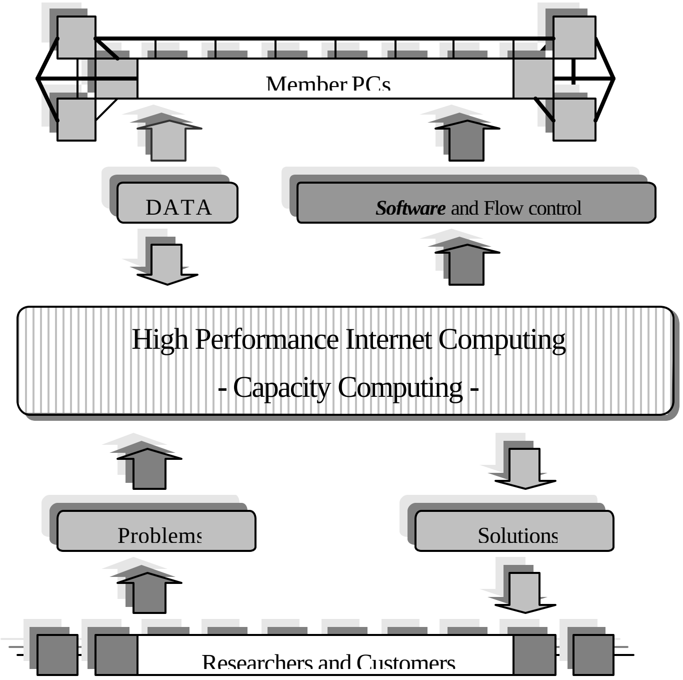

Member PCs
DATA
Software and Flow control
High Performance Internet Computing
- Capacity Computing -
Problems
Solutions
Researchers and Customers

Figure 16: the Internet computing paradigm

We see that such an architecture is composed of two parts: member PCs and researchers (or customers). Member PCs provide idle CPU cycles and receive programs and blocks of data to analyse. Researchers (or customers) generate problems, send data to and collect results from member PCs which have performed the calculations. This is a continuous process and is the route by which we obtain solutions to predefined problems which are distributed in this way. Obviously researchers should know where (and when) to send data to, and from where to receive results. Effectively, the problem amounts to preparing work-units of data (along with the appropriate software). These work-units must be capable of being worked on separately, independently and in parallel - none of the pieces depends on any of the other pieces. Work-units are then downloaded to machines whose users have decided to accept work-units from that particular project, and the calculations performed (subject also to time limits users may have imposed on their 'donation' of resource to the project). After a few hours or days of computation, depending on how high-powered their computers are, the results from the work unit are uploaded to reflect the contributed computing time, then a new work unit is downloaded to start the cycle again. As this is all that is really needed, Capacity Computing has relatively simple client application mechanisms that are responsible for automatically communicating with the servers (e.g.: *Entropia*, *SETI@home*) and for receiving assignments to perform applications tasks. Note that in general, participants can not submit and execute their own applications, and they have to choose from the available projects. Thus, Capacity Computing does not need complex application and resource descriptions but the basic client-server mechanisms, which are hidden from end-users inside software, in order to exchange information about the current member PCs and servers states. Both *Entropia* and *SETI@home* provide members with client software, which can be downloaded from heir homepages. In the case of *Entropia*, a free program is available that activates and recycles a PC's idle resources and quietly and unobtrusively utilises the unused cycles on behalf of the remote project. *Entropia* safely isolates distributed computing applications, avoiding any unexpected and unwanted interactions with desktop applications. Distributed applications run solely in the background. Desktop applications maintain priority access to system resources with no competition from distributed applications. The *Entropia* client checks for the availability of processor cycles every second, even between keystrokes, to ensure that there is never a disruption of service to primary desktop applications. Behind primary applications *Entropia* analyses data from the project for which the desktop machine was enrolled, and frequently notifies the user about the actual state of the analysis. In the case of *SETI@home*, the client program runs on each client computer and looks and behaves like a screensaver. In fact, the *SETI@home* screensaver is a complicated piece of scientific analytical software. It performs a large set of mathematical operations on the data that has been downloaded from the Berkeley SETI program. It runs only when the machine is idle, and the user can choose to display any one of several different colourful and dynamic "visualisations" of the SETI process. The standard screensaver modes include a map of the world showing the location of all machines currently participating in the project, a map of the sky showing what areas have been covered by the survey and the location of the patch of sky currently being analysed and colourful, changing patterns that correspond to the Fourier transforms currently being undertaken, and "straight" graphs showing results of the currently evolving data analysis.

II.10.3 Contract specification methods

There is no information about the contract specification. Due to the fact that the idea behind Capacity Computing is to take advantage of the unused processing cycles of personal computers, end-users (members) can define the availability of their machines. Thus, the members provide their resources for Internet Computing client applications according to their needs. In other words, the availability of machines, and as a consequence, the analyses performed on a given machine are controlled by the end-user and not by the application.

II.10.4 Resource management

Nowadays, Internet Computing is just a special case of something much more powerful: the ability for communities to share resources as they tackle common goals. Science today is increasingly collaborative and multidisciplinary, and it is not unusual for teams to span institutions, states, countries and continents. So-called Grid technologies seek to make this possible, by providing the protocols, services and software development kits needed to enable flexible, controlled resource sharing on a large scale. In order to integrate the kind of totally different resources (Grid technologies), dedicated information services and brokerage services are currently being developed as well as advanced reservation and scheduling mechanisms. There is a basic difference between the Grid and Internet Computing. In the case of Internet Computing the focus is on dedicated software in client-server architecture, so neither advanced information services, nor special scheduling mechanisms are required.

II.10.5 Communication methods

In fact, High Performance Internet Computing is a set of technologies that enable the direct exchange of services or data between computers. In that computing environment: servers, desktops and notebook PCs in a network become peers that contribute all or part of their resources - such as processing power or storage - to the enterprise. This type of architecture transforms client computers from mere consumers of services to service providers as well, and uses their collective computing power and storage capacity to perform data-intensive calculations or simulations over a network without overloading the network infrastructure.

Obviously, when client machines communicate with a server, then an appropriate application running on the server should monitor and store information about client resources: type of CPU, amount of memory, connection bandwidth and more. But this kind information is rather more useful for statistics than for immediate resource management or scheduling. A third server system contains the work unit storage, handling distribution of work units and storage of returned results. This server supports two types of communication. The first requests a work unit. The response to this request is to select a work unit chosen from those stored in temporary storage. Priority goes to those units that have not previously been sent, or those that were sent but for which no results were received. In the second type of communication, the client program returns a result to the server. The server extracts the volunteer machines identity from the results and stores the results in the science database and updates the volunteer's statistics in the user database. The reply to this request includes the volunteer's statistics so that the client program can display them.

Communication models: *Entropia* uses the peer-to-peer concept. Peer-to-peer software enables individual computers to combine processing power and to communicate with one another over the Internet without going through a central server. Periodically client applications work is automatically synchronised by the servers, using an ordinary Internet connection. A similar concept is used in *SETI@home* project, so an appropriate mechanism rapidly ports applications to run safely on the distributed computing Grid. In most cases, Windows executables can be ported to the *SETI@home* application without any source code required.

Considering projects based on the types of architecture described above, one of the most important questions to be asked relates to what kind of communications methods should be used. We know that *Entropia* and *SETI@home* have to exchange information and send data between PC's. The potential of existing web-based services is clear: standards for the transmission of information simplify things enormously. TCP/IP, HTTP, and HTML, the three major open standards on which the Internet is based, transformed the world in the 1990s. Nearly universal support of these standards by vendors and businesses has made communication over the Internet relatively effortless. For that reason, the huge amount of PC owners with home, business or school access to the Internet gives us a great opportunity today to build large distributed applications. But we should be aware that it is not easy to build as efficient and scalable distributing capacity computing services which are as efficient and scalable as *Entropia* and *SETI@home.*

In the case of *SETI@home,* connections are made only when transferring data. This occurs only when the screen saver has finished analysing the work-unit and wants to send back the results (and get another work-unit.) Users can control connections, optionally transferring data automatically as soon as it is done with the current work-unit or manually at some appropriate time they. Due to the relatively small amounts of data involved, transmission takes only few minutes with most common modems and disconnection happens immediately after all data has been transferred. A similar mechanism is implemented in *Entropia* client software.

Communication protocols: Today, standard TCP/IP protocols, and the experience of real capacity computing projects are very useful when building new high performance and Internet distributed scientific applications which could be supported by users around the world. However, due to the nature of the Internet and of TCP/IP routing protocols, delivery time can't be guaranteed, nor can quality, and for critical systems proprietary connection methods, data distribution methods, and other algorithms are needed.

Quality of Service: Not yet supported.

Services supporting collaboration and remote instrument control: Collaboration is generally made among the members in order to improve the number of resources. Internet Computing users can form "groups"—for example, students in a school, or employees of a company. Groups can compete to analyse the most data, or they can meet and discuss the project via the web.

II.10.6 Data management and integrity (data access, data migration, replication, data storage, data transfer)

High Performance Internet Computing has limitations. Distributed computing works best with applications and instances of problems that can be broken down into individual, independent computations. Highly interconnected modelling - such as predicting the weather, or simulating nuclear explosions - will always require a real supercomputer. The linchpin of the whole scheme remains individual PC users, who must be persuaded into downloading a piece of software, and then allowing data from a central computer to be downloaded and processed on their machines when not in use. This distribution of processing power across the Internet is a stunning concept, and projects like *SETI@home* and *Entropia* prove that it works better than anyone might have expected. We are not looking at the end of the centralised server, at least not yet, because Internet Computing essentially involves building new infrastructures with implications for security and reliability.

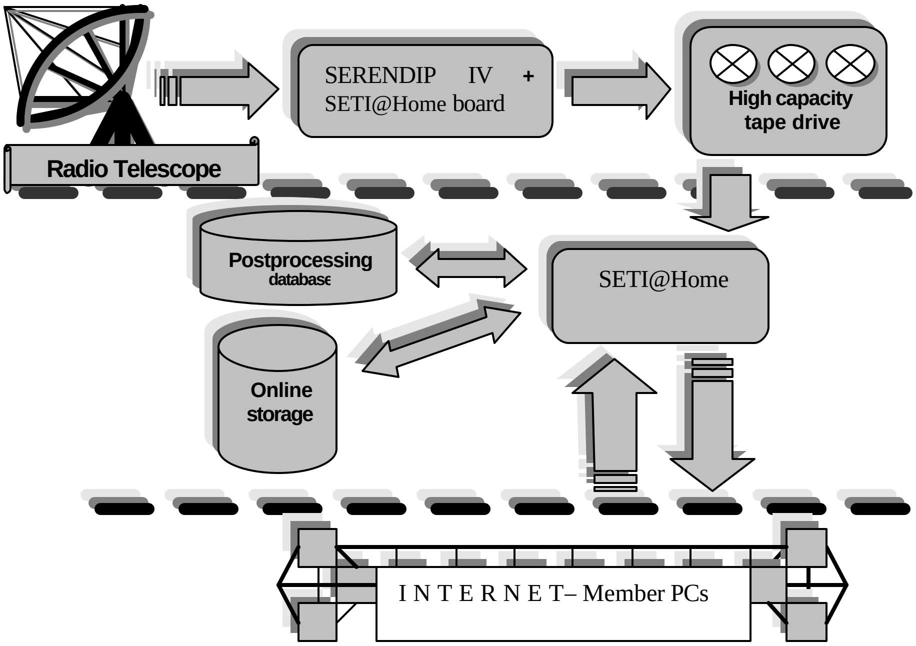


Figure 17: SETI@Home

Although there are many constraints and problems concerning High Performance Internet Computing, and we need to know more about additional mechanisms like data management and integrity when tackling large scientific problems with many thousands of independent units of data distributed around the world, Internet Computing does work. Let us examine how it works in a real project. In the figure below we see in some detail the data management mechanism which is applied in SETI@home.

Packets of data received at the *SETI@home* server over the Radio signals are sent to Berkeley's Project SERENDIP, and then scattered through the Internet to people around the world to analyse. The results of this analysis are ultimately sent back to the SERENDIP team, combined with the processed data from the many thousands of other *SETI@home* participants, and used to help in the search for extraterrestrial signals. *SETI@home* keeps track of the work-units in Berkeley with a large database. When the work-units are returned, they are merged back into the database and marked "done." *SETI@home* computers look for a new work-unit for participants to process and send it out, marking it "in progress" in the database. *SETI@home* send out each work unit multiple times in order to make sure that the data is processed correctly (for example: a computer is not able to complete a work unit, or a computer crashes and loses the results).

Data are recorded on high-density tapes at the world's biggest telescope dish — the 1,000-foot Arecibo Observatory in Puerto Rico. This data fills about one 35 Gbyte DLT tape per day. At Arecibo, *SETI@Home* also tape-record a small portion of SERENDIP's total bandwidth. Because Arecibo does not have a high bandwidth Internet connection, the data tape must go by conventional mail to Berkeley. The data is then divided into 0.25 Mbyte chunks (called "work-units"). *SETI@home* looks at 2.5 MHz of data, centred at 1420 MHz. This is still too broad a portion of the spectrum to send to people for analysis, so *SETI@Home* breaks this spectrum space up into 256 pieces, each 10 kHz wide. This is done with a program called the "splitter". These 10 kHz pieces are now more manageable in size.

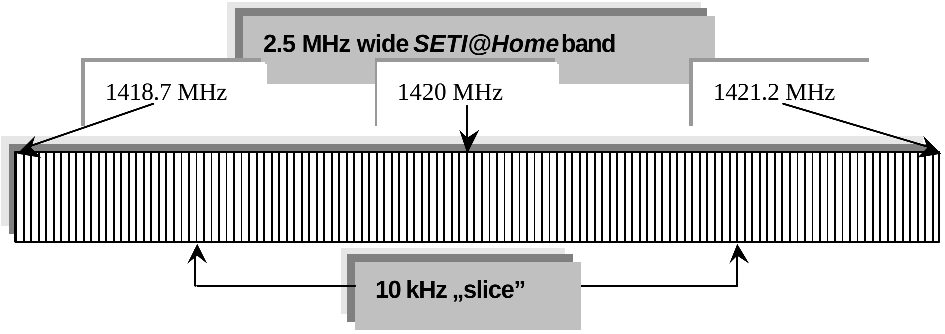


Figure 18: the SETI@Home work units

To record signals up to 10 kHz, the equipment has to be capable of recording the data at a minimum of 20,000 bits per second (kbps). *SETI@Home* sends people about 107 seconds of this 10 kHz (20kbps) data. 100 seconds times 20,000 bits equals 2,000,000 bits, or about 0.25 megabyte given that there are 8 bits per byte. This 0.25 megabytes of data constitutes a "work-unit." Additionally, *SETI@Home* sends members lots of additional information about the work-unit, so the total comes out to about 340 kbytes of data.

The limitations of this system are that only a 2.5 MHz piece of the observed spectrum will be analysed by *SETI@home*, and of course the data processing does not occur "real time" so that interesting signals must be followed up at a later date. The tremendous advantage of this scheme is that it permits looking for a variety of signal types that the current SERENDIP processing cannot yet uncover, and many thousands of interested people will become active participants in SETI.

II.10.7 Security

Security is a major concern. With distributed computing, laboratories will effectively be exporting data to anonymous desktops outside the firewall, and obviously researchers can lose control over their data or just lose it. For that reason if we think about real scientific problems with data distributed across the world, authentication, authorisation and accounting services are needed for users, as well as confidentiality, and integrity for data transfers.

In the *Entropia* project, participants are automatically enrolled in all available non-profit projects by default when they join *Entropia*. The list of currently available projects is listed on the *Entropia* projects web pages. Users may select or change which of these they wish to support on the members web site at any time. When an individual first joins *Entropia*, a membership account for their PC is set up and the PC is automatically added to it. Any number of additional PCs can be later added to the same *Entropia* membership. Each additional PC enrolled in an *Entropia* membership increases the cumulative tasks and time needed to track that members activity / profile prior to display on the members web site. Additionally, users may create or join a team. The statistics of each *Entropia* members in a team are pooled to yield the statistics for that team. Each user may join, create or change teams at any time. Similar, but rather simpler authentication and authorisation mechanisms are implemented in *SETI@home*.

Entropia and SETI@home client applications have not access to any files on member's personal computers and have no knowledge of what people do on their computers. Both try to provide airtight protection of application data by fully encrypting it and isolating it completely from the PC user. Additionally, all network communications are fully encrypted as well. Only the *Entropia* and *SETI@home* network servers have access to computational results.

Authentication, Authorisation and Accounting mechanisms: There are two basic mechanisms used for accounting and members authorisation in Internet Computing:

- *User Account Area*. In order to protect an account from modification by others, both e mail address and password are required for access. If the member loses this information, this cannot be obtained from the Internet Computing project. The member has to create a new account with a valid email address in order to access any future credits Access to the earlier account is lost.
- *User profiles*. These provide a way for individuals to share backgrounds and opinions with the community.

Credential delegation: Not supported.

Fault tolerance: A job is not complete when the client has finished its work, the job. In the case of *SETI@home* the client program returns a few potential signals per work unit. Errors made in the processing computers generate some of these signals. Typically, processors, memory, and disk systems are fairly reliable. However, *SETI@home* uses thousands of years of CPU time per day, magnifying even low error rates to a relatively high absolute level. Even if undetected errors occur only on average every 10^{18} machine instructions, SETI@home would see several per day. Additional errors can be introduced in result transmission because of broken connections or malfunctioning HTTP proxies. To combat these effects, we examine each signal to see if the parameters match their permitted values. We also send each work unit to multiple volunteers and cross-check the returned values to verify accuracy.

II.10.8 References

[1] Sullivan, Werthimer, Bowyer, Cobb, Gedye, Anderson "A new major SETI project based on Project Serendip data and 100,000 personal Computers" Proc. of the Fifth Intl. Conf. on Bioastronomy, 1997

[2] www.setiathome.ssl.berkeley.edu

[3] www.entropia.com

[4] www.peertopeercentral.com

[5] www.openp2p.com

[6] www.p2ptracker.com

[7] www.stanford.edu/group/pandegroup/Cosm

[8] www.parabon.com/cac.jsp

[9] www.fightaidsathome.org

[10] www.mersenne.org

[11] www.climate-dynamics.rl.ac.uk

[12] www.parabon.com

[13] www.popularpower.com

II.11 Grid FAQ

1. What is “the Grid”?

“*The Grid is an emerging infrastructure that will fundamentally change the way we think about – and use – computing. The Grid will connect multiple regional and national computational Grids to create a universal source of computing power. The world “Grid” is chosen by analogy with the electric power Grid, which provides pervasive access to power and, like the computer and a small number of other advances, has had a dramatic impact on human capabilities and inexpensive access to advanced computational capabilities, databases, sensors, and people, computational Grids will have a similar transforming effect, allowing new classes of applications to emerge*.”

-Ian Foster, Carl Kesselman, “The GRID Blueprint for a New Computing Infrastructure”.

2. What is the difference between the Internet 2 and the Grid?

There is some confusion about the differences between the Internet and the Grid. The Grid is not simply a more powerful Internet, or its direct descendant - Internet 2. The Grid is on the Internet because it uses standard protocols and mechanisms, but it has some unique characteristics. The main characteristics are dynamic sharing, single authentication, and virtual collaborations. Each of these capabilities depends on functions that do not exist in the Internet:

- tools for building collaborations and the shared environment;

- globally unique identification of data, and methods to share it;

- a shared security model capable of several levels of secure access;

- local control of access policies which are globally readable;

- accounting and audit trails for verification of usage and charges.

3. What is the difference between the Data Grid and the Computational Grid?

The main difference between the Data Grid and the Computational Grid is that the Data Grid relies upon emerging Computational Grid technologies. The Computational Grid is expected to make feasible the creation of a giant computational environment out of a distributed collection of files, databases, computers, scientific instruments and devices. Based on this environment, we are able to create specific and problem oriented Grids. One example of this is the Data Grid, which aims at the sharing and controlling of huge amounts of distributed data over the Computational Grid environment.

4. Is there any difference between Grid computing and distributed computing?

In fact, there is a difference between Grid computing and distributed computing. The growing popularity of the Internet along with the availability of powerful computers and high-speed networks due to the introduction of low-cost commodity components are gradually changing the way we do computing from distributed computing into Grid computing. These new technologies enable the clustering of a wide variety of geographically distributed resources, such as supercomputers, storage systems, data sources, and special devices, that can be used as a unified resource which allows an increase in the `quality’ of computing. Consequently, Grid computing is a great opportunity, challenging the way we think about more complicated computations, and differing from distributed computing in computation size, scope, heterogeneity and communication. Grid computing is distinguished from conventional distributed computing by its focus on large-scale resource sharing, innovative applications, and, in some cases, high-performance orientation.

5. What is the Resource Management System?

Generally speaking, Resource Management Systems are responsible for managing all the available resources in a way that gives them optimal performance, utility and efficiency. In Grid computing, Resource Management Systems are applied to a large number of shared resources such as networks, computers and other resources. Resource Management Systems are responsible for locating computational and communication resources on remote systems, and for incorporating these resources into parallel and distributed computations. The complexity of the resource management requires advanced scheduling mechanisms, resource allocation, resource reservation and brokerage tools for efficiently using the Grid's environment. All these mechanisms are essential to any credible Resource Management System used with the Grid.

6. What are the major benefits using Grids and who should consider using them?

The Grid is the next evolutionary step which takes computing into the 21st century. Similar to the Internet, which was developed in research laboratories and became the base infrastructure for the world-wide exchange of information, Grid technology is fast becoming the basis for distributed research science computations. Only recently, experimental scientists were using the Grid in order to perform testbeds and experiments but already, computational scientists and engineers are using the Grid for advanced scientific instruments, Teraflop desktops, collaborative engineering, distributed supercomputing, etc. Tomorrow, associations and corporations will need the Grid in order to improve their collaborative capability and global functionality. But what will be in the future…? As Grid technology comes into widespread use, it will fundamentally change the way we think about collaborative global computation.

7. What is Globus? Legion? Condor? What is their relation with Grids?

The Globus Project (http://www.globus.org) is leading the definition of standard Grid protocols and APIs, in such areas as security, resource management, data management, and information discovery. The open source Globus Toolkit, which provides a reference implementation of these Grid protocols and APIs, has been adopted my most of the major Grid projects world-wide, to provide a common, robust infrastructure for building applications that exploit distributed, heterogeneous, Grid-enabled resources.

Condor (http://www.cs.wisc.edu/condor) is a tool for harnessing the capacity of idle workstations for computational tasks. Condor is well-suited for parameter studies and high throughput computing, where sub-jobs generally do not need to communicate with each other.

Legion (http://www.cs.virginia.edu/~legion) is developing an object-oriented framework for Grid applications. The goal of the Legion project is to promote the principled design of distributed system software by providing standard object representations for processors, data systems, etc. Legion applications are developed in terms of these standard objects.

Condor and Globus are complementary technologies, as demonstrated by Condor-G, a Globus-enabled version of Condor that uses Globus to handle inter-organisational problems like security, resource management for supercomputers, and executable staging. Condor can be used to submit jobs to systems managed by Globus, and Globus tools can be used to submit jobs to systems managed by Condor. The Condor and Globus teams work closely with each other to ensure that the Globus Toolkit and Condor software fit well together. The two technologies are in some respects complementary: Globus focuses on low-level services and Legion on higher-level programming models. There are also significant areas of overlap. It may be useful to note that the Globus Toolkit is being used as the basis for numerous production Grid environments (from modest collaborative research projects to huge international scientific ventures), whereas Legion's community of users is smaller and more focused.

Portals

8. What is portal computing?

"Portal computing differs from what most people associate with the term 'portal.' Portal computing integrates business processes to connect customers, partners, suppliers, and employees, and is an evolution from client-server and Web computing. While client-server computing suffered from the

inability to connect client computers to heterogeneous systems across the enterprise, its successor -- Web computing -- required special, time-consuming programming to connect Web servers to each of these heterogeneous sys tems. Today, portal computing alleviates many of these problems because it offers a middle-tier application layer that provides the glue between clients and backend systems across multiple hardware and software platforms."

- Sun Microsystems' definition

9.What is a portal access to the Grid?

Basically, the portal access to the Grid is similar but not exactly the same as portal access to the Web. The essence of the difference between the two lies in what we expect to do with Grid and Web access. Grid portals differ in several ways from current, customisable portals. The objective of a Grid portal is to provide a consistent, easily used interface to a complex environment so users can rapidly get their work done without being computer experts. Portal access to the Grid enables users to do computing, run codes, manipulate data and files and otherwise use components of the computational Grid through a web interface. Where single-server Internet portals are designed to tie individuals to a service, Grid portals are designed to enable easy access to a diverse set of resources. Naming both types as "portals" could create confusion.

Grid Security

10. Why is Grid security so important?

Because Grid resources are managed by many different (often international) organisations, with different security requirements and possibility conflicting security policies, many people are working on issues relating to authentication and authorisation in Grid environments in order to develop a Grid security infrastructure. This inf rastructure will enable organisations providing computing resources to specify policies to protect their equipment, and in turn to protect other computations that run using common resources. Grid security solutions should provide flexible support for communication protection and should enable stakeholder control over authorisation decisions, including the ability to restrict the delegation of rights in various ways.

11. What is Certificate Delegation?

A user must be able to endow a program with the ability to run on that user's behalf, so that the program is able to access the resources on which the user is authorised. The program should (optionally) also be able to conditionally delegate a subset of its rights to another program (sometimes referred to as restricted delegation). For example, in the Globus environment, the special process called gatekeeper, can act as representative of client and allows a client to delegate a proxy certificate to the gatekeeper server.

12. What is the security policy?

The security policy is specified by configuring access control lists associated with individual resources. Administrators, who are responsible for controlling all the individual resources, can manage and control security policy by allowing users and objects to be grouped so that when a particular policy is changed in one place, the change affects all resources to which the policy applies.

13. What is "single login"?

Single authentication means that you identify yourself once, and that authorisation is automat ically forwarded where needed. This capability exists within a single network entity of trusted systems and users, but the SA capability is capable of global authorisation from a single login. Thus, across geographic and organisation boundaries, your authentication enables automatic access to all functions that you are entitled to access.

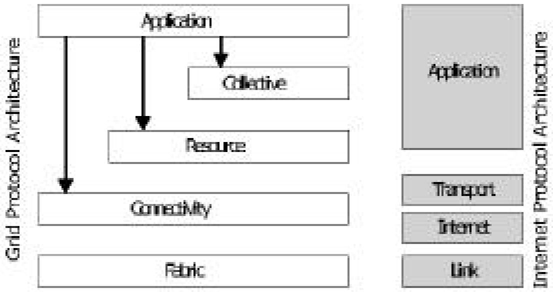
Grid Standards

14. Are there any Grid standard protocols?

Today, there are many efforts to define standard Grid protocols and APIs such as security, resource management, data management, and information discovery. One of the main and most successful project that helps develop standard Grid protocols is the Globus. The open source Globus Toolkit, which provides a reference implementation of these Grid protocols and APIs, has been adopted by most of the major Grid projects world-wide, to provide a common, robust infrastructure for building applications that exploit distributed, heterogeneous, Grid-enabled resources (for example: Grid Resource Information Protocol (GRIP), Grid Resource Access and Management (GRAM), extended File Transfer Protocol (GridFTP)).

15. Is there any standard Grid architecture?

Today, it is hard to say that there is a standard Grid architecture, but rather set of proposals. The figure below illustrates the proposed layered Grid architecture and its relationship to the Internet protocol architecture.



Due to the fact that the Internet protocol architecture extends from network to application, there is a mapping from Grid layers into Internet layers.

16. Is there a Grid standardisation body?

Yes, there is a global Grid standardisation body, called Global Grid Forum. www.gridforum.org

Grid testbeds and projects

17. What is the Grid Testbed?

In order to put the Grid idea into reality, a set of experimental Grid Testbeds have been established around the world. The Grid Testbed encompasses many organisations and is a real uniform running environment whose behaviour can be tested and monitored in order to facilitate further development.

18. Are there any references to Grid projects and testbeds world-wide?

There are many existing Grid projects involved with developing the Grid Testbed. The most important ones are:

http:// www.Gridforum.org

http:// www.globus.org

http://www.aei.mpg.de/~allen/TestBedWeb/PR/index.html

http://proj -dataGrid.web.cern.ch/proj-dataGrid/

http://hipersoft.cs.rice.edu/grads/

http://www.griphyn.org/

http://www.euroGrid.org/

http://www.research-councils.ac.uk/escience/

Other Grid-related technologies

19. What is peer-to-peer computing?

Peer-to-peer computing is a set of concepts rather than a specific technology. Put simply, peer-to-peer computing is the sharing of computer resources and services by direct exchange between systems. These resources and services include the exchange of information, processing cycles, cache storage, and disk storage for files. Peer-to-peer computing takes advantage of existing desktop computing power and networking connectivity, allowing economical clients to leverage their collective power to benefit the entire enterprise.

In a peer-to-peer architecture, computers that have traditionally been used solely as clients communicate directly among themselves and can act as both clients and servers, assuming whatever role is most efficient for the network. This reduces the load on servers and allows them to perform specialised services (such as mail-list generation, billing, etc.) more effectively.

20. What is ASP?

ASP (Application Service Provider/Provision) is a new way to sell and distributed software and software services. Although ASPs were possible before the advent of the Internet, the Internet makes them so easy to create that they have proliferated hugely in the last several years. The ASP model can be extremely appealing to companies, especially small business and startups, because it can drastically lower the costs of software and services. With the advent of the Grid, the ASP model could expand rapidly as a consequence of more widely available high-quality resources such as computation, applications, services and security.

Part III The study

III.1 Introduction

III.1.1 Rationale

As was emphasised in Part I, the central aspect that policy makers, Grid developers, and service providers must consider in the first instance is users' requirements and how these can be met by emerging Grid technology. These requirements must be clearly understood and addressed, as they will form a pre-requisite to a widespread acceptance and rapid uptake of Grid technology by the user community. Indeed, the routine use of early production Grids will be key to the long-term success of Grid computing. Unlike earlier technological milestones in computational science and engineering, such as the advent of High-Performance Computing (HPC), where users' requirements and expectations were sometimes guessed, it is important for people developing and operating the early production Grids to establish a direct and permanent contact with the end-user community at large.

The massive investment in Grid-related technology which has taken place over the past couple of years has been accompanied by a hype which often raised unrealistic fears and expectations about Grid computing. It is therefore important to raise the awareness of the user community whilst still ensuring that the information they receive is both complete and accurate and also in a form which can be easily understood. Although there are now a profusion of publications on Grid computing distributed either through mainstream HPC conferences such as Supercomputing [14] or dedicated bodies such as the Global Grid Forum (GGF) [4], most of the documentation is concerned with fairly specialised technical issues. For example, details of APIs etc. are essential to the developers and implementers communities but are proving simply irrelevant and confusing to users who are only concerned about higher level services and policies. Much more effort must now be dedicated to explaining what Grids actually are, what they can offer (their direct relevance and immediate benefits) as well as their limitations (kill the hype). The true challenge is to demonstrate how users' expectations can be met and even on occasion surpassed. There is now some clear evidence that such dissemination activities have started taking place at a national level, for example through the UK High-end Computing (UKHEC) series of seminars [15]. The question is whether these initiatives are as effective amongst the user community as they are for the Grid developers and resource providers. Similar initiatives also exist elsewhere in Europe. Another danger is the over-simplification of the description and issues relating to Grid Computing. This is exemplified by the fact that Grid Computing is often presented to the general public as 'the next-generation of the Internet' (e.g., [16]), which has probably instilled confusion into a number of prospective users from the scientific community.

The purpose of the survey presented in this third part is thus to gain a better understanding of typical European user groups' composition, working practices, and applications bottlenecks with a view to establishing their key requirements. In addition to this, a careful study of the level of awareness in these groups will give some indication of the current understanding of what users regard as key issues, as well as their expectations and reservations towards Grid Computing. These findings, together with the previous section in which technical and implementation issues were first introduced, will form the basis of a series of recommendations to policy makers and resource providers in the final part of this study.

III.1.2 Methodology

In most statistical surveys of this kind, such as the DIRECT survey [17], a very large population of people (typically well over 1,000) is invited to participate in the survey and no attempts are made to check for unacceptable bias. The opinion of a graduate student is thus given equal weighting to that of a group leader, regardless of the fact that they may not possess the same breadth of perspective on the issues under scrutiny and are probably not vested with any decision-making powers. This is a worrying fact considering that this first category of respondents are much more numerous and are more likely to have the time to take part in lengthy surveys. The conclusions drawn from such statistical data should therefore often be treated with caution.

Bearing this fact in mind, the authors took a different approach and decided to open the survey *by invitation only*. The main objectives of the study were discussed with the partners from the ENACTS network during the project's kick-off meeting held in Bologna on 12 and 13 February 2001. It was agreed that this survey should aim to contact a minimum of 70 European group leaders, expecting to get around 50 replies. The authors significantly exceeded these targets and obtained feedback from 85 group leaders either in person or through their named authorised representative. (Note: some of the statistics quoted in this report are based on 84 responses, as one respondent only partially completed the questionnaire.)

The selection of suitable candidates was done according to the following criteria:

- Eligible participants must be a group leader or a named (authorised) representative from this person;
- The geographic distribution (at a national level) should be in-line with the demographics and the predominance of the HPC infrastructure; and
- The sampling should target all areas of computational science and engineering (again trying to reflect their relative importance at a national level). In this way the resulting population is expected to provide a reasonable representation of prospective Computational Grid users over the next three years.

The questionnaire was comprised of 48 points split into seven main categories:

1. *Start:* prompts for personal details of the respondents as well as key characteristics of the group that they represent (e.g., scope, level of co-ordination, size, etc.). See "Section 1: Start," page 121.

2. *Awareness:* purports to establish the level of awareness within the user community. This is achieved by presenting the respondents with the five main categories of Grid models described in Part II and asking them to comment on their suitability for their group's requirements. As a reminder, the computational Grid models were categorised as follows:

   - basic portal computing (wokbenches);
   - 'advanced' portal computing (problem solving environments);
   - small-scale Grids (load-sharing and cycle stealing systems);
   - toolkit approach (e.g. Globus); and
   - Internet computing and peer-to-peer computing.

   Although this taxonomy significantly differs from those typically presented in other Grid publications (e.g., [18]), this classification can be justified by bearing in mind that the aim of this section is to determine how Grid infrastructures appear to their prospective users (in term of features and usability), rather than their underlying middleware and protocols. Another key aspect of this section is to find out which factors are most likely to entice end-users into adopting Grid-based resources routinely and to incite them to tackle larger and more complex problems (see "Section 2: Awareness," page 125).

3. *User profile:* this section is concerned with the groups' application areas, as well as their level of experience in applications development (i.e. package users vs. developers). This latter point is particularly interesting as differing working practices are expected to have a considerable impact on the types of Grid a user will wish to access (see "Section 3: User profile," page 146).

4. *Application profile:* following on from the previous section, it also appears essential to gain a better understanding of how the characteristics, requirements and bottlenecks of today's mainstream applications will influence the adequacy of the various Grid models. Typical obstacles to make these codes Grid-compliant (mostly portability issues and network awareness) have also been investigated (see "Section 4: Application profile," page 156).

5. *Infrastructure:* the performance of local networks to the main national backbone is known to be one of the main bottlenecks for distributed computing (typically for I/O intensive applications, distributed computing and metacomputing). This section investigates the characteristics of the local networks as well as the type and amount of resources available to these groups within their

own institute/company, at a national level and elsewhere (see "Section 5: Infrastructure," page 170).

6. *Security & services:* security is often put forward as the main reason for the reluctance of service providers to join a Grid infrastructure and for users to overlook such resources. Respondents were asked to comment on the level of control and guarantee they would seek before considering joining and using a Grid environment. The respective importance of typical 'added services' and the concept of quality of service (QoS) are also discussed (see 'Section 6: Security and services," page 174).

7. *Future needs:* this section of the questionnaire was initially included for the second ENACTS study ("The HPC Technology Roadmap") jointly undertaken by NSC and CSCISM [19]. The key issue is to elucidate how users' requirements are likely to evolve over the next ten years. This section provides a valuable insight about how the findings from the fourth category will evolve with time (see "Section 7: Future needs," page 183).

In addition to the selection criteria presented earlier, the selection of participating research groups was also based on their reputation and scientific excellence. The ENACTS consortium is composed of partners from 13 different countries from the European Union and associated countries, most of them operators of national or regional HPC facilities. As each partner is therefore expected to have a good overview of the research activities undertaken in their home country, it was agreed during the kick-off meeting that the task of selecting individual groups within each country represented by an ENACTS partner would be left to them to decide. For other countries not represented within ENACTS (e.g. Germany), and for these cases where too few group leaders replied to the original invitation to participate, the ENACTS Team at EPCC endeavoured to take overall responsibility for the selection of additional user groups.

As most participants asked to remain anonymous, the list of participants has not been published in this document. Table 4: indicates the number of research groups which were contacted, how many of these agreed (in principle) to participate, and how many did indeed take part to the survey.

Country	*Contacted*	*Replied*	*Answered*	*Target*
Austria	2	1		
Belgium	1			1
Czech Republic	5	4	4	1
Denmark	6	5	5	5
Estonia	1			
Finland	21	8	6	5
France	18	5	4	8
Germany	31	15	13	10
Greece	12	9	6	5
Hungary	1			
Iceland	1			
Ireland	7	6	3	3
Italy	12	6	4	8
Netherlands	5	2	2	3
Norway	6	5	5	5
Poland	4	3	3	5
Portugal	8	7	2	2
Spain	12	8	8	8
Romania	1	1		2
Slovakia	1	1	1	1
Sweden	17	9	7	5
Switzerland	1			4
United Kingdom	30	14	11	10
No answer			1	
Total	203	109	85	91

Table 4: nationality of participants

For the sake of completeness and transparency, all the statistical data from the questionnaire together with the raw data and all the com ments made by the participants is presented in this document. A copy of the questionnaire has also been included in Appendix B.

III.2 Section 1: Start

This section establishes the profile of user groups who took part to the survey. The main factors considered when establishing the profiles were:

- geographical distribution of the groups;
- their size and scope; and
- their overall level of co-ordination.

III.2.1 Geographical distribution

The survey achieved most of its targets with the exception of a lower than hoped for level of participation from French and Italian researchers as shown in Figure 18. It is worth noting that the same phenomenon had been observed during a previous survey [17].

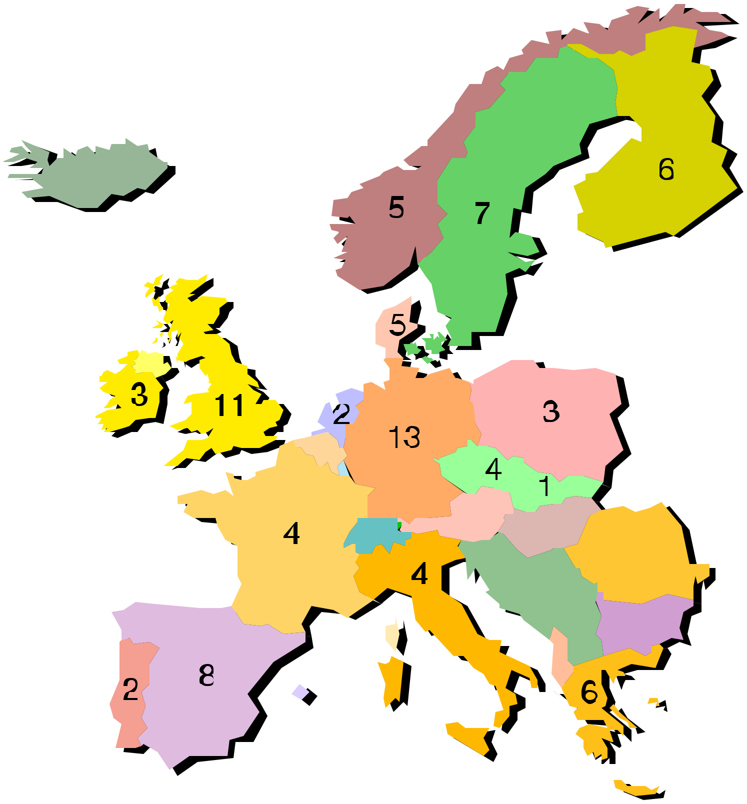


Figure 18: geographical representation of user groups

A good geographical distribution was essential to ensure that the influence of specific factors introduced by national policies on funding research and on infrastructure such as HPC (and networking) is reflected in this survey. For example, the national strategy on the provision of HPC services i.e. the balance of investment between centralised national services and smaller distributed facilities, has proven to have a significant impact on the type of scientific challenges undertaken. Indeed, large-scale resource constrained applications (a.k.a. the 'Grand Challenge' problems) cannot be tackled without access to large-scale HPC facilities. This has been observed in Denmark where the funding of the IBM SP2 at UNI-C was discontinued in August 2000, to be replaced by support for smaller-scale distributed systems such as Beowulf clusters. Such policies also strongly influence the type of research activities undertaken, so it would be expected that they will indirectly have implications for the expectations their user community will place on Grid environments in terms of allowing them to tackle Grand-Challenge problems. As the first generation of production Grid testbeds are expected to be established at a national level, this type of consideration has to be borne in mind throughout the survey (although there are notable exceptions such as the Datagrid [20], EuroGrid [21], and Damien [22] projects, funded under the EC's Information Society Technology (IST) Programme [23]). The statistical treatment presented here takes this into account and any discrepancy based on such factors is highlighted.

III.2.2 Size and scope of user groups

The Grid purports to foster collaborative projects by creating so-called Virtual Communities (VCs) of researchers. These communities are expected to form and evolve naturally as distributed problem-solving environments (PSEs) gain in functionality and become more readily available. This effort is exemplified by the current deployment of Access Grid nodes across the world at a very fast rate: The number of Access Grid nodes registered at the Argonne National Lab [24] now exceeds 70[1], including ten in the EU. This number is expected to treble by mid 2002.

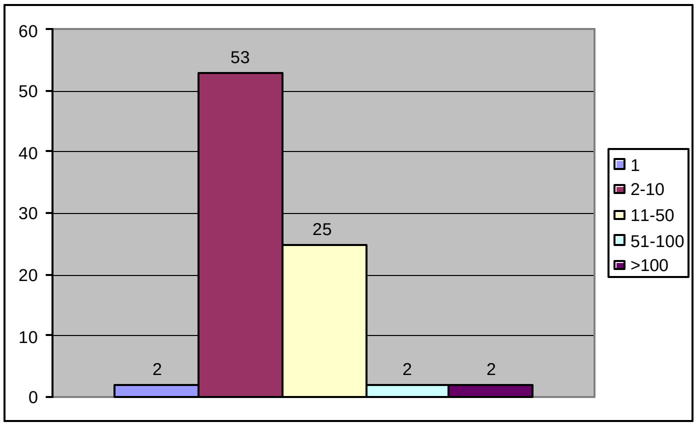


Figure 19: size of research groups

The benefits of this technology are expected to increase considerably over the next few years (Metcalfe's law) and will be equally important to research groups of all size. Small groups will be better equipped to sustain closer contacts with colleagues at remote locations, whilst larger groups will find it easier to set up more concerted actions and a higher level of co-ordination between their constituent sub-groups. The net effect of setting up such an infrastructure (provided easy access can be secured) will potentially have a tremendous impact on the way researchers are currently working, to the same extent as E-mail and the World Wide Web revolutionised working practices a decade ago. One logical implication of these new opportunities is that the average size of user groups would be expected to increase.

Finally, it should also be pointed out that although the natural early users of the Access Grid will be in computational science and engineering (as access will principally be provided by regional HPC centres), its impact will be felt far beyond this to benefit non-computational sciences such as the social sciences.

[1] Number of sites registered at Argonne National Lab on 13th July 2001 [10].

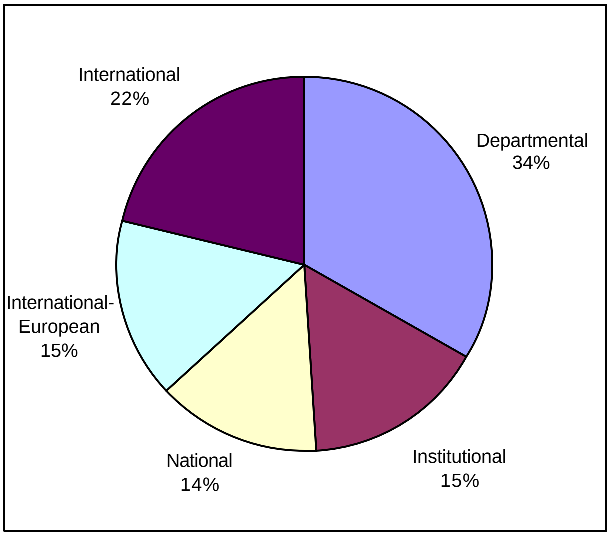


Figure 20: scope of research groups

III.2.3 Level of co-ordination

Over half the groups polled are engaged in national and/or international collaborations. This in not unexpected, as the vast majority of research groups targeted by this survey have achieved a significant level of recognition within their own field. However, purely departmental or institutional groups were equally well represented. This is important as many research groups across Europe fit within this category. A strong correlation was observed between the scope of the research groups and their level of co-ordination, as shown in Figure 21. Research groups based in the same university or organisation benefit from a much more centralised co-ordination. As the scope of the groups expand to the national and international level, the increase in group size combined with practical considerations such as communication issues, results in a more flexible level of co-ordination, normally through a different primary investigator (PI) for each distributed sub-group.

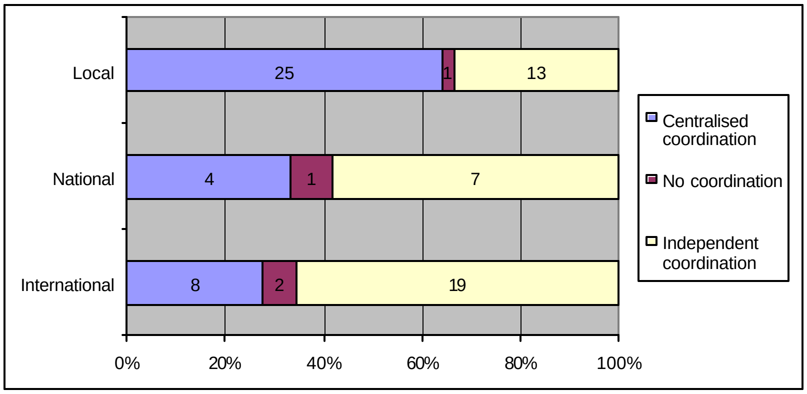


Figure 21: level of co-ordination

The high level of co-ordination recorded in this survey is the most striking aspect of this section. It is particularly re-assuring to see that over 90% of the groups polled are co-ordinating access policies for their resources in some way. Indeed, most of the new overheads and obstacles associated with Grid computing are likely to be political (e.g., allocation policy for individual users and sub-groups, and possible links with other groups) and administrative. For example, as most Grid middleware requires a regular user account for each new Grid user, the overhead associated with user administration, accounting, resource management etc. will increase accordingly.

The existence of highly co-ordinated hierarchical structures within the user community will prove a major factor in facilitating the deployment and acceptance of computational Grids. Indeed, concerted political decisions should become easier as representatives from the user community will be clearly identified and invested of real negotiation and decision-making powers. These group representatives are also bound to possess a more global understanding of the implications and challenges posed by the deployment of production Grids. On the one hand, there will be opportunities to increase the net amount of resources available to their group and to undertake new scientific challenges enabled by the Grid. Conversely, it is expected to facilitate the sharing of at least some resources between groups, either by setting up their own Grid and allowing access to their collaborators, or by joining existing Grids. This issue is discussed further in "Section 5: Infrastructure," page 170 and "Section 6: Security and services," page 174.

III.2.4 Summary

The research groups participating in this survey cover a wide geographical distribution over the European Union and associated countries. This was important in order that the impact of national policies for the provision of HPC and networking on user communities could be assessed.

The size and scope of the respondents' groups comprise a mixture of small to medium size groups (two thirds of the groups have between 2 and 10 researchers) and larger groups (11-50 researchers). Half of the groups are involved in collaboration at the national or international level. The emergence of the Access Grid technology is expected to improve opportunities for increased collaborations between these groups, and indeed to benefit communities from other areas such as the social sciences.

The overall level of co-ordination for all these groups is well over 90%. This ranges from highly centralised co-ordination for local groups to a more flexible structure made of sub-groups with their own PIs for those groups which are involved in large international collaborations. The high level of co-ordination throughout is comforting as it provides an excellent framework for addressing the overheads inherent to Grid computing, both for groups wanting to utilise and for those willing to provide Grid-based resources.

III.3 Section 2: Awareness

As was emphasised previously, the raising of awareness in the user community is the key to a wide and rapid acceptance of Grid technology for routine production work. This section was therefore designed to establish a snapshot of the degree of awareness and to understand a number of specific issues such as:

- Users' perception about the usefulness of various Grid models;
- Major benefits users expect from Grid computing;
- Preferred scope and access restriction of computational Grids;
- Past experience of various Grid enabling technologies;
- Key features Grids must provide to entice users into migrating to a Grid infrastructure;
- Preferred computing architectures to access through a Grid;
- Any major factors which may compromise or delay the uptake of Grid computing in the user community; as well as
- Influence of Grid Computing on the type (scale) of scientific problems being tackled.

III.3.1 Basic awareness

The survey was conducted between 12 April 2001 and 29 May 2001. This was a period of time when the EC and national research councils were starting to advertise funding opportunities for Grid-related projects. Considerable funds had been allocated for the upgrade of the underlying network infrastructure, essential to the deployment of wide area Grids and virtual private networks (VPN). For example the €80M GÉANT project will create a pan-European backbone at Gbps [3]. Funds had also been allocated for the development of enabling middleware such as the Damien (€2.0M) and GRIP (€1.8M) projects and for testbed projects [23]. These initiatives are perfectly exemplified by the Datagrid project [20] run by CERN and its partners. Complementary actions such as the £98M UK e-Science programme [25] were also put in place at a national level. A number of high-profile pilot projects have since been funded by the European Commission [23] and national research councils [26].

The level of Grid awareness reported by users in the survey is shown in Figure 22. Since this survey was undertaken, the advent of Grid-based computational science (or e-Science as it is now commonly called) has been widely publicised in the user community. For this reason, it is likely that the actual level of Grid awareness is now higher than Figure 22 would suggest.

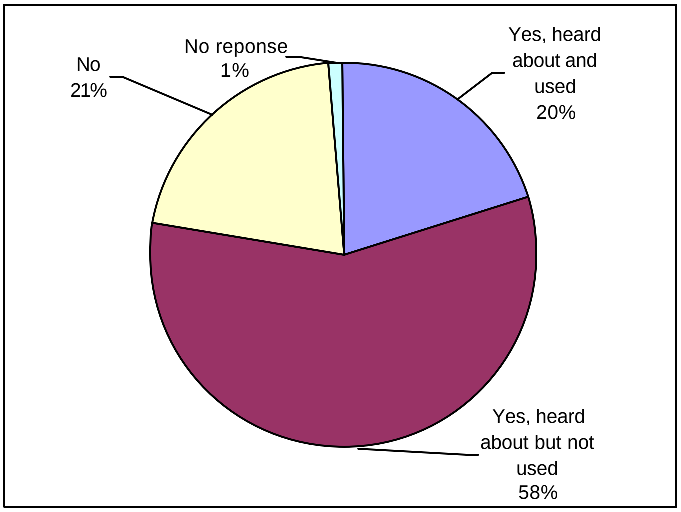


Figure 22: previous encounters with the Grid

Although 77% of the respondents claimed to have heard about ‘the Grid’, most (74%) admitted to having no real practical experience of using any type of Grid. However, a closer study of the data reveals noticeable discrepancies across geographical borders and research communities. This is particularly noticeable when looking at the geographical distribution of the groups who have had some prior practical experience of using a Grid (see Figure 23).

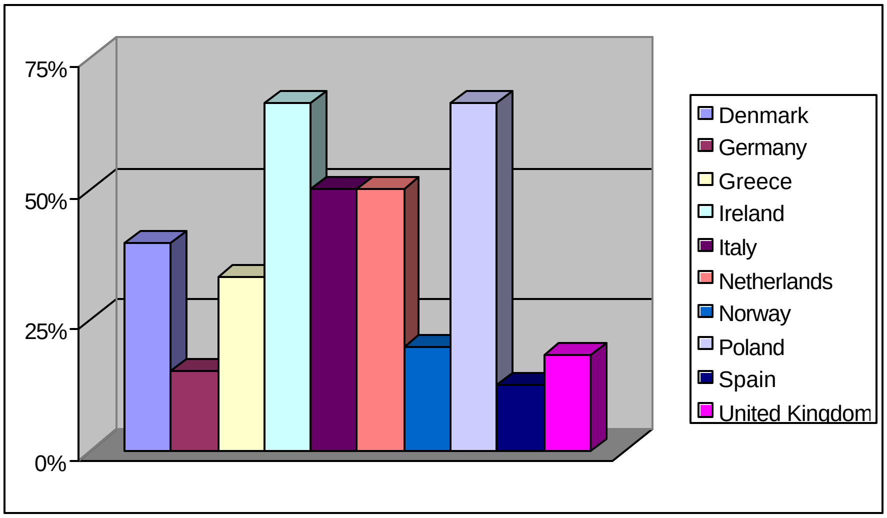


Figure 23: geographical distribution of groups with prior (practical) experience of Grid computing

The data presented in Figure 23 can only indicate trends as the statistics are based on such small samples - for example, the statistics for Ireland and Poland which appear as the most ‘Grid-aware’ are based on the answers from only of six respondents. However, it is clear that national initiatives will have a wider reach to promote Grid computing to prospective end-users than will larger-scale initiatives. This is not surprising considering the continuous and effective dialogue established by the national Research Councils with the local research community. In comparison, EC-funded programmes have yet to make a direct impact on the grass root research community.

Applications area	*Prior experience of Grid Computing*	*Number of Respondents*	*Experience level for Research area*
Astrophysics	2 (12%)	7	29%
Computing	10 (59%)	14	71%
Mathematics	1 (6%)	2	50%
Physics	2 (12%)	17	12%
Telecommunications	1 (6%)	3	33%
Other	1 (6%)	11	9%
All other disciplines	0 (0%)	30	0%
Total	17 (100%)	84	

Table 5: scientific areas of groups with prior (practical) experience of Grid computing

The variations observed between the different research communities is more significant as it is based on slightly larger samples. Not surprisingly, the majority of users who had the opportunity to test drive Grid testbeds are involved in Computing. Three of them actually declared being involved in the development of Grid middleware or deployment of Grid testbeds, whilst others were looking at computer science applications of the Grid (performance predictions, automatic parallelisation, automatic scheduling, metacomputing, etc.). Four of these groups also indicated that they had developed and were experimenting with their own Grid testbed(s). The results presented in this table can be placed into perspective by comparing this to the respective representation of each applications area. With the exception of computing (i.e. computer science and HPC) ,which is the most closely related to Grid computing, it is noticeable that Grid computing has already been embraced by many in the astrophysics community, and to a lesser extent tried out by physicists. The number of respondents for the other disciplines listed in this table is too small to get a statistically significant idea of their level of awareness.

The predominance of astrophysics as *the* Grid-aware scientific community will not surprise anyone who has followed the evolution of Grid computing over the past couple of years. Indeed, this community has been involved in pioneering activities such as the development of the Cactus problem-solving environment [27]. The versatility of Cactus was demonstrated as early as in 1998 when the Cactus group performed an intercontinental, distributed simulation of the full 3D Einstein equations of general relativity, calculating the collision of black holes and neutron stars. The simulation itself was distributed across supercomputers using tightly coupled supercomputers in Europe and the US and presented at Supercomputing '98 [28]. This group was also instrumental in setting up the European Grid Forum; their involvement in Grid research has also been noticeable in the EuroGrid consortium [21], and is being pursued through their active participation to the GGF Applications Group [29] and the Gridlab [5] project [1] recently funded under the IST programme of the European Community.

The reasons for this interest in Grid computing are particularly interesting to consider. They can be put down to a combination of motivating and enabling factors:

1. the astrophysics community enjoys a high-level of national and international collaboration (e.g. through initiatives such as the Astrophysics Simulation Collaboratory (ASC) [30], or the Virgo consortium [31];
2. constituent research groups are well co-ordinated;
3. many of the scientific problems of interest in this discipline are Grand-Challenge applications and/or time-critical applications such as interfacing to experimental devices which require innovative approaches to get solved;
4. strong links with local HPC centres and their personnel have been established and maintained for long periods; and
5. good awareness of technological solutions such as PACX-MPI [32].

All these factors combined together have established the motivation and infrastructure to investigate the benefits of Grid computing at a very early stage indeed. These motivations are perfectly exemplified by the Gridlab project referenced earlier.

The key issue might be that some Grand Challenge problems have such large requirements that they are only tractable on a handful of supercomputers around the world. The difficulty of 'booking' such resources and the associated delays has meant that metacomputing has formed a very attractive alternative. This problem becomes even more acute when dealing with time-critical applications such as links to experimental devices. Astrophysicists and their collaborators have developed very clever technological and algorithmic solutions to mask the latency and limited bandwidth between the supercomputers forming the metacomputer. The old yet tenacious belief that metacomputers are unsuitable for tightly coupled applications is simply not justified any longer. The only real barrier to production use of metacomputers is the lack of mechanism to pre-book all the required resources for a given time window (synchronisation). This concept, known as 'advanced reservation', is currently being tackled by the GGF working groups from the 'Scheduling and Resource Management' area [33].

Research groups from other disciplines could equally benefit from this approach to tackle larger scale or more complex applications in the terascale regime (Tflops, Tb). The different perspective between research groups with practical experience of Grid computing and the rest of the respondents is highlighted in several of the following sections.

III.3.2 Preferred Grid models

The participants were asked to comment on the usefulness of various Grid models for their own research. The classification followed the taxonomy presented in Part II where a detailed discussion of the issues underlying these models is also presented.

[1] This project is led by Jarek Nabrzyski from the Poznan Supercomputing and Networking Centre (PSNC), co-author of this report.

For each of these Grid models the respondent could select one from the following levels of interest: 'very useful'; 'useful'; 'of some use'; or 'not useful'. By constructing the question in this way it was possible to express an interest in more than one grid model. There were 217 'useful' or 'very useful' replies for the various Grid models (see Figure 24). As the survey had 85 respondents this means that on average each respondent considered that between 2 and 3 (2.6) different Grid models would be 'useful' or 'very useful' to them.

Attempts have been made through this report to identify the suitability of these various Grid models for different user groups and application types. The methodology consisted in extracting selectively the answers for respondents who selected 'useful' or 'very useful' for each of these models. When filtered in this way there was no significant difference between responses for the different Grid models. However, as many replies were 'double counted' it only indicates that this approach was not a suitable way to filter the responses.

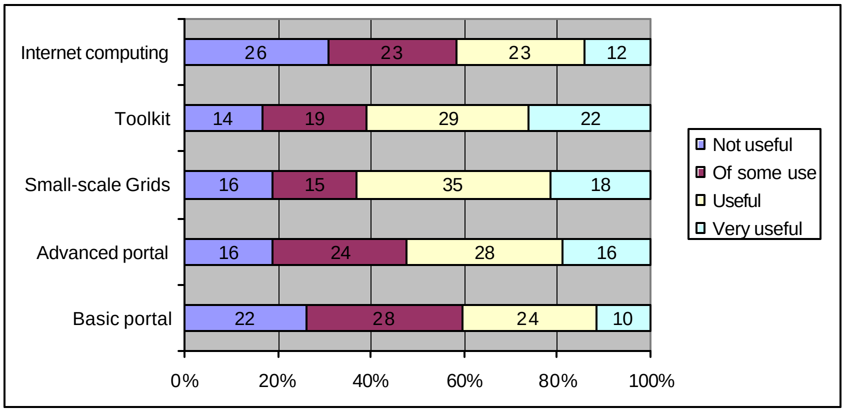


Figure 24: usefulness of various Grid models (all respondents)

As shown in Figure 24, the small-scale Grids and Toolkit-based Grids are perceived to be the most useful. This response is not surprising considering the amount of publicity which surrounded the Globus toolkit [34] over the past year, culminating in Europe with the organisation of the first Euroglobus workshop at the Marina di Ugento (Lecce, Italy) in June 2001 [35]. In fact, Globus is on the way of becoming a de-facto standard, especially as the work from the GGF working groups often takes care to produce 'Globus-friendly' recommendations and specifications. This popularity has been confirmed in this survey with Globus proving to be the most commonly referenced system amongst users with prior experience of Grid technology (see Figure 28, page 139). The predominance of Globus is expected to grow even further as many national research councils currently regard Globus-based solutions as the only ones that are strategically viable.

The popularity of small-scale Grids is due to different factors. Unlike Globus, which is a fairly recent product, small-scale grid middleware such as LSF [36] and Condor [37] have already been in existence for many years and therefore benefit from a strong user base. In fact, their existence pre-dates the theoretical concept of 'the Grid' which is accepted to have originated following the seminal book by Foster & Kesselman in 1998 [38].

The relative lack of interest for portals was rather surprising. It was expected that its user-friendliness - a very important feature for prospective users of the Grid (see Figure 29, page 141) - to be attractive to most users outside the Computer Science community. A combination of factors may explain this disaffection:

- misconception about what these systems can actually offer;
- fear of being 'locked' into using standard packages available through the portal, with no possibility of accessing your own code or modifying it;
- fear of overhead to get custom applications incorporated in a portal system.

The first point is especially pertinent for the less experienced users. This misconception is cultivated by the 'traditional' examples of Web portals such as Yahoo, Lycos, etc. combined with the description of 'the Grid' being 'the new generation of the Internet'. Access to *functional* portal testbeds would be expected to increase considerably the support for this approach.

The remaining issues are legitimate. In the early days at least, the integration of an application within a portal system will rely on the service provider developing a purpose-built module or plug-in for this application. This process will undoubtedly require the participation of skilled staff at the service provider and therefore will involve overheads in development time and its associated cost. These overheads could only be justified if the application code is stable enough over time and used by a sufficient number of users. Codes 'in development' and test applications are obviously not suitable for a 'basic portal' environment. A discussion of user and application profile and their correlation with the perceived usefulness of these Grid models is discussed in the next two sections.

At present users are placed in a similar situation with the Grid as they were a decade ago with parallel computing. They need to be informed, trained, and be offered support to get started on these new systems. The onus will be placed on service providers (HPC sites mostly to start) to provide a high level of support e.g. through a mechanism resembling that of the UK EPSRC High-Performance Computing Initiative (HPCI) [39]. The research councils could thus financially support work by the HPC centres to integrate application codes in portals (a cost-effective solution using a plug-in architecture) and, for instance, make mainstream production codes Grid-aware.

With the exception of portal computing, who's popularity remained pretty much unchanged, users with practical experience of Grid computing passed a more enthusiastic verdict altogether. Figure 25 shows that this trend is particularly noticeable for the toolkit and small-grid models, which showed a gain of over 20% in their perceived usefulness compared to the average in Figure 24, page 128. It should also be noted that users with little or no awareness of Grid computing based their answers and comments mostly on the short descriptions of the various Grid models included in the questionnaire, *not* the in-depth discussion from Part II. These descriptions have also been included for reference in Appendix C.

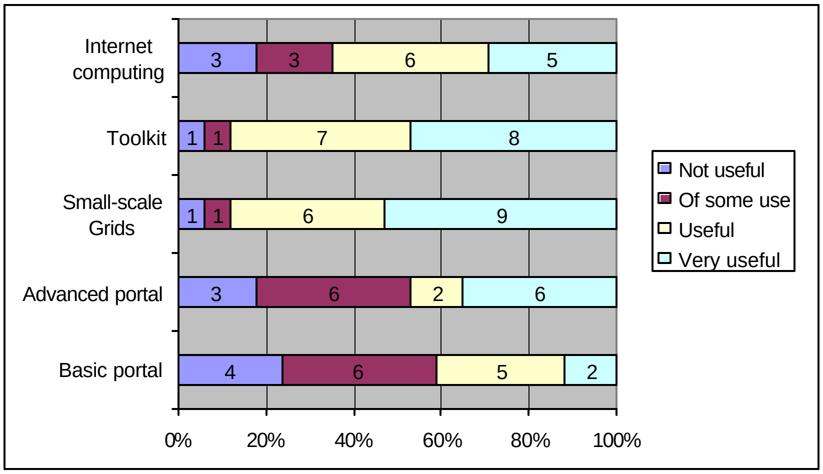


Figure 25: usefulness of various Grid models for groups with prior experience of Grid computing.

The conclusions which can be drawn from these variations are quite encouraging as they suggest that providing access to Grid testbeds will increase overall support from the user community. This is an initiative which has already been undertaken by some HPC centres (e.g., see [40]) and should be encouraged! However, access to testbeds is not sufficient in itself. Bearing in mind that attendance to Grid conferences (such as GGF) by end-users is virtually null, a better strategy would be to demonstrate the effectiveness and benefits of Grid computing at user conferences using packages and application codes used by that research community (e.g. Gaussian at a Chemistry conference).

It should be borne in mind that users who are already engaged in testing Grid based solutions would be expected to be more favourable to this concept in the first instance. Another possible factor explaining these variations is that unlike portals (for which underlying issues are quite easy to understand), the concept, issues and benefits associated with the three other models are not as straightforward to grasp without having had the opportunity to use them.

Another way of interpreting the data was to filter out the answers of respondents who had singled out only one model as being 'very useful' and correlate this with their prior level of exposure with Grid computing (see Figure 22, page 125 and Figure 24, page 128). The motivation for this approach was to separate respondents who have a strong opinion of what will be useful to them from the others, who are more likely to have less focused expectations.

At the two extremes, the reported level of exposure to Grid computing of the supporters of the basic portal and toolkit approaches is as would be expected. The high level of experience amongst the group advocating the toolkit approach is not so surprising considering that this was until recently the only truly usable approach for scientific computing. The answers from the respondents who had not heard of 'the Grid' in general is equally striking, with the majority of them (80%, see Table 6) being attracted by the prospect of accessing computational Grids through portals. The particular significance is that these respondents (who had not heard about Grid computing before taking part to the survey) had so far been shielded from all the hype and misconception surrounding 'the Grid' and associated Grid enabling technology. Their reaction does therefore provide a good feel for the genuine reaction of the user community at large.

	Basic portal	*Advanced portal*	*Small-scale Grids*	*Toolkit*	*Internet computing*
Heard about and used	0	2	1	4	1
Heard about but not used	1	3	2	2	1
Never heard about	2	2	0	1	0
Total	3	7	3	7	2

Table 6: respondents mostly interested by a single model, and their prior exposure to Grid computing

III.3.3 Perception and benefits of computational Grids

Reluctance to adopt Grid computing is mostly based on misunderstanding of what Grids can actually offer and the level of overheads they will introduce. It was therefore important to check whether the models presented in this survey did indeed correspond to the respondents' perception of the Grid *before* starting the survey. It was reassuring to see that 79% of the participants confirmed that their understanding of Grid computing coincided with the descriptions provided in this report.

Two distinct categories were identified amongst respondents who had a different perception of Grid computing:

1. Those who perceived the Grid as dealing with other issues (other aspects of the Computational Grid such as Metacomputing, and other types of Grids such Access Grids, Information Grids, Data Grids, Virtual laboratories, Computat ional steering, etc.); and
2. Those who had not heard of the concept at all.

As was emphasised by some of the participants, the key question users want to see answered is:

"*how do these possibilities can really help my work?*"

One of the respondents added their own vision of what computational Grids should be about. Judging from other comments, this vision is shared by many:

> *"I do not really have a good idea of what grid computing is. What I would like to see is a very simple user interface where I can write my programs in a convenient language (say C or ZPL for parallel programs) and then only have to type say:*
>
> *% remote Program.z*

> *and then the system finds an appropriate computer to compile and run my program on (and it should find any local input-files I defined) and after it is finished I will find the resulting files in the directory I called the program from. Ideally this would also work for interactive programs. There should then also be a simple utility to find the status of submitted applications:*
>
> *% status remote*
>
> *Program.z is running on 1024 processors on asci.lanl.gov*
>
> *That is my vision of a no-hassle grid computing where the user does not have to worry about where the programs will be run and on what kind of architecture."*

	Do these models correspond to what you had previously heard about Grid technology?		***Could your research group benefit from a Grid network?***	
	All replies	*Experienced users only*	*All replies*	*Experienced users only*
Yes	66 (79%)	15 (88%)	77 (91%)	16 (94%)
No	8 (9%)	2 (12%)	3 (4%)	1 (6%)
No answer	10 (12%)	0 (0%)	4 (5%)	0 (0%)

Table 7: perceived features and benefits of Grid computing

Despite the somewhat lukewarm verdict expressed in Figure 24, page 128 (overall, only 52% of the respondents found these models 'useful' or 'very useful'), it is reassuring to see that so few group leaders thought that their group would not benefit from one way or another to access a Grid. The comments they made provided many insights on users' expectations. The comments fell into seven main categories (the number in brackets indicates the number of related comments on this topic):

1. CPU time (9);
2. addressing large-scale problems (8);
3. shared code/packages (7);
4. visualisation and analysis tools (7);
5. shared /distributed data (6);
6. large data stores (4);
7. security concerns (2).

However, the key benefits revolved around only four central points:

1. Need for standardisation: respondents pointed out the benefits of a uniform interface to access remote resources, preferably also including access to local resources. Only two people explicitly specified that this interface must be GUI-based, although this point might have been seen as obvious for other respondents (such as users accessing the Grid from a Microsoft Windows desktop system).

   Standardisation efforts now constitute one of the key objectives of the GGF so the benefits will soon become apparent to service and application providers. On the other hand, users are more concerned about the 'standardisation' of higher level services (method of access and operation) than middleware. The accelerated development and standardisation of user interfaces has therefore to be encouraged, e.g. as part of the GGF Grid Computing Environment (GCE) research group. In addition to this, the status of Globus as a de-facto standard has caused some concern in the user community as many (wrongly) expected it to be a monolithic, self-contained system with a user-friendly interface and advanced built-in features such as multi-criteria scheduling. Such user-friendly and fully featured interface between the Grid middleware and users is sadly still missing although competing approaches (such as Unicore) are a step in the right direction.

2. Ease of use: by far the most popular point, this concept was considered in its wider meaning. Beside being user-friendly, the system has to take care of portability issues, be capable of operating in heterogeneous environments, provide seamless access to virtual (i.e. distributed) datasets, manage automatic data migration for visualisation, be stable, suitable for computational steering and offer fully-featured debugging and performance analysis tools. One of the respondents, probably also involved in co-ordinating his/her group's resources, suggested that the Grid ought to make accounting, user administration and resource management tools more user-friendly.

   The importance of this concept has been given different priorities depending on the type of Grid under scrutiny. Indeed, whilst the Data Grid community has achieved great strides in developing user-friendly interfaces for the retrieval and analysis of distributed datasets via Web portals (e.g., the CLRC Data portal [41]), the development of such systems for computational Grids is still in its infancy. Underlying issues to integrate such systems in computational Grids are obviously more complex as they have to deal with potentially many more site-specific and application-specific parameters and configuration issues. Despite these difficulties however, a number of prototype portals for computational grids are currently being developed. For instance, EPCC is currently developing computational portals for the Virgo and MHD consortia [42]. Other similar projects have been undertaken as part of the GGF GCE research group. A good example of this is the Gaussian98 portal developed by (NCSA) and the Grid Access Portal for Physics Applications (GRAPPA)'s Science Portals Project [43]. The 'Grid Resource Broker' (GRB) [108], co-developed by the HPC Lab of the University of Lecce (Italy) and CACR of the California Institute of Technology, is another example of ubiquitous Grid computing framework.

   More mature examples of such systems also exist in the Data Grid world. Thus, the San Diego Supercomputing Centre's (SDSC) Biology workbench [106], which allows biologists to search many popular protein and nucleic acid sequence databases is a good illustration of this development. In this system, database searching is integrated with access to a wide variety of analysis and modelling tools, all within a point and click interface that eliminates file format compatibility problems.

   It should be noted that the presence of a GUI is particularly important considering the increasing number of prospective Grid users accessing the Grid from a Microsoft Windows environment (a surprisingly high 36% of the respondents).

   Portability issues are equally crucial. Although it may be argued that application codes developed following the standards for C or FORTRAN are fully portable, this would be overlooking the fact that many legacy codes currently in use do not strictly abide by these standards and need to be extensively tested on each new platform. The key to solving this problem is to accelerate the development and encourage the use of integrated systems such as Cactus or high-level libraries such as PetSC [44]. The situation is expected to improve with time as the new generation of computational scientists are taught Java as a first (and main) programming language [34]. The final point worth a mention is the slow but steady uptake of computational steering. Computational steering has previously been defined as the ability to visualise the data from a computation in progress and to modify the future behaviour of the computation in response to this. As graphics workstations are getting increasingly affordable and as the bandwidth between HPC centres and their users' desktop (end-to-end bandwidth) has increased orders of magnitude from tens of kilobytes per second only a few years ago to tens of Megabytes per second now (see "Section 5: Infrastructure," page 170), an increasing number of research groups are now eager to benefit from remote visualisation and computational steering. A very good example of efforts to develop a suitably integrated computational steering environment is illustrated by the UK EPSRC-funded Reality Grid project [9] which extends the concept of a virtual reality centre across the Grid and links it to massive computational resources at HPC centres and experimental facilities.

3. Collaborative environment and sharing resources: the term 'resources' is here also taken in its wider context to include not only compute cycles, but also access to (distributed) datasets, tools, simulation codes, and packages. Although some of these practices are already under way, e.g. sharing of simulation codes through community based initiatives such as the UK Collaborative

Computational Projects (CCPs) [45] or through various libraries such as the Computer Physics Communications programs archive [46], other aspects are still in their infancy. Thus, sharing packages will require license issuers to provide more flexible/dynamic licensing schemes than those currently in existence in order to allow 'floating licenses' across a Grid. For instance, requests for a package/license may have to be specified in the job's resource description language in the same way as one specifies a requested number of PEs or memory when submitting a batch job. Software would thus become a resource at the same level as memory or PEs. An interesting development in this context is the advent of repositories of Grid-related material like that developed and maintained by Daresbury laboratories [47].

Just as for simulation code repositories, the concept of sharing datasets is not novel and many consortia, e.g. in climate modelling, have been participating to this type of scheme for quite a long time. For instance the OCCAM consortium [48] have been making their data available through the Web for a few years. Catalysed by the expected success of the Datagrid project, the data Grid community is now multiplying its effort to set up topical data repositories for most major disciplines (see for example the AstroGrid and GridPP projects [49]). It is worth pointing out that these efforts extend far beyond the traditional HPC community to include experimentalists.

It is also worth mentioning the great progress made as part of other Data Grid activities worldwide. This is for instance the case of the 'Data Intensive Distributed Computing Research Group' which is involved in the development of architectures for cost effective high-performance data-intensive distributed systems, and mechanisms for managing and distributing huge scientific experiment data sets [104]. Such objectives require high throughput solutions that are 'network aware' and scale to the wide area. Tools for monitoring and analysing distributed system performance have also been developed and integrated as part of the 'Distributed Monitoring Framework' (DMF).

Systems capable of placing PIs in a situation to carry out the management of their own resources by themselves are already in existence at some European HPC sites such as CSAR in the UK [50]. This type of system, whilst still suffering from the slow latency characteristic of accessing Web based services, does already include most of the features allowing the sharing of computational resources across a number of systems and users. For instance, a hierarchical structure can be set up with independent sub-groups, and resources can be easily transferred between users, sub-groups, and even external groups using a system of generic tokens to convert between heterogeneous resources (CPU cycles, disk quotas, support staff etc). Such systems form an excellent basis for the administration of computational Grids. This effort is already underway, and steps have been already undertaken by the GGF 'Accounting' working group to publish recommendations advocating this approach [51].

Another interesting aspect worth considering is that this sharing can be implemented in a fairly flexible manner; it can either take place within a collaborative framework, or without any contacts between the groups or individuals concerned, e.g. by exchanging resources using a trading pool system. However, the fact that so many respondents expressed the wish to increase their opportunity to establish new links with remote colleagues to form new collaborations indicates that both methods are complementary and therefore likely to co-exist.

4. Optimised use of existing compute resources: this is the final recurring theme of this section. This optimisation takes several forms: first, through cycle stealing and load sharing systems and secondly, by ensuring that jobs are re-directed to the most adequate systems.

    This latter point is probably the most crucial. Many instances have been reported when policies in force at some HPC centres favoured throughput (capacity computing) rather than Grand Challenge applications (capability computing). It is common sense that small jobs (<16PEs) and bandwidth parsimonious / latency tolerant applications (e.g., using trivial parallelism) should be prevented from clogging up high-end HPC systems and be re-routed towards more adequate, low-cost systems, such as Beowulf clusters, medium size departmental servers, or even clusters of workstations. This use of the Grid alone would have a considerable impact on the throughput of true Grand Challenge applications.

    This point highlights the need for generic large-scale computational Grids to include complementary hardware capable of covering applications with varying levels of requirement, rather than a simple collection of high-end systems. Typically, such a Grid would be composed of a mixture of high-end systems (large MPP and/or shared memory systems) and smaller enterprise-type servers and Beowulf clusters. Serial pre/post-processing jobs with large memory requirements are sometimes unavoidable, but they should be run on dedicated systems rather than on large shared memory systems.

III.3.4 Possible scope of computational Grids

The 'Grid' concept is very versatile. Experience has shown that early debates in the HPC community were made particularly difficult by the fact that most interlocutors had a different understanding of what this term encompassed. This phenomenon was even noticed during the ENACTS kick-off meeting and at GGF3 where many speakers categorised Grids according to different criteria. Ultimately, there is no such thing as a universal definition of 'the Grid' and one should speak of 'Grids' rather than 'the Grid'.

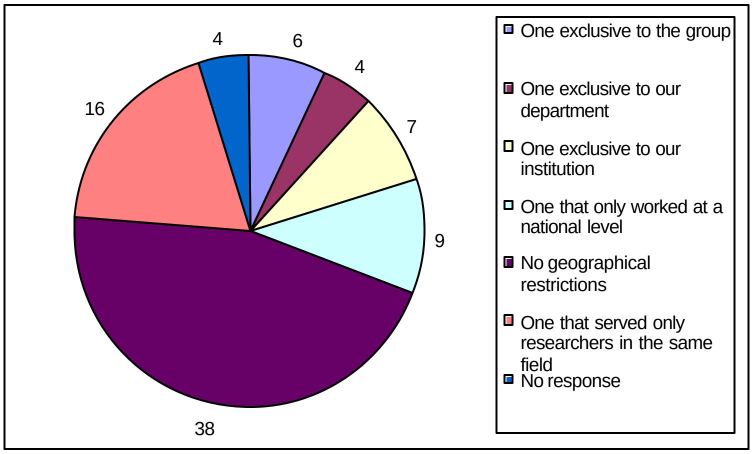


Figure 26: preferred scope of Computational Grids

Besides the traditional taxonomy made according to the purpose of the Grid (e.g., Data Grid, Access Grid, Computational Grid, etc.), one may also categorise Grids by their access restrictions. The issue of access restrictions to a Grid may at first look secondary but it is in fact a crucial point as it is directly dependent on the funding of the infrastructure. For instance, one would envisage that compute resources owned by a University may be available to researchers from this same university through a local Grid, whilst facilities funded at a national level would primarily be available to researchers working within that country. Of course access from third-party may be allowed either through active collaboration with an authorised research groups, or through other arrangements involving trade of resources or financial arrangements. The only aspect which is slightly novel is the concept that groups working in the same field may decide to set-up international Grids for exclusive use by researchers working in the same field. This option seems very attractive for groups enjoying a high-level of co-ordination, typically the particle physics and astrophysics communities, and/or for access to specialised hardware. Examples of such collaborations are already common within these particular communities, and Grid projects such as Astro-Grid, GridPP etc. [49] already confirm that successful collaborations established for access to HPC systems and that collaborative research projects will be extended to the Grid level for the sharing of resources. This concept is called in this study 'topical Grids'.

A number of respondents raised important issues associated with the deployment of international Grids. For instance, security, liability and overheads are extremely important issues for which little or no answers have yet been provided. The issue of security has been discussed at great length in Part II and is actively discussed in the GGF Working group 'Grid Security Infrastructure (GSI)' and 'Grid Certificate Policy (GCP)' [52]. Interested readers should refer to the GGF's material for an up-to-date progress report on this issue.

The issue of liability and accountability is altogether more complex. In an international infrastructure where participating sites may be operating according to different economical models (e.g., owned, fixed cost publicly funded, or Private Finance Initiative (PFI)) and policies, the issue of charging, accounting, and liability is non-trivial. Different policies may make some use of a resource that is perfectly legitimate at one participating site and illegal at another? This phenomenon has been observed on a few occasions by EPCC through its EC-funded visitor programme TRACS [7]. For more serious breaches of security occurring through hacking or deliberate abuse of the system, the prospect of legal prosecutions may be too complicated to be a true deterrent across national borders. Liability itself is a complex issue which will be discussed later in this section. The bottom line is "who can be held responsible when something goes wrong?". This point is crucial, especially when considered within the context of Quality of Service (QoS, e.g., bandwidth reservation for guaranteed performance and turn-around time) which is a pre-requisite for a likely use of Grids by business and industry. A system based on financial compensation for breach of contract (e.g. if a job fails to complete by a certain time) will be required in order to satisfy this category of users. Grids will need to evolve rapidly towards a fault tolerant system (possibly with automatic checkpointing) capable of performing dynamic (agent-based) resource discovery for job migration and automatic restart. This particular topic is currently being investigated as part of the Gridlab project.

Operators of HPC national services usually commit themselves to comply with a service-level agreement (SLA) which defines the minimum level of service to be provided to their users. Whilst a similar concept can be envisaged for Grid infrastructures, its implementation will require more negotiations amongst the parties involved (funding agencies, service providers and the user community). A number of mundane issues will need to be discussed, e.g. should responsibilities for user support be assigned to a single site or distributed across participating sites? If so, which criteria should be used to decide which site will have the responsibility for a user X running a job Y on a system Z? Maybe the solution is a policy in which users are 'allocated' to participating centres according to their research area. Participating sites would thus put together topical user support teams. Alternatively, it may be better to provide support through the local HPC centres that users already know and trust by establishing a network of regional Grid helpdesks such as that being developed in the UK [53]. Variations on this theme are too numerous to be discussed further in this report.

On a more pragmatic level, the issue of overheads must not be underestimated. These overheads can be caused by the simple increase in user administration tasks (registering users, creating accounts, issuing and managing public key certificates etc.), but also those associated with the deployment of a more complex infrastructure. As the average level of utilisation of HPC resources in Europe is noticeably higher than in North America, many representatives of European HPC centres were concerned in the early days of the GGF that the underlying issues of accounting, charging and scheduling may not given the importance they deserve. Indeed, strategic orientations in working groups sometimes gave the impression of being controlled by American research groups. This worry was not dispelled considering the presence of 26 US Chair persons (out of 31) at GGF2. European centres must now be encouraged to adopt a more concerted action to design a list of requirements and specifications for smart automated systems for charging, accounting, and scheduling within a Grid environment which will take into account the specific environment in which they have to operate. The ensuing challenge is to make sure that these specifications get taken on board by the relevant working groups. The most appropriate way to do this is to participate actively to the relevant GGF working groups and if appropriate, propose the nomination of a suitable European representative from this interest group as a co-chair in relevant groups.

As was emphasised by several participants, one of the pre-requisites for accounting and charging in a Grid environment will be the adoption of a common currency to trade resources. Rudimentary systems are already in operation in some European HPC centres [51,54] and could be extended to meet the requirements of a Grid environment. However, one crucial aspect currently missing from these systems is Grid economics. In a dynamic environment where new resources may come and go and the level of fluctuation in the availability of specific resources can be very high, the actual unit cost for a given resource is also expected to fluctuate to reflect the fundamental market rules of supply and demand. However, this approach will have serious implications as all Grid nodes will in effect be placed in competition with one another to provide resources. As many sites will be funded according to different schemes (possibly involving profit-making organisations), this competition may generate more problems than benefits for the user community, who require stable environments. It is therefore essential for funding bodies to establish smart rules which will reward the use of supercomputing systems for capability applications rather than pure capacity. Using raw CPU usage, as is commonly the case, is inadequate since it does not take into account the suitability of a job for the system in question. Further criteria, such as requests for a certain QoS, special services (e.g. software licenses) etc. will also need to be taken into account in order to use Grids effectively.

Whilst some groups may be ready to allow others access to their systems when they are underused, they are equally determined to be given the same opportunity when they are a situation where resources available to them are insufficient. The following comment perfectly illustrates the need for working towards a common Grid currency and smart charging mechanisms:

> *"In my ideal computer world there would be an intelligent server system that would only send jobs to appropriate computers and if anyone would use my workstation via the grid I would earn credits that I could use later when I have a burst of activity and need to get many results fast on computers anywhere in the world.*
>
> *It might be a nice gadget to see my computing credits being filled up when someone is running a job on my computer."*

As was pointed out by one of the respondents "*different levels of Grid functionality are required for different purposes*". The solution is therefore to have a mixed, hierarchical organisation of such Grids. Some may be more suitable for one purpose (e.g., international Grids to access large distributed datasets and high-end supercomputers for the most demanding Grand Challenge applications), and others for some other purpose (e.g. code development). This concept of differentiated services on specialised Grids definitely seems to offer more benefits.

The motivations for choosing a particular type of Grid were quite varied and many respondents expressed an interest in more than one type:

> *"Two levels are interesting: one at group level with tight and adjustable shared task grid, and another one primarily to store data in standard form world-wide, and perform standard tasks.*
>
> *Other types are also possible but only in more special situations."*

However, the majority of users (45%) picked a Grid 'with no geographical restrictions' as the most suitable for their group. No doubt the administrative and political overheads to set up such Grids will be considerable unless the providers actually 'owns' the resources, which is quite common for groups in High-Energy Physics (HEP). The basic reason for this is simple:

> *"We are interested in getting calculations performed as fast as possible. It doesn't matter where the results are calculated."*

… although other respondents expressed reservations based on the quality of the link to the remote site and possible problems with time-dependant access policies (which are fairly typical):

> *"I don't have strong geographical preferences as long as there is a reasonable fast (interactive) access to the machine*
>
> *if the machine is in a different timezone it allows for interactive testing during normal times at my place."*

The limited support for smaller Grids e.g. within a research group, department, or university is principally due to the lack or inadequacy of resources within this structure:

> *"Would be useful for the group or department level only if the hardware available were increased an order of magnitude."*

Going one step up, the national Grids may be the answer to overcome this localised starvation of resources and provide a more homogeneous access to compute resources. Of course, this ability will be greatly dependent upon the source of the funding and associated contractual obligations.

> *"At an European-wide level, we should create Consortia or pool of structures to be deployed in the most 'transparent' way from the end-user communities. This with a (primary) aim of better using European computing resources and to decrease the impact of the lack of large computing resources in some area of computational science."*

National Grids established from separate university Grids may only form a consortium of universities with no access to researchers from outside these universities. The role of the funding agencies will then be to legislate and co-ordinate access to these Grids. An added benefit of these national distributed Grids is that they will form an excellent ground to promote code sharing across research groups.

Large institutions such as CNRS, INFN, the Max Planck Institute etc. can actually benefit from the best of both worlds: fewer administrative and political overheads than independent sites, but enough resources to allow building a meaningful Grid:

> *"What I actually have in mind is a grid for the Max Planck Society, for all of Germany. I think the problem which should be kept in mind when grid technology is developed is that of funding. I can hardly imagine a future in which cycles are 'just there', and all you need is a technology to get access. Rather, at least in the high-end sector, there will be an institution who PAYS for the cycles - actually substantial amounts of money. Thus, such a system has to be able to do some sort of accounting. I fear that a system which is too open will simply be too difficult to handle, and generate too much overhead when institutions start to not only share, but actually TRADE cycles - but maybe I am wrong, and all this can be done easily. In that latter case, probably the maxim 'the bigger the better' will work best."*

This respondent raised one of the key issues that many choose to ignore: CPU cycles (amongst other resources) cost *real* money. The prospect of resource owners simply 'giving them away' is not realistic, especially if this philanthropic gesture results in generating administrative overheads and raising security concerns. Judging from some of the replies from the less well resourced respondents, the concept of Grid seems to cultivate the wrong impression that more resources will become available… for free.

One of the strongest points emerging from the comments is that the user community are cautious and will require access to proper testbeds before giving their support to any particular type of Grid. A sensible approach to this problem has been described as follows:

> *"Initially there will be some geographical restrictions. We start from local access only first (testing purpose), but we will gradually relax that restriction."*

This is probably the key to setting up successful testbeds: to encourage large universities to set up a local inter-departmental Grid. Administrative and political overheads will be quite low, security issues easier to manage, and users more likely to adopt this technology if it is controlled and managed by their own institution. Furthermore, technological requirements, principally networking, will be easier to understand within a single institution. This 'Win-Win strategy' has the following benefits:

- It provides local computing services and support teams hands-on experience with the underlying technology (installation, configuration, support, etc.), thus preparing them to get involved in larger scale Grid project at a later stage; and
- It provides users with a new environment which will 'feel safe and comfortable' as both support and decision making will be taken by local (well-known) groups and committees.

These two facts will make an ideal stepping stone towards getting these local Grids to merge with similar ones at neighbouring universities in due course, with minimal fuss for both the support teams and the local user community. The more delicate question of how to encourage universities to set up such projects and commit some of their resources to a Grid infrastructure is discussed in Part IV.

III.3.5 Provision for interactive access within a computational Grid

The provision for interactive access is a topic which often divides the HPC user community, and is equally expected to cause some controversy in the context of a computational Grid.

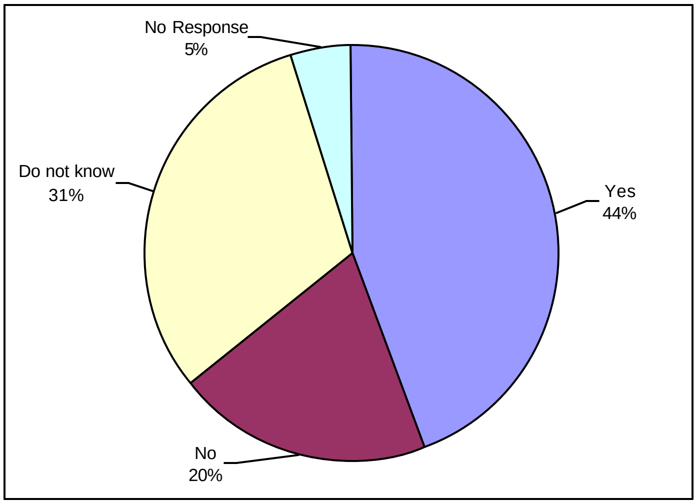


Figure 27: user requirement for interactive access

Based on the respondents' feedback, the motivations for interactive access can be split into six distinct categories (the number between brackets still indicates the number of related comments):

1. during the development cycle: compilation, debugging, tuning, validating (17);
2. for accessing and processing data (7);
3. for interactive visualisation and computational steering (4);
4. required by some applications such as CAD packages, MSI products, Matlab, etc. (2);
5. for access to some services (2); and
6. to build confidence in Grid environments (1).

The solution put forward by a number of these users is to reserve a pool of systems available for interactive access. The main motivation is of course to simplify the development cycle. Although remote compilations via a Web system are conceivable, they would not provide the same level of control as true interactive access and so would be an obstacle to a purely batch oriented system. Debugging and short test runs are also noticeably more difficult to carry out in a batch environment as turn-around times are often unknown. Users of packages and stable codes (the so-called 'black box users', see discussion of user profiles in the next section) are outside the development cycle and could therefore make use of pre-compiled binaries. In such an environment, the service providers must commit themselves to support a number of named applications and packages for which they would provide pre-compiled binaries (a minimum set would be generic 'debug' and 'optimised' versions) to users of their facilities through integrated portals. Another restriction that must be observed during the development cycle is that all its stages must be undertaken in a stable environment, i.e. with a given system type, operating system, and programming environment (libraries, tools, etc.). This is also essential to achieve repeatability whenever required (e.g., non-deterministic bug, optimisation, etc.).

The problem of data processing is slightly more difficult as some analysis procedures require human interaction and are therefore unsuitable for a batch environment. It appears obvious that Grid systems, including the main production system, must provide some level of support for interactive jobs as debugging, tuning and validating may not be undertaken on a different system. The use of Grid economics (e.g., differentiated charging cost for interactive and batch access) could be used to dissuade abuse of the systems earmarked for interactive use. It should be noted that the issue of interactive use of the Grid is taken seriously by the GGF and is presented as one of the central research topics of the ACE working group. Indeed, as specified in its own charter:

> *"The Advanced Collaborative Environments (ACE) Grid Working Group complements other Grid Working Groups by providing human-centered techniques and technologies for facilitating interactive, collaborative, and immersive access of Grid resources from any where and at any time."*

III.3.6 Prior experience with Grid-enabling technologies

The data presented in Figure 28 provides further information about the nature of the prior experience of respondents who described themselves as users with *some* experience of Grid technology (see Figure 22, page 125).

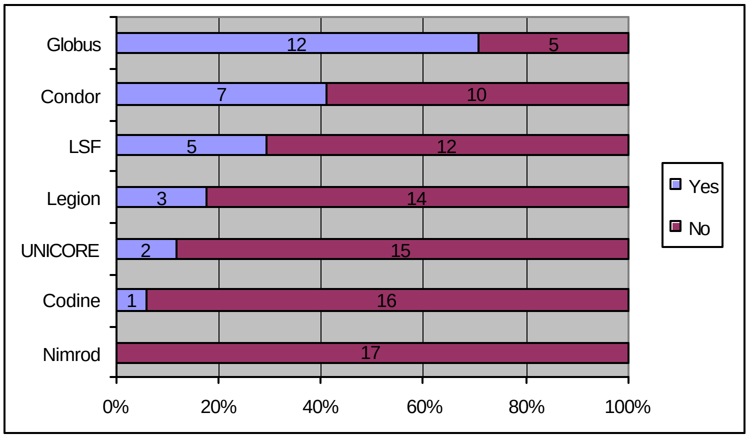


Figure 28: prior experience of various Grid-enabling technologies

The first point to mention is that these results are slightly different from those obtained when considering the full 85 replies. Thus, whilst the number of answers for Legion, UNICORE, Globus, and Condor are roughly identical (one more vote for UNICORE and two for Globus), a much stronger discrepancy can be observed for Codine (6 votes) and LSF (13 votes). This variation relates to the fact that these two systems are primarily used as scheduling systems for clusters of servers, which many respondents did not regard as being Grid related per se. This is another example of the difficulty posed by the different understanding of what constitutes Grids and Grid technology! Codine's new name, Sun's Grid Engine, will clearly re-position it as Grid middleware. Several other technologies were also mentioned. These include Cactus, NQE, IBM load-leveller, Java RMI, PVM, VAX OS, OpenPBS, and *"own solution based on Java RMI"* (GRISK project [55]).

The comments made by respondents about their experience of these technologies were rather varied. A consensus emerged amongst Globus users that is was potentially promising, but definitely not ready for production use[2]. Indeed, this middleware was described by some users as *"having an untidy design and to be user-unfriendly"*, and to be *"very limited"*, *"very low-level and restricted"*, and *"time-consuming to set up"*. On the other hand Condor and UNICORE received only praise. The relatively low level of experience with UNICORE was surprising. This is even more unexpected considering that only a single German respondent (out of 13) had had a prior encounter with UNICORE. With the noticeable exception of the INFN Grid [56], which is already used for production work, most other groups were mainly experimenting with this new concept and technology. The ENEA Grid [57] also appears to be a mature environment:

> *"We have adopted a scheme with LSF and AFS, where all the resources of ENEA computing centres are related and let available by a GUI which allows to select the resources on the basis of the evidence of their current*

[2] These comments refer to Globus v1.x. Globus v2.x (still in Beta at the time of writing) has undergone major improvements which make it more robust and easier to install and configure.

> *status (and use). This is a very simple and immediate tool which greatly helps the use of these resources."*

Progress with large scale projects is difficult as Grid testbeds involving multiple sites is hampered by the administrative and political overheads described earlier. As one of the respondents said:

> *"There is no easy use so far. Effectiveness is hampered by too much of administrative problems between sites."*

III.3.7 Key features of computational Grids

There was some discussion earlier in this chapter as to how best to combine various architectures in order to fulfil particular needs of the user community. In fact, this discussion should be extended to include not only basic hardware considerations but also configuration issues which impact both the job throughput and the average turn around time. The respondents were therefore asked to rank the importance of these factors in a Grid environment. The results are shown in Figure 29, page 141, where the factors sorted by order of importance using a simple (but arbitrary) weighting scheme:

$$W = 1.5 \times n_{very\_impor\tan t} + n_{impor\tan t}$$

Based on this classification the factors were classified and ranked as follows:

1. primary resources: faster CPUs, more PEs, and more memory;
2. infrastructure policy and capability: longer run-time, ease of use and throughput;
3. secondary issues and hardware features: system (non-)availability, turn-around time, memory per PE;
4. production run environment: large storage devices;
5. minor issues: security of code/data, I/O performance, etc.

The 'primary' resources (performance, number of PEs and memory size) are by definition the limiting factors for our ability to solve Grand Challenge problems. It is therefore not surprising that they came top of the list in this particular order. The need for faster CPUs (and implicitly higher memory-to-CPU bandwidth) was particularly noticeable. The good news is that this acute need for increased resources could indeed be partially solved by computational Grids. Whilst the availability of faster CPUs is essentially dependent on technological breakthroughs (e.g. new developments in semi-conductor technology and computer architecture), the greater availability of memory and PEs can be solved for example by accessing remote resources with the required specifications, by forming metacomputers, or simply by getting small jobs off the high-end HPC systems so as to reserve them for Grand Challenge applications.

The second category consists of justifiable grievances from the user community, mostly caused by policies commonly implemented at some computing facilities. The time windows (preventing long run-times) are indeed purely a restriction arising from the infrastructure operator. Whilst some sites adapt their limits to requests from the user community, others sadly do not. In past years, users were simply expected to run their simulations on a larger number of PEs to allow their simulation to complete within a certain time window. Whilst this practice was usually acceptable with distributed memory systems like the T3E on which applications typically scaled reasonably well, the same is not true of shared-memory systems which now constitute the bulk of HPC systems. Their design intrinsically places a bottleneck on scalability beyond 64PEs.

The concept of 'ease of use' refers to difficulties in using the facilities, principally how to develop applications (overhead having to learn a new programming environment for each new system) and submit jobs (different queuing systems with cryptic flags). It would be a major issue for most users if the interface developed to access the Grid was any more complex than the current interface to HPC sites. This fact explains the considerable hostility of users towards Globus, which is regarded by many as far too unfriendly.

Throughput is a more complicated problem to solve altogether. It is a fact of life that some groups will go through burst of extreme activity followed by more quiet periods. It is therefore unavoidable that this will result in job congestion and a less than optimal throughput for all the users. Grids may be particularly useful in this context as operators could have the option to re-route/migrate particular jobs (e.g., jobs explicitly marked as 'urgent' and allowed by their owner to run off-site) whenever their expected starting time is delayed beyond a critical level and thus are at risk of not completing by the deadline specified. Once again, 'filtering' small jobs off production systems may significantly improve the overall throughput of Grand Challenge and time-critical applications.

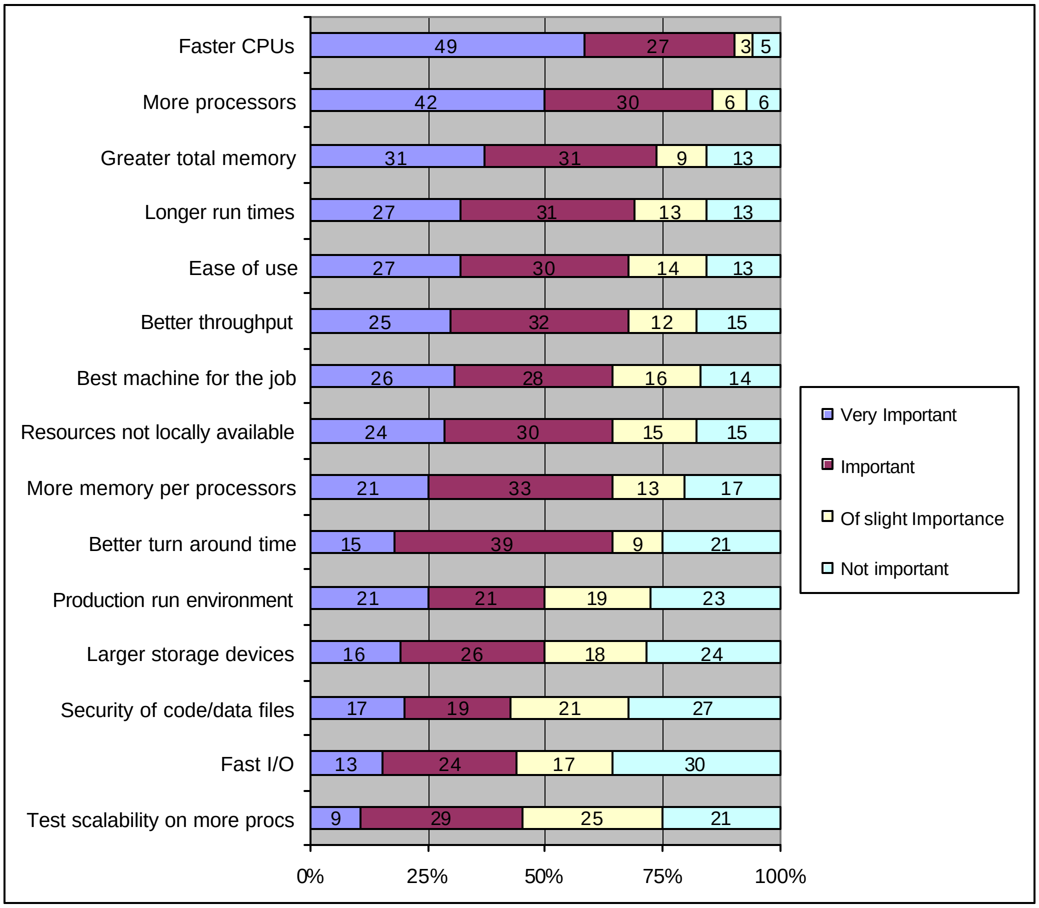


Figure 29: importance of various factors in a Grid environment

The third category principally touches on the concept of (lack of) access. The purpose of computational Grids will also encompass provision for accessing computer architectures (or configurations) which are more appropriate for any particular type of application. This advocates once again the creation of Grids composed of a variety of architectures, or alternatively the possibility of migrating jobs between collaborative Grids.

No evidence emerged from the data in the ranking of these factors depending on which type of Grid the respondents had selected as 'Useful' or 'Very useful'. This was somewhat surprising - for instance, it would be expected that 'longer run times' was more of an issue for the portal and toolkit approaches then for say, peer-to-peer and Internet computing. .Some significant variation was expected

Finally, it should be noted that the importance of these factors differed considerably for industrial users. Indeed, results from a study undertaken by Datamat [58] established that industrial users have the following requirements:

- low cost/high bandwidth network access;
- QoS;
- security (authentication, non-intrusion); and
- long-term secure storage.

A short discussion on specific requirements of industrial users has been included in "Section 3: User profile," page 146.

III.3.8 Most wanted resources in a Grid environment

The access to remote resources was regarded as important for the usefulness of computational Grids (see Figure 29, page 141). The actual motivation for such resources is summarised as in Table 8:

Evaluate performance	26	31%
Test portability	14	16%
Use only once	6	7%
No response	39	46%

Table 8: use of remote resources

A closer look at the comments made by the 'No response' revealed that their most common use (16 respondents out of 39) would be for production runs. In addition, the following needs emerged:

- access remote programs (including Grid-based computational tools);
- access to a faster machine for computations (large parallel computers or systems with better characteristics, e.g., larger aggregate memory) and metacomputers;
- data processing using large shared memory systems (including access of large distributed archives);
- visualisation;
- access to high-end HPC systems; and
- access for very large (not every-day) computations.

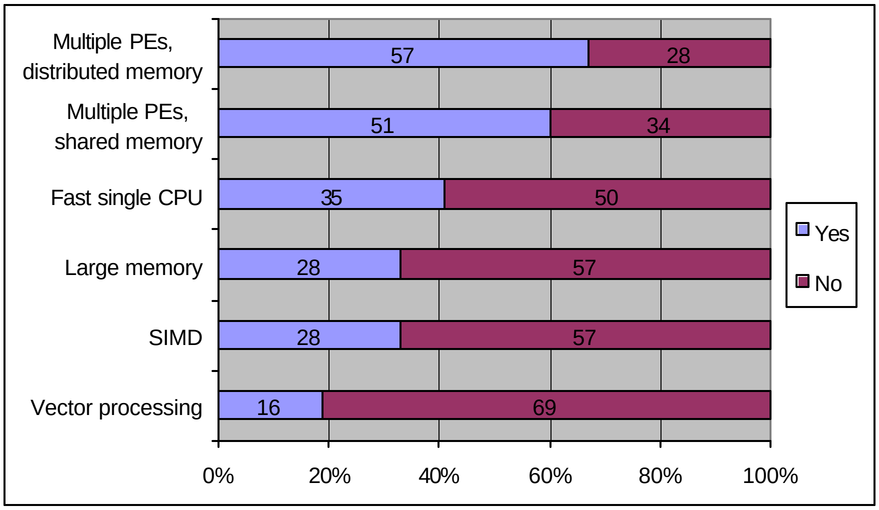


Figure 30: preferred platforms in a Grid environment

Figure 30 shows that the platforms of choice are large parallel systems for production runs (distributed memory systems in particular because of their better scaling capabilities), and large memory systems for data processing. It is actually surprising that 'old-fashioned' architectures such as SIMD and Vector computers are still rated quite highly. Their popularity can probably be explained by the cost-effectiveness of purpose-built systems such as the APEmille which can deliver Tflops for specialised applications at a fraction of the cost of conventional systems [59]. The role of such systems in a computational Grid is therefore likely to be restricted to topical Grids – a remark which also applies to custom-built systems such as the GRAPE [60].

These data provide yet another confirmation that Grid infrastructures should not only provide access to high-end parallel supercomputers, but also to suitable (large memory) servers configured to support sequential jobs for data processing. Unnecessary network traffic could thus be avoided by carrying out on-site data processing.

In the great scheme of things, as one respondent put it:

> *"It [the type of architecture available] should be irrelevant."*

III.3.9 Grid computing and Grand Challenge problems

The emphasis for Grid testbeds in the UK has been placed on enabling terascale computational science and engineering. The Grid is seen as the means by which the next generation of Grand Challenge problems will become tractable for an increasingly larger fraction of the research community. When asked if Grid computing will influence them to tackle larger scale problems, the majority of respondents confirmed that they do indeed expect this to be the case. Another point worth pointing out is that a large proportion of respondents were undecided, which suggests that many researchers are either not clear about what Grids will eventually enable them to do or are simply cautious about its ability to deliver. As was emphasised earlier, many users expressed the wish to get access to a testbed before deciding what attitude to adopt towards Grid computing.

Yes	60	71%
No	8	10%
Do not know	14	17%
No response	2	2%

Table 9: use of the Grid to tackle 'bigger problems'

As before, the comments provided further insights into this issue. Ranked by the number of comments, users anticipated that the new capabilities enabled by Grid computing will allow them to:

- increase the system size, use finer meshes, or more complex potentials to simulate more realistic systems (23);
- study new, more challenging problems (16);
- coupling models on heterogeneous hardware (8);
- perform comparative and distributed data analysis, distributed data mining (5);
- overcome HPC site size and time limit, access cheaper resources, and cope with outburst of activity more efficiently (5); and
- get near real-time response to critical applications (1);

Of course this classification was not as straightforward to make as previous ones since they were often required interpretation of the replies. The numbers enclosed should therefore regarded as only an indication. A number of noticeable variations were also noted between the various research disciplines. For instance, the concept of coupling models (typically on heterogeneous architectures) is one which emerged from virtually all the respondents involved in weather modelling or meteorology, e.g.:

> *"We would be encouraged to couple ocean, atmospheric and bio-geochemical models at high-resolution possibly each running on a different platform."*

and

> *"Coupling of multi-component models for Earth System Science experiments."*

The perception about the level of maturity of Metacomputing was seen to vary significantly from the believers to the more pragmatic. For instance, two respondents made the following comments about this approach:

> *"Porting of a code on a 'mixed' architecture (MPP+Vector+specialised machines) in order to increase the level of computational efficiency reached on a given application."*

and

> *"MD simulations with very large number of particles, full QCD simulations on larger lattices.*
>
> *The question is: Would I spent a lot of time to execute these codes on connected systems or would I prefer to run these on one larger system? At the moment I would prefer the latter one, because it is too complicated to have a combination of systems combined for a longer time period (not to speak from data distribution)."*

The final trend to report was the large number of large-scale applications put forward in the biological area. Simulating complex biological phenomena (in a realistic manner) requires resources on a scale that even the better equipped HPC centres have difficulty in meeting. The prospect of 'the Grid' unleashing a new class of applications which could potentially allow breakthroughs in medical science is simply too important to be overlooked:

> *"Simulation of large biomolecules, in vivo simulation of biochemical reactions, computer assisted drug design."*

Thanks to its user-friendliness (portals) and versatility, the Grid will have the unexpected benefit of providing an environment suitable for new categories of users (e.g., genomics, medical research). This community predominantly stayed away from HPC facilities in the past because of the steep learning curve required to make use of their resources and the relative lack of power in the early days. This new category of users will expect to run their simulation in a similar environment to that offered by their everyday Windows-based application (this issue is discussed further in the next section).

A negative answer to the proposition of simulating larger problems should not be regarded as negative per se though. It appears that some areas of science are already tackling extremely challenging problems, which are simply big enough to stretch the concept of Grid computing to its limit:

> *"I believe that HEP provides sufficiently complex problems for the time being."*

III.3.10 Summary

Overall, the respondents showed a reasonably high level of awareness although large variations could be observed across national borders and scientific areas. This variation becomes even more noticeable when considering practical use of Grid-related technologies. The astrophysics community clearly stood out as the most Grid aware due to the combination of factors discussed in this section.

The preference for a particular type of Grid models as described in Part II is strongly correlated to the level of awareness of the respondents. Thus, whilst the least experienced favoured Grid portals, users at the other end of the spectrum were more in favour of Grid Toolkits.

It was particularly comforting to notice that over 90% of the respondents thought that Grid computing would enable them to advance their research, either through the increased provision of computational resources (CPU time, memory, etc.), or through improving access (visualisation and data analysis tools, etc.) and increased collaboration (sharing codes and data). This optimistic verdict was accompanied by a number of underlying pre-requisites which aim to implement Grids with an easy to use, standardised method of access, in which data and resources could be more readily shared with other researchers, and where existing resources would be used more effectively.

It was also noticeable that most respondents favoured Grids with no geographical restrictions, although most of them were equally fast to point out possible pitfalls of such infrastructures. Another popular concept was that of topical Grid, in which resources are shared amongst researchers working in a related field. The lack of interest for smaller Grids (at a University or departmental level) was explained by a chronic shortage of local computational resources. All in all, it emerged that a combination of these types of Grids would provide the best solution. Policy issues inherent to them were also discussed, with a particular emphasis on the influence of funding and co-ordination.

Provision for interactive use within a Grid infrastructure also proved to be a contentious issue. However, as justifications (such as code development and data processing) were considered, the motivation for building Grids composed of a mixture of complementary systems (both in terms of specifications and configuration) rather than like systems would provide the best framework to cater for these applications.

Provision for interactive use within a Grid infrastructure also proved to be a contentious issue. This was justified by the need for code development, data processing etc. Taking this into account, it seems likely that Grids composed of a mixture of complementary systems (both in terms of specifications and configuration) would provide the best framework to cater for this type of application.

Amongst those who reported an earlier use of Grid technology most had used Globus, with systems such as LSF, Condor, and SUN GridEngine also getting a noticeable mention. However, comments did in no way represent a confidence vote for the former, towards which many criticisms were expressed.

The key features required from a computational Grid were primarily those enabling Grand Challenge applications (CPU power, memory, etc.), followed by infrastructure policy and capability (ease of use, throughput and longer run-time). The most popular hardware resources were large parallel systems and fast (large memory) serial systems.

Finally, the vast majority of respondents expect Grid computing to enable them to tackle more challenging problems. This included not only the same research with larger or more detailed systems, but also new applications, especially in novel computational fields such as biological and medical sciences.

III.4 Section 3: User profile

The aim of the 'User Profile' section of the survey was to gain some understanding of users' application areas and level of applications development experience. This latter point will be a determining factor for the adequacy of the Grid models presented earlier in this document. This section summarises the user responses to the questions in this section, and correlations with findings from the previous sections.

III.4.1 Field of work and problem under investigation

The area of work of respondents is shown in Figure 31. Approximately 75% (64 out of 85) of the respondents work in chemistry, engineering, physics, astronomy or computing, which also constitute the bulk of traditional HPC users.

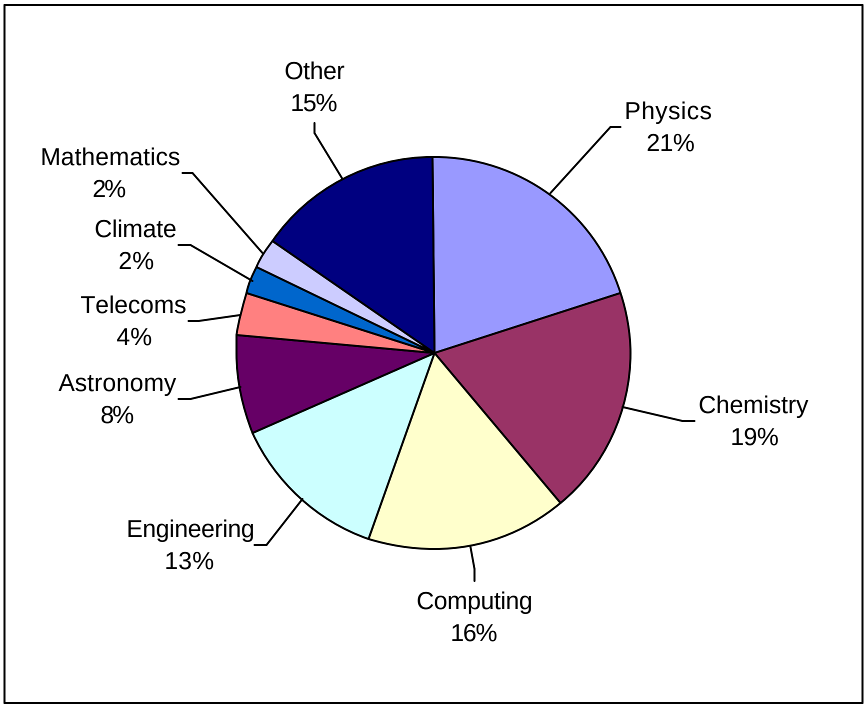


Figure 31: field of work

The 'other' category corresponds to categories represented by a single group (biochemistry and genetics), other fields of work (bioinformatics, geology, oceanography, user support, etc.) or to a category close to one of the main areas but judged sufficiently distinctive by the respondent to be described aside. This was the case for air pollution modelling, molecular modelling, atmospheric chemistry and meteorology, semi-conductor physics, etc. which could have been included in one of the ten main categories. Five respondents also specified being involved in multi-disciplinary research, e.g., *"a combination of physics, chemistry, and materials science"* or *"engineering science and computer science"*.

A closer look at the detailed description of the problems under investigation revealed a wealth of information about the respondents' research interests which cannot be exploited statistically. It was particularly important to try to establish a more accurate profile of the scientific communities that operators of Grid infrastructures and funding bodies will need to establish a dialogue with, especially in emerging and rapidly developing fields. In particular, new areas linked to medical research definitely could be singled out. Some of these applications include:

> *"Protein structure comparison and protein structure prediction:*
>
> *Genome analysis - annotation, comparison of genomes*
>
> *Microarray (gene expression) data analysis"*

> *"Protein unfolding and stability. My specific problem is studying the stability of the eye lens protein family gamma-crystallins, using several unfolding protocols. The objective being an understanding of the early events involved in cataract formation."*

> *"We are currently developing novel methods to cope with computationally intense problems in protein modelling, genomic and proteomic analysis, material science (at the atomic scale), astrophysics. We mainly deal with 'particle' models but, more recently, we have also tackled problems of sequence analysis and finite elements (for the simulation of EM propagation in different media)."*

One motivation for supporting this area of research effectively is the level of investment that the pharmaceutical industry would bring into Grid research and infrastructure if was felt that this technology could advance their research significantly. As many analysts have pointed out, in the same way that physics used to push the development of HPC in the nineties, biological sciences will do the same for Grid computing in this decade. A number of pilot Grid projects for biological sciences have since been put in place at a national and European level. Good examples of these include the MyGrid project [12], led by the European Bioinformatics Institute (EBI) and funded as part of the UK EPSRC's E-science pilot programme [25], the genomic application testbed developed as part of the EC-funded DataGrid project [20], and the GenoGrid project [61] in France.

Another interesting novel area is the development of Grid middleware and Grid related software technology:

> *"We are evaluating a platform that we built for developing application-enabled mobile agent systems."*

The Gridlab project [5] is a perfect example of such efforts.

The final important category is the area of distributed data mining, both for scientific research and for other purposes. The reasons for not overlooking this category of users are in some ways similar to those quoted earlier for medical research, as it is one of the most promising application areas in term of industrial relevance of Grid computing.

A complete transcript of all the descriptions is provided in Appendix D.

III.4.2 Features of mainstream research areas

A number of characteristic features were identified across research fields by studying the awareness level in the user community (for instance, see Table 5, page 126). The top five groups (physics, chemistry, computing, engineering, and astronomy) reached a consensus in recognising the potential benefits of Grid computing for their research groups – the lowest level of support was 93%.

A good example of these differing requirements was the need for interactive access as shown in Figure 32, page 148. The discrepancies observed between the two extremes (astrophysics and engineering) indicated a noticeable difference in working practices and requirements. Whilst the astrophysicists require interactive access, essentially for data manipulation and visualisation (3) and minor code tweaking and test compilations (3), users from the chemistry and engineering communities tend to use standard packages or applications (possibly with built-in visualisation tools) and are therefore not so dependant on interactive access. In fact, the justifications for interactive access from these two groups were typically non-specific, e.g.,

> *"Experience has shown that there will be situations where you need to intervene interactively."*

> *"Interactive access is more suited to build up user trust into the grid environment."*

or could easily be met with a suitable environment such as advanced portals:

> *"For compiling, unless we already have a binary version for that target machine."*

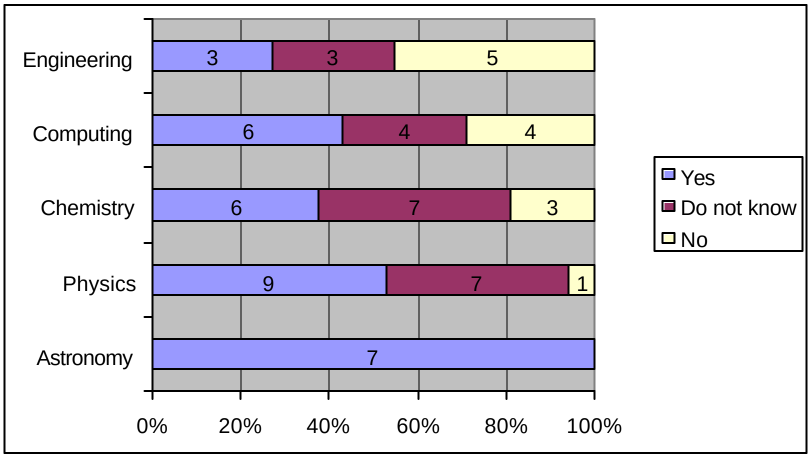


Figure 32: need for interactive access

This point highlights the fact that one configuration / policy will not meet everyone's needs, and advocates the needs for topical Grids whenever a community's need significantly diverge from others'. Note that the new user groups in the biological (and social) sciences will also expect their applications to run with an 'interactive feel'.

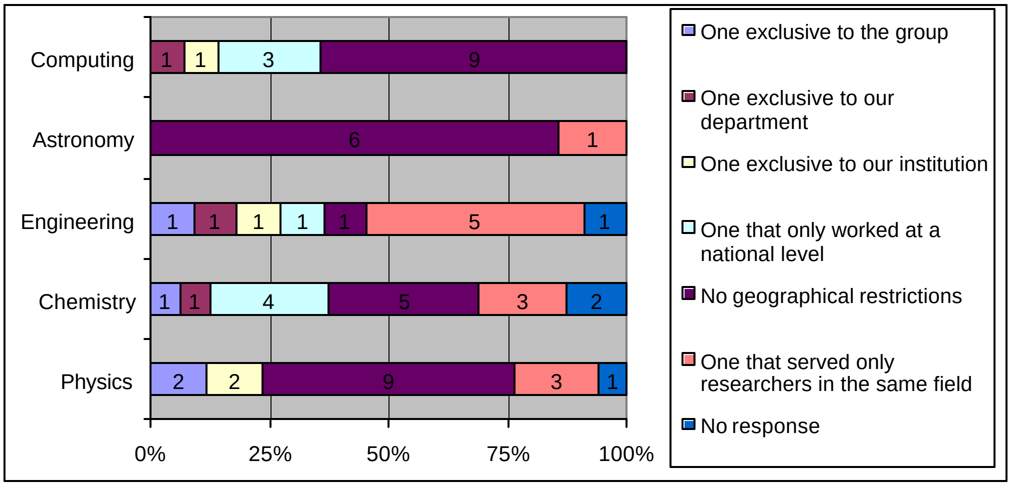


Figure 33: scope of preferred model

When considering the preferred scope of computational Grids, two groups stand out once again from the rest: astrophysics and engineering (see Figure 33, page 148). Whilst the former are more or less unanimous in favouring Grids with no geographical restrictions, the engineering community is more cautious and preferred smaller scale and topical Grids. The reasons for these differences are once again made clear by looking at the respondents' comments. Whilst the astrophysicists are already engaged in international collaborations, and are therefore more open to the sharing of application codes, data, and resources, the engineering community is engaged in smaller scale collaborations, mostly restricted to their own organisation (see Figure 34). Groups from the other three research areas followed a more balanced distribution.

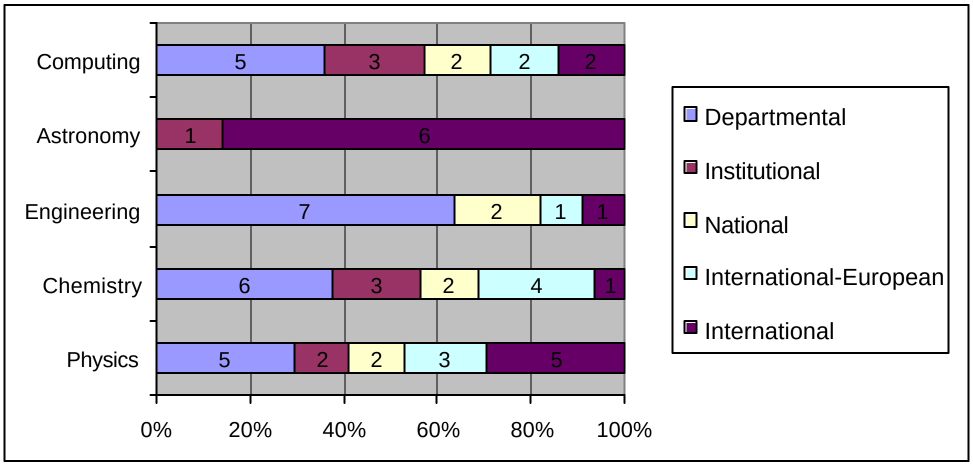


Figure 34: composition of the group

It was equally interesting to notice a marked difference in the choice of the most useful Grid model for these different groups. These findings are summarised in Table 10.

	Physics	*Chemistry*	*Computing*	*Engineering*	*Astrophysics*
Grid model considered most useful	Toolkit approach and small-scale Grids	Advanced portals and basic portals	Toolkit approach	Small Grids and toolkit approach	Advanced portals

Table 10: preferred Grid model

The preference for a particular Grid model was established using a weighing scheme similar to that used earlier, i.e.:

$$W = 1.5 \times n_{very\_useful} + n_{useful}$$

These choices are in good agreement with the data presented earlier:

- basic portals, which are the most user-friendly but only suitable for pre-compiled applications and with no provision for interactive access, were particularly popular with the chemists, known users of packages and stable production codes;
- small Grids, which are more suitable for a small-scale deployment within a department, are more suitable to the composition of typical engineering research groups. Physicists also expressed a clear interest for this technology, possibly as a 'free' complement to their HPC resources through cycle stealing and load sharing;
- advanced portals, which provide an ideal framework for groups with few stable production codes whilst still allowing basic re-compilations and on-line visualisation, etc. has been favoured by the

chemistry and astrophysics communities. The availability of the Cactus environment was probably a determining factor for this choice;

- toolkit approach is probably more appealing to users who like feeling in control (command-line oriented) and are also more suitable than portals for communities where the level of code re-use and code-sharing is quite low. This solution has been preferred by the physicists, engineers, and computer scientists. Note that the physicists (and to a lesser extent the engineers) are experienced HPC users and as such not afraid of using technologies based on command line and setting up batch style configuration files. The latter category, the computer scientists, possibly expressed this preference as this approach is the closest to some of research topics;
- Internet and peer-to-peer computing only achieves a certain level of popularity in the computer science and physics communities. The former once again because it is an area related to their research interests, whilst the latter also regards this technology as being able to provide more 'free' cycles.

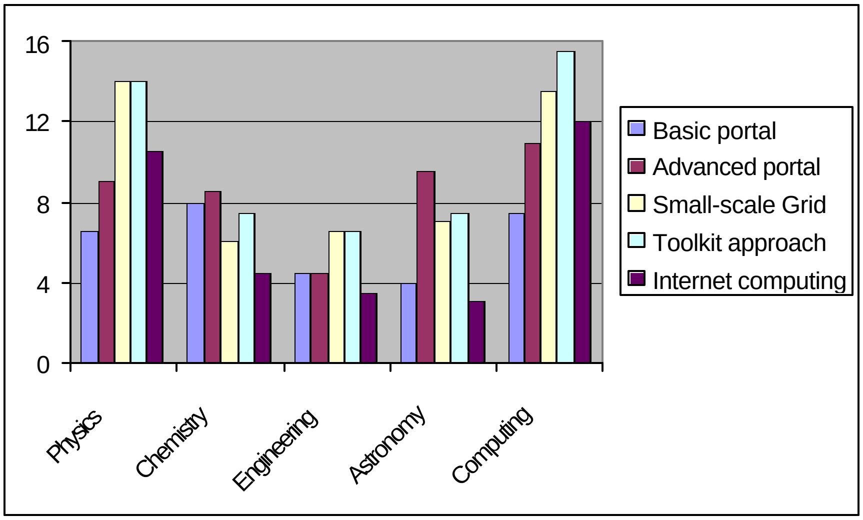


Figure 35: preferred Grid model

Note that these results may vary slightly depending on the weight associated to the 'very useful' category. A weight of 2 has also been tried to study the shift in preferences. This resulted in a tiebreak giving the preference to the toolkit approach, basic portal, and small-scale Grids for the physics, chemistry, and engineering communities respectively.

The final set of data from "Section 2: Awareness," page 125, that should be mentioned is the importance of various named factors in a Grid environment.

The methodology used to derive the classification from Table 11, page 151 was quite simple: the number of 'very important' and 'important' votes were counted and weighed as described earlier (weight=1.5), and then sorted in decreasing order. The data presented correspond to the Top 5 resources in a Grid environment. The numbers in brackets represent the weighed number normalised by the num ber of respondents for each group.

A number of conclusions can be drawn from these data. First, the need for faster CPUs was uniformly perceived as the most important factor in computational science and engineering. The second most important category was a greater amount of total memory and increased number of processors (PEs). This fact is confirmed by their appearance in the Top 3 of all computational science disciplines represented. The need for longer run-times seemed particularly acute for the physicists, chemists, and engineers, but oddly enough not so much for the astrophysicists. These latter actually differ once again from the other categories by selecting factors such as 'access to remote resources' and 'access to better resources for the jobs' (both factors are obviously linked) which have been left out by the other computational disciplines. The authors cannot explain this particular point.

Physics	*Chemistry*	*Computing*	*Engineering*	*Astrophysics*
Faster CPUs (1.32)	Faster CPUs (1.34)	More PEs (1.18)	Faster CPUs (1.14)	Greater total memory (1.29)
Greater total memory (1.00)	More PEs (1.03)	Faster CPUs (1.04)	More Pes (1.09)	Faster CPUs (1.21)
More PEs (1.00)	Greater total memory (1.00)	Ease of use (1.00)	Longer run-times (1.00)	More PEs (1.21)
Longer run-times (1.00)	Longer run-times (0.94)	Best machine for the job (0.96)	Greater total memory (0.95)	Best machine for the job (1.14)
Better throughput (0.88)	More memory per processor (0.91)	Test scalability on more processors (0.82)	Better turn-around time (0.73)	Resources not locally available (1.14)

Table 11: most important resources in a Grid environment

The overall conclusion of these findings is that although the physics, chemistry, and engineering communities (which represented 53% of the respondents) have roughly identical requirements, other user groups such as computer scientists and astrophysicists have additional considerations such as access to different architectures not available locally, user-friendliness, and testing the scalability of their algorithms.

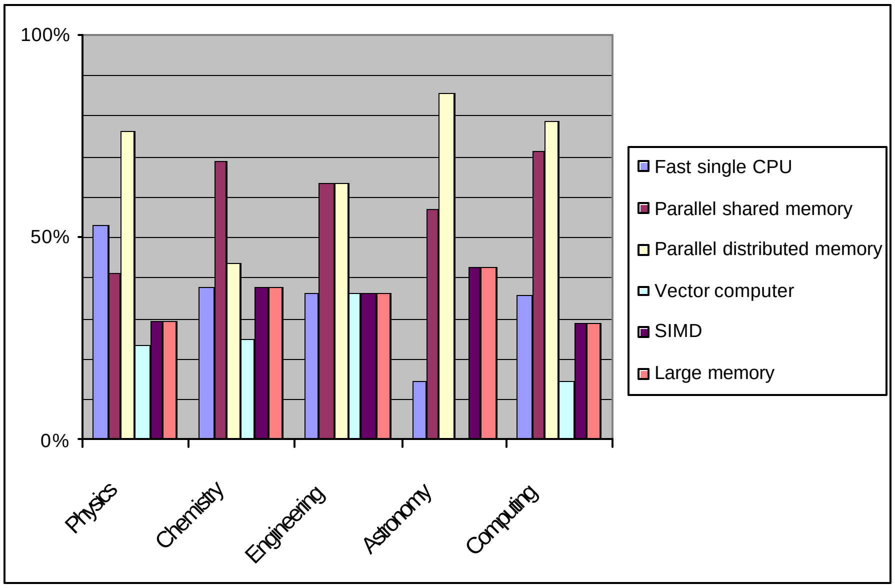


Figure 36: important architectures in a Grid environment

Interesting variations were also observed when analysing the type of system deemed to be important within a Grid (see "Section 2: Awareness," page 125). Although parallel systems were everyone's favourite, the preference for a particular type of architecture (shared vs. distributed memory) varied across the fields. The importance given to the former was not correlated with the reliance on OpenMP as expected (see 'Section 4: Application profile," page 156), but instead with the availability of critical software such as Accelerys' (formerly known as Molecular Simulations Inc.) which until recently were only available for Silicon Graphics systems. Another important point was the importance given to the availability of fast serial systems (or by extension, fast parallel systems configured to offer adequate support for serial processing).

As a member of the GGF Applications Working Group pointed out following a presentation of these findings, these results could be expected. Making a parallel with telephony, just as people would say that the main feature they expect from their provider is cheap calls (the same answer would have been made 10 years earlier!), the features that they actually value the most are those linked to technological advances (e.g., mobile telephony, fast access to the Internet, etc.).

Operating system		
Unix	54	64%
Windows 2000	18	22%
Windows NT	12	14%
Total	84	100%

Table 12: operating system used

An interesting factor with respect to the interoperability of Grid components, is the dominance of various operating systems in user groups. A summary of this data for all respondents is included in Table 12, whilst Figure 37 provides a breakdown across main research fields.

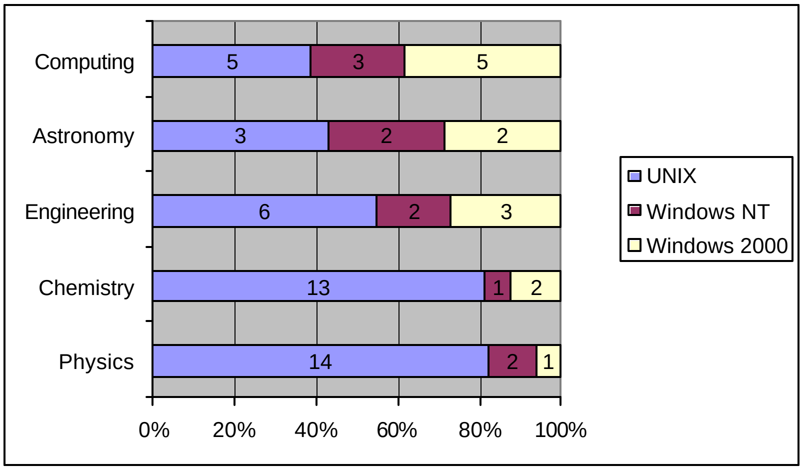


Figure 37: operating system used (by research area)

Whilst UNIX is still the dominant operating system, it is worth pointing out that the Microsoft Windows is quickly catching up. This has definite implications on the development of top-level Grid applications as a growing proportion of users will have little or no experience of UNIX-style operating systems and their reliance on command line driven events. This class of users will certainly expect to carry out all the required operations on the Grid through sophisticated graphical user interfaces. The new disciplines singled out earlier in this section (medical sciences, bioinformatics, data mining, etc.) also generally fall into this category.

Do the working habits developed by using any particular OS actually influence the adequacy of a Grid model over another? This issue was studied by considering separately the data for each OS (UNIX, and all flavours of Windows), and measuring the fraction (expressed in percentage) of respondents who have either described a model as 'useful' or 'very useful'. No weighing has been used to allow the normalisation of these data. These results are presented in Figure 38, page 153.

It was surprising that these data suggested that Grid computing is better received amongst the users with Windows as their regular desktop OS. This finding was confirmed by the higher level of support for all the Grid models presented, with a noticeable increase for small-scale Grids and advanced portals. An increase in the popularity of all types of portals was obviously expected but failed to materialise.

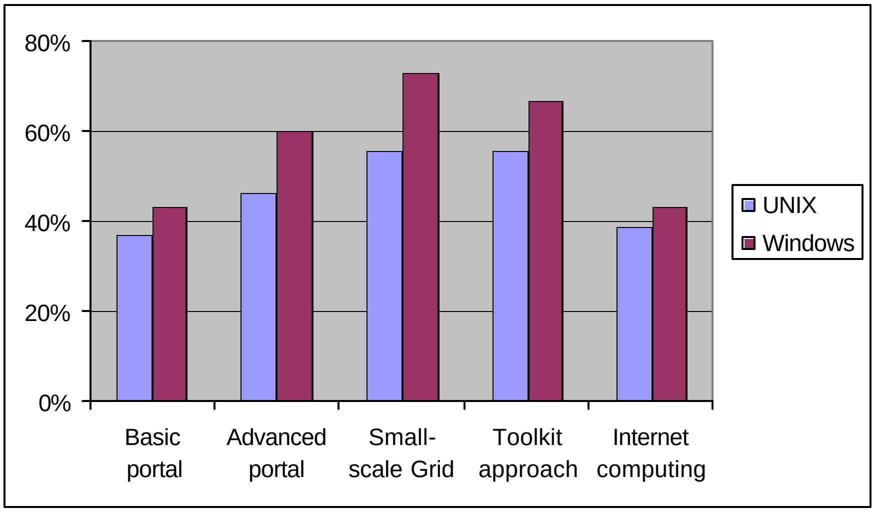


Figure 38: preferred Grid models

III.4.3 Composition and structure of research groups

Respondents were asked to estimate the amount of code development done in their groups. Three categories were defined:

- black box users: only pre-compiled applications are run. Input parameters are defined and results extracted, but the user does not modify and/or have access to the source code;
- mainstream users: this group of users may make occasional modifications to the source code or may recompile using, for example, a different set of modules. However no significant code development is done;
- user-developers: as the name implies, code development is an integral part of this activity. New runs often involve recompilation or modification of the existing code.

An indication of the percentage of each group that operates under each of these models is shown in Figure 39, page 154. Not unexpectedly, most groups contained a mixture of types of user. Figure 39, page 154 also shows that mainstream users and user-developers outnumber black-box users for most groups.

Of those groups involved in a single type of activity, 14 groups are 100% user-developers. All of these groups had heard about the grid; 8 had previous Grid experience in Globus (5 groups), LSF (3) and Condor (2). Perhaps not surprisingly, only 3 of these groups felt that the basic portal model would be useful or very useful. The toolkit approach was favoured by 12 of them with somewhat lesser interest in the other grid models. It is also worth noting that out of the 14 respondents who are 100% user-developers, 8 are computational scientists.

At the other end of the scale, only 2 groups (both chemists) run only black box type applications. Neither of these groups had heard of grid technology, but felt that benefits to them from a grid network might *include "More computational resources"* and *"Collaboration with other research groups"*.

A closer look at the break-ups by discipline provides some valuable insight about the level of experience within these different groups. Although this data was somewhat difficult to interpret quantitatively, they provided some clear indications of the user profiles within these groups as well as their likely working practices.

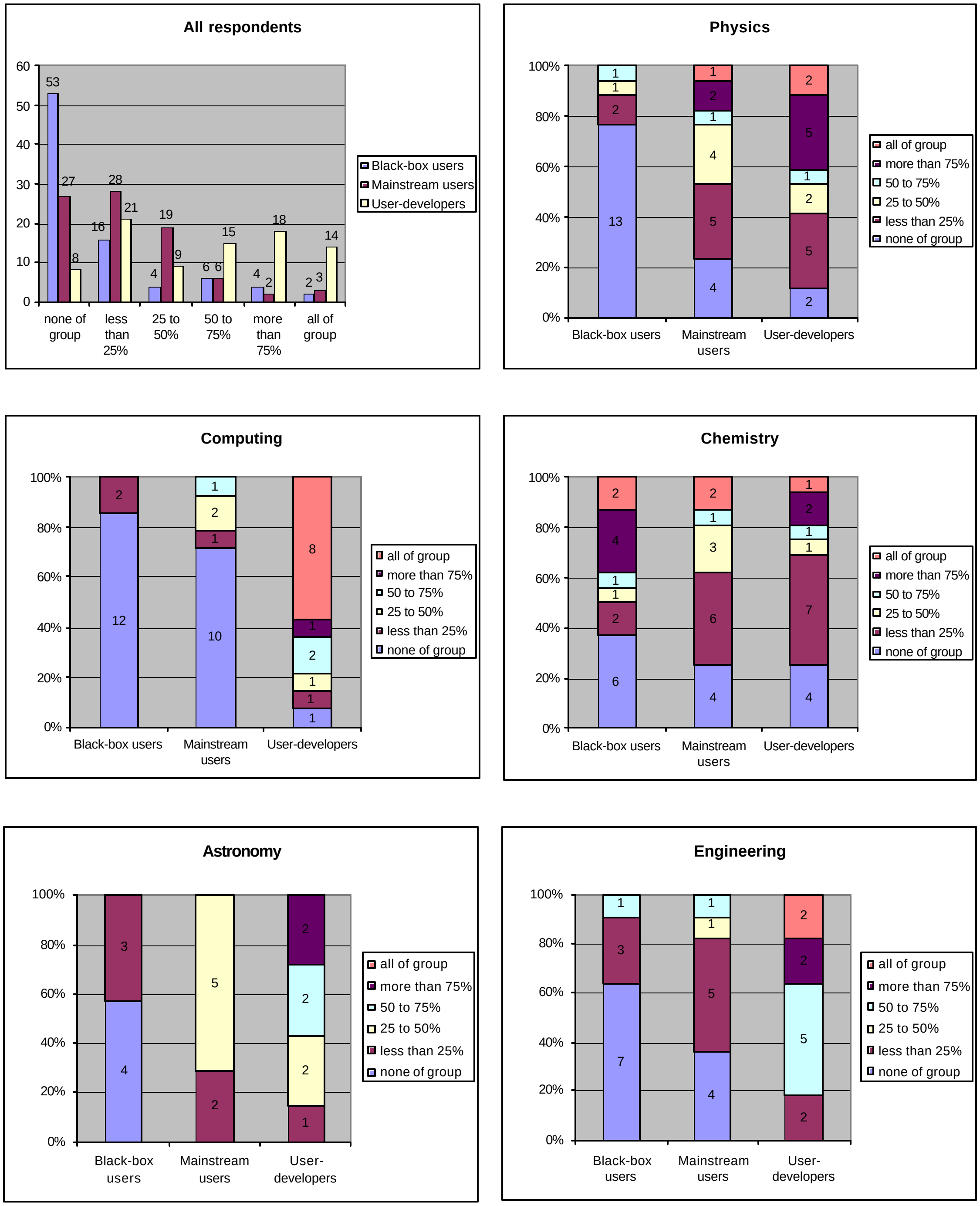


Figure 39: extent to which groups are involved in active code development

Not surprisingly, two groups appear at the extremes. First the computing community, which is composed almost exclusively of developers, and therefore does not include any black-box users. At the other end of the spectrum, the chemistry community includes a significant fraction of black box users and very few user developers. These results are once again consistent with previous findings and initial expectations.

The remaining three groups have a more balanced composition, although each of them exhibit characteristic features:

- astrophysics: this community has a lower number of user-developers than the other remaining two research areas. This is a sign that the development of production codes is undertaken by fewer 'specialists' within the group who are mostly focusing on this task;

- engineering with the exception of the computer scientists, this group is the one with the highest level of user-developers. This suggests a lower level of co-ordination and collaboration within these groups. Whilst some other groups have adopted a structure in which code development is the responsibility of a few experts and the rest of the group focus on the application, the data suggest that this community is composed of smaller sub-groups, all developing their own specific code independently.

- physics: this community has a surprisingly high level of user-developers. Their composition is close to that of engineering groups, but with a higher proportion of mainstream users.

There are a number of issues arising from this. The strong fraction of user-developers (with the exception of computer science where this is obviously a natural occurrence) indicates a low level of co-ordination and collaboration. Scientists are likely to benefit from having their production codes developed by experts through community efforts (such as that illustrated by the UK Collaborative Computational Projects, the CCPs [45]) which would maximise code re-use instead of having to re-inventing the wheel. In addition, it is even possible that some users regard the term 'black-box users' as pejorative and may be reluctant to admit falling within this category.

The overall composition is likely to significantly shift towards the black-box users end of the spectrum as emerging disciplines such as those outlined earlier (medical and biological sciences, data mining, etc.) are much more likely to be composed of a majority of black box and mainstream users. This shift will hopefully also become noticeable for the physics and engineering communities as funding bodies are now encouraging code re-use and community efforts.

III.4.4 Summary

Respondents to the survey were drawn from a wide range of disciplines, although three-quarters of the respondents work in 'mainstream' HPC fields of chemistry, engineering, physics, astronomy and computing. However, the emergence of new disciplines linked to the biological and medical sciences was also noticeable.

Most groups include several types of users ranging from black-box users running pre-compiled applications or packages through to users actively developing code. This is entirely consistent with the fact that most respondents reported an interest in a range of Grid models, as each model would be expected to be more suited to a particular type of usage, e.g. basic portal computing is limited to users running black box applications, whereas the survey showed that a significant number of user-developers already have some experience of various components of the Grid 'toolkit'.

A review of the data from the previous section ("Section 2: Awareness," page 125) showed a number of variations across the fields when considering factors such as the most important resources in a Grid environment, or again the preferred type of Grid model and its scope.

Finally it is worth noting that over a third of users use the Windows OS, and that this number may increase further with the emergence of the new disciplines from the biological sciences; it is therefore important that tools developed for Grid applications will run on both Windows and Unix platforms (interoperability and user-friendliness).

III.5 Section 4: Application profile

III.5.1 Introduction

The aim the 'Applications Profile' section was to elicit more detailed information about users current applications codes. Questions were asked relating to:

- languages used;
- current computational bottlenecks;
- typical input/output data size and format;
- typical job requirements;
- additional requirements, e.g. numerical libraries; and
- portability of code between platforms.

This document summarises the user responses to the questions in this section.

III.5.2 Languages and parallel environments

The languages used in current applications are shown in Figure 40. Approximately 55% of applications are written in Fortran (including HPF), while C and C++ account for 37%. Java is used in less than 8% of codes. The suitability of Java for scientific codes will be discussed later in section "Application portability," page 164.

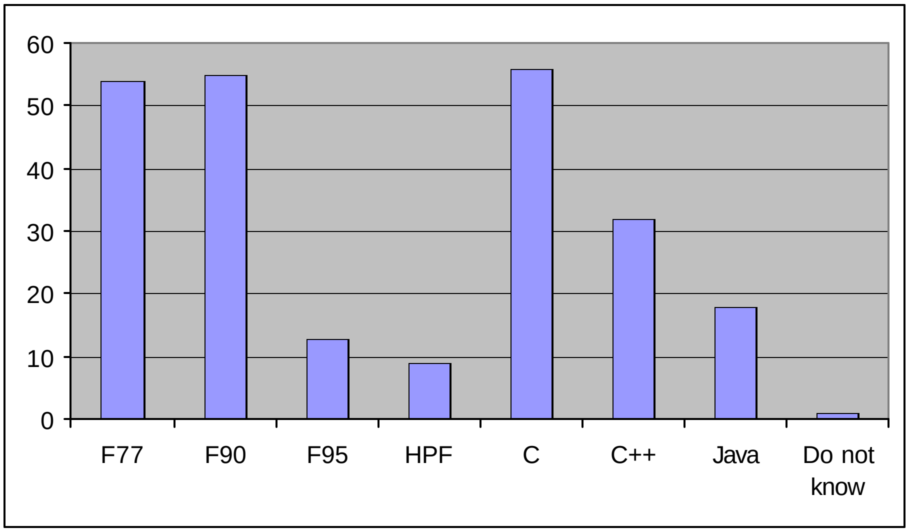


Figure 40: languages used in current applications

These figures reflect the overall usage across the range of scientific communities; however the range of languages is different within different groups of users, as shown in Figure 41, page 157.

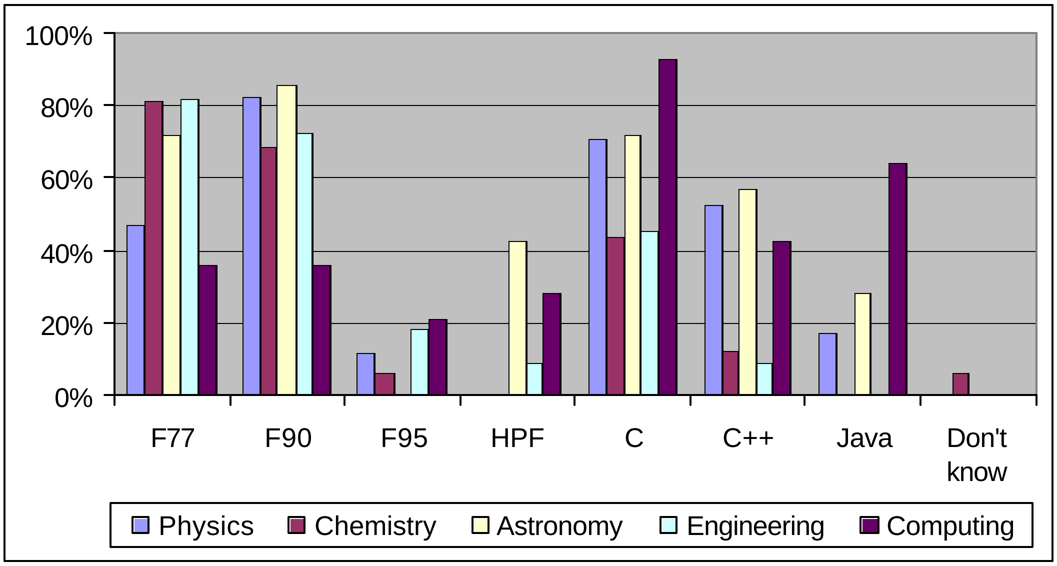


Figure 41: languages used in different communities

There are a number of points arising from this data:

- As might be expected, Fortran is still the most widely used language among the scientific communities.
- Overall, the number of F90/F95 application codes just about exceeds those written in F77. This long overdue phenomenon is particularly noticeable for physics codes, by opposition to, say chemistry and engineering where many legacy codes are still in use.
- The influence of this legacy for both the engineering and chemistry communities can also be observed when considering the popularity of newer, more sophisticated programming languages. Thus, C is noticeably less used than within other communities (under 50% of them are using at least one application written in C), usage of C++ (a more recent and 'abstract' language) goes down further to 10%, and no one in these communities seem to make use of Java applications.
- The computer science (including HPC) and astrophysics groups use a wider range of languages. Interestingly, C is used by over 90% of this first group of users.
- There is an increasing use of Java from the physics, astronomy and computing communities – in fact over 60% of the computing groups report using Java - but it is not used at all by groups in chemistry or engineering.
- HPF achieves a certain popularity in the computer science and astrophysics. Whilst the former principally use this language to undertake research (e.g., on compiler design or on autoparallelisation), the latter are still using HPF codes for production. The well-known Virgo consortium for instance, is still using an HPF/Craft implementation of their production code *hydra* [62].

The parallel programming environments used in applications are shown in Figure 42, page 158. Note that the categories 'Don't know' and 'HPF' have been double counted, i.e., taken into account both in Figure 41 and Figure 42, page 158, as they are relevant to both datasets.

Not surprisingly, over 50% of parallel codes are based on MPI, while SHMEM (15%), OpenMP (14%), HPF (11%), and PVM (8%) only accounts for about 48%. This split was broadly reflected among all the major groups of users, although one can observe particular characteristics, such as the large number of SHMEM codes in communities known to have used extensively large MPP systems in the nineties (principally Cray T3D and T3E). This is the case, for instance, of the physics and astrophysics communities. To a certain extent, the proportion of PVM (and to a lesser extent SHMEM) applications still in use witness the level of usage of legacy codes.

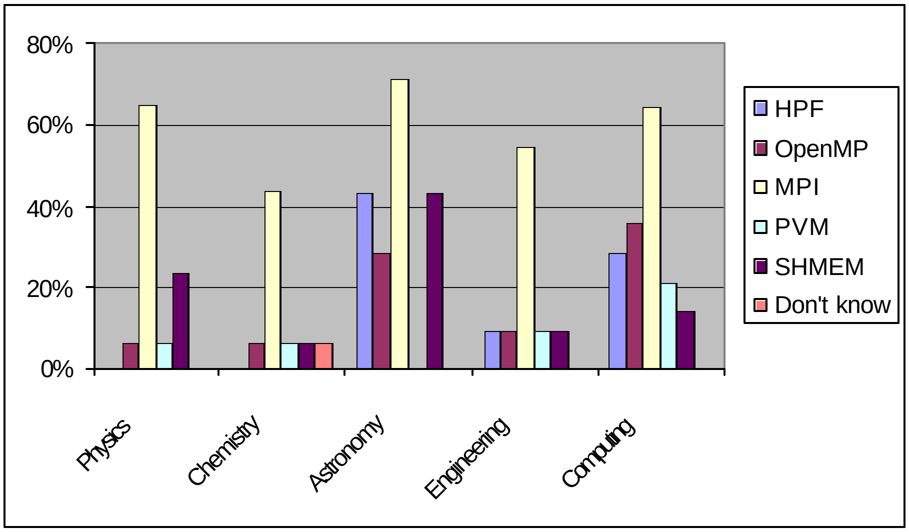


Figure 42: parallel programming environment used in applications

The evolution of these various languages and parallel environments in the next ten years will be discussed in further detail in "Section 7: Future needs," page 183.

III.5.3 Application bottlenecks

Understanding what the bottlenecks are in users' current applications is a key requirement for determining the adequacy of the various Grid models, as well as their composition, configuration, and usage policy. Users were asked to rate a number of possible bottlenecks according to the following criteria:

- Critical prevents the user from running their codes.
- Major work can be done but the quality is severely affected.
- Minor can work-around but would prefer some improvement.
- Not an Issue

The responses are summarised in Figure 43, page 159. The biggest single issue was constraints on CPU power; approximately 80% of users considered this to be either a critical or major factor affecting the quality of their work. Memory capacity and CPU to memory performance were also seen as significant bottlenecks in current applications. Note that as many users are not aware of the true limitation posed by the CPU-to-memory bandwidth, many of them place the blame for slow computations on CPU power instead.

Not surprisingly, these bottlenecks were found to be consistent with the importance of the various factors (CPU power, network performance, etc.) presented in Figure 29, page 141.

Another way to consider these data is to split them in three main categories. By order of importance, these would be:

1. CPU power;
2. memory characteristics (amount and performance); and
3. network and disk performance and capacity.

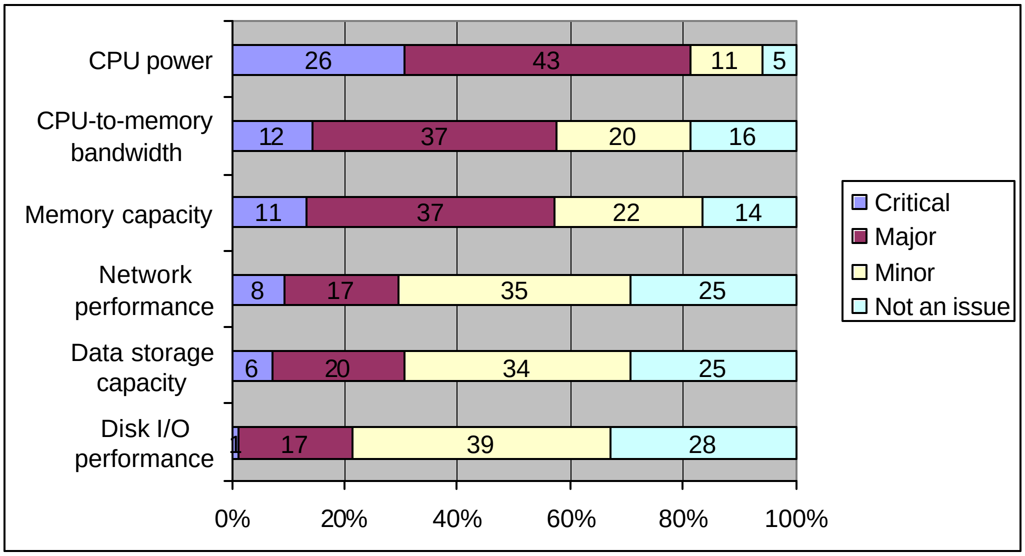


Figure 43: bottlenecks in current applications

This categorisation is very interesting because it allows easier prediction of the way application bottlenecks are expected to be shifted in a Grid context. For instance, network performance is expected to be one of the main bottlenecks as it is not only crucial for metacomputing applications, but also for many data Grid activities, and more generally, any steering / interactive operations (including code development) for which low latencies are essential.

Furthermore, although storage capacity, disk I/O and network performance are considered to be at most a minor issue for the time being, this question relates to issues with users' present applications and not how users perceive the future. As one respondent rightly pointed out:

> *"You naturally scale to problems so that they fit to the computer you have access to => you can always use something larger/faster..."*

Novel working practices, such as applications relying on access to large distributed databases, will place further strain on network and disk performance. On the contrary, a typical bottleneck such as 'Memory capacity' has been considerably improved as the typical amount of memory per PE has gone up from 128Mb to 1Gb per node within the last five years. It is therefore expected that this will be seen only as a secondary bottleneck in the near future.

Field of work	*Physics*	*Chemistry*	*Computing*	*Engineering*	*Astrophysics*
CPU power	**0.94**	**1.22**	**0.82**	**0.95**	0.86
CPU-to-memory bandwidth	0.62	0.63	0.64	0.68	0.64
Memory capacity	0.56	0.78	0.57	0.73	**0.93**
Network performance	0.29	0.13	0.54	0.50	0.57
Data storage capacity	0.24	0.53	0.29	0.27	0.50
Disk I/O performance	0.00	0.22	0.29	0.09	0.29

Table 13: major bottlenecks for each application area

These findings were relatively consistent across research groups, although a number of application-specific features could be identified. The data presented in Table 13 were obtained by filtering the data for each of the five major research areas, applying the weighing scheme described earlier, and normalising the result by the number of respondents for each discipline. The main points are:

- the need for increased 'CPU power' is consistent across all disciplines, although it seems to be particularly acute for the chemistry community;

- the answers from the astrophysics community differ once again from those of the other groups, as they regard 'memory capacity' as their key bottleneck;
- the performance of the network infrastructure is noticeably more noticeable for the astrophysics, computing, and physics communities that for any other; and
- the same comment applies for the astrophysics, computing, and chemistry communities regarding the importance of disk I/O performance.

No direct correlation could be observed in these data between any given bottlenecks and the preferred choice for a particular type of Grid model, so model-specific statistics are not presented here. This suggests that the user community does not regard the various Grid models present ed in this document as providing a solution to a particular type of bottleneck. This fact is particularly surprising considering the specificity of these models (e.g., Internet and peer-to-peer computing are probably unsuitable models for applications withhigh memory requirements).

Other bottlenecks, such as 'poor turnaround time', were also mentioned by some respondents.

III.5.4 Input and output data requirements

The range of input and output data requirements is shown in the figures below. Most I/O data is in plain text/ASCII or binary format (see Figure 44).

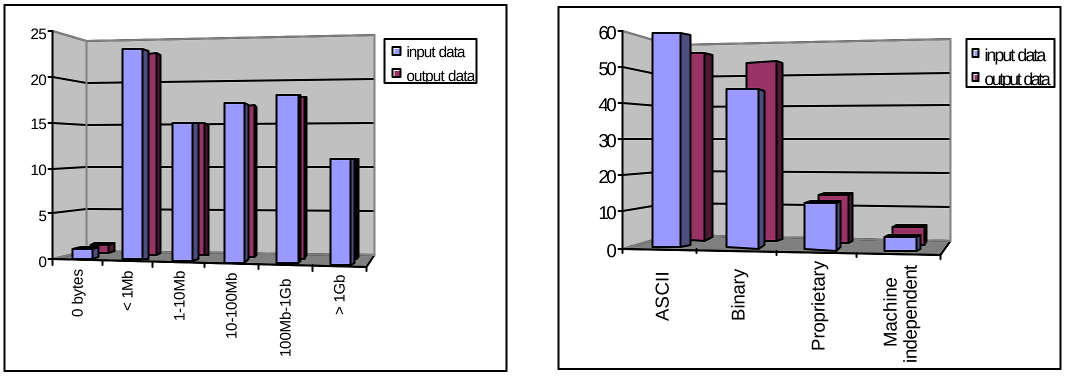


Figure 44: input/output data requirements (left) and input/output data formats (right)

Although the distribution of input and output data are the same, they were not correlated for any individual user – large input data does not imply large output data and visa versa. This is illustrated in Figure 45, which shows the absolute difference between input and output data requirements.

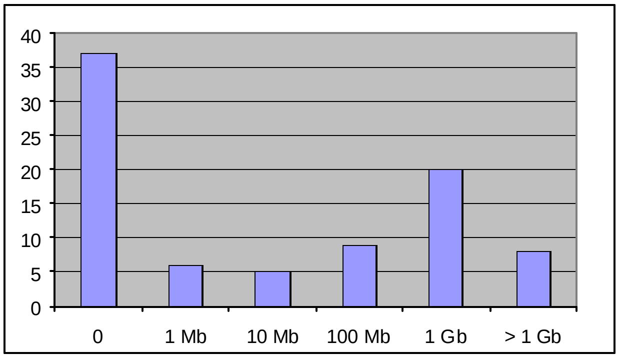


Figure 45: absolute difference in size between input and output data

One of the questions raised by a respondent in this section perfectly illustrates the type (or level) of information Grid operators and those in charge of dissemination and information activities on Grid computing will need to address:

> *"Will there be Grid support for Matlab?"*

III.5.5 Typical job requirements

Details were provided from most users giving their typical job requirements in terms of:

- memory used by their application;
- number of processors and memory per processor for distributed applications;
- run time; and
- temporary disk storage required.

As these questions were quite detailed, only a subset of the respondents actually provided a detailed answer. The breakdown per topic is as follows:

Answer provided?	*Yes*	*No*	*Don't know*	*No answer*
Memory used by application	63	4	8	10
Typical run-time	72	3	5	5
Temporary disk storage	41	25	12	7

Table 14: feedback level for main topics

The typical memory requirements presented in Figure 46 were obtained by multiplying the amount of memory requested by the number of PEs (for distributed memory applications, such as MPI-based applications), and by directly using the users' specifications for shared-memory applications.

These results are quite surprising considering the importance given to the availability of large memory elsewhere in the survey. For instance, the importance of greater total memory in a Grid environment was judged as 'very important' or 'important' by 37% and 37% of the respondents (see Figure 29, page 141). In addition to this, this factor was considered to be the most important factor in a Grid environment for the astrophysics (and second most important for the physics) community. Finally, the lack of memory was reported to be a critical or major bottleneck for 13% and 44% of the respondents respectively. Looking at the data however, the actual requirements appear relatively modest as 84% of the respondents reported using less than 10Gb of memory!

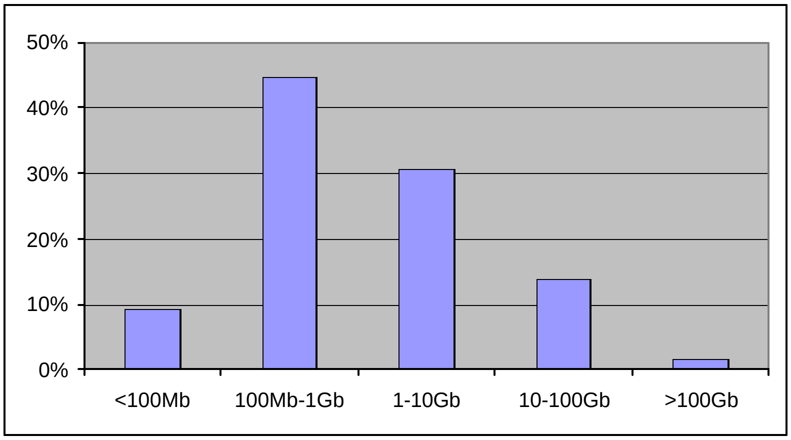


Figure 46: memory requirements

It was possible to gain a better understanding of this discrepancy by filtering this subset of respondents. In fact, the subset is composed of a mix of users with actual large memory requirements, but also of users whose main resources are systems with low memory (see "Section 5: Infrastructure," page 170). Note that this problem can become particularly acute for computing resources not operated via a batch system, which seems to be the case of most departmental servers.

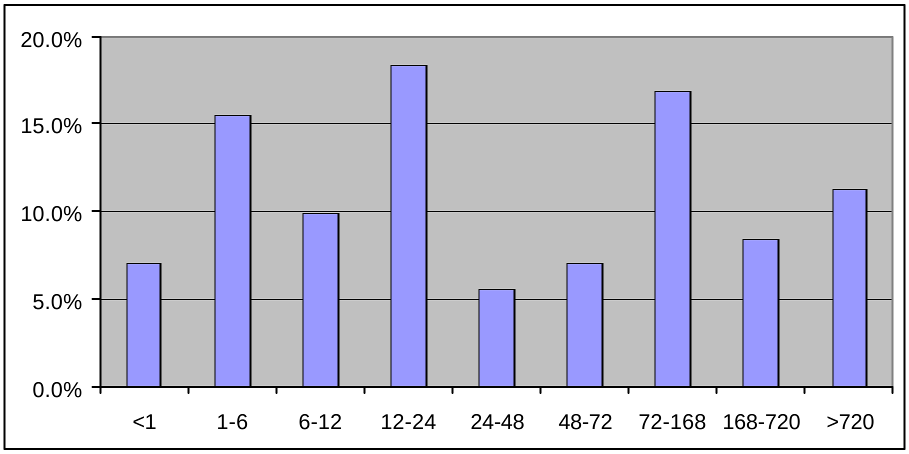


Figure 47: job run-time distribution (clock time, in hours)

One of the complaints reported in the "Section 2: Awareness," page 125 was the lack of provision for long jobs in HPC centres. The data from Figure 47 suggest that many users typically run their application for much longer than any HPC centre would let them do. These results suggest a clear split between users who have access to remote general-purpose HPC sites (most of them being amongst the 32% of respondents who run their application for 72 hours or less), and the users who rely on dedicated sites (e.g. own systems or local resources). This is the case for the 37% of respondents who typically run their application for 3 days or over.

Long run-time is often associated with serial jobs, which would introduce a considerable bias in these statistics. It is therefore important to consider the difference in the job size distribution between applications requiring short and long run-times. These limits have been set to 12 hours and less (typical maximum for HPC centres over weekdays), and over 72 hours (typical maximum run-time for HPC centres).

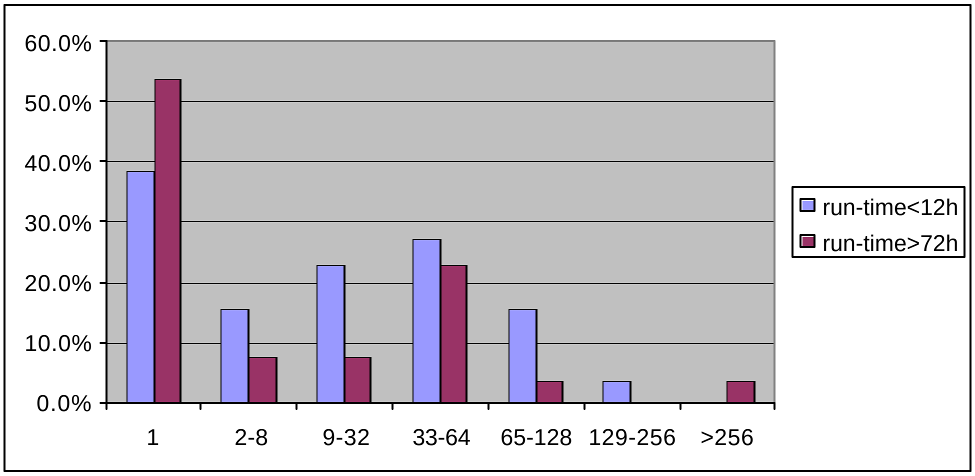


Figure 48: job size distribution for shorter and longer jobs

The data from Figure 48, page 162 confirm a bias from serial jobs, although this latter is much smaller than expected. Leaving aside these serial jobs, we can see that 31% of the jobs running over 3 days are indeed large parallel jobs running on 64PEs or more. These jobs are typically Grand Challenge applications which suffer from low time limit restrictions. Having to checkpoint and re-start a job was a solution which was criticised by most of the respondents concerned. It is indeed important to bear in mind that the need for 'longer run times' was rated as the fourth most important factor (with 69% of the respondents describing this as 'Important' or 'Very important'), and almost considered as important as the availability of greater total memory. Clearly, this category of users will not considering migrating to a Grid infrastructure unless they can get the guarantee that their application will not suffer from shorter time limits.

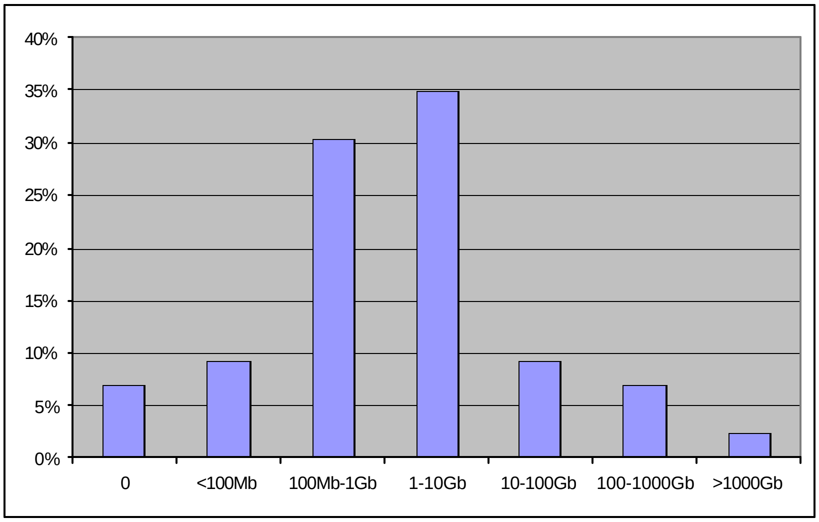


Figure 49: temporary disk requirements

The next important - yet often overlooked – factor is the need for temporary storage. This includes both storage space required for the data generated during the course of a simulation and any input files required beforehand. The vast majority of users (80%) do not require excessively large amounts of temporary space. The remaining respondents - those requiring 100Gb or over – may find that many sites will not be able to guarantee such a large amount of disk space on the fly. It is therefore important to include this requirement as part of any Grid's job submission file to avoid starting applications on system s which could not meet their requirements during the course of the simulation.

III.5.6 Stability of application code

Figure 50, page 164 shows that users typically run their applications less than 10 times before recompilation. About half of the recompilations (41) are needed to look at new problems; just under a third (24) are done when combining a different set of modules and 6% of users (5) need to recompile before every run. Conversely, almost 20% of users (15) stated that they never need to recompile their application.

These data suggest that accessing Grids through basic portals may not currently be the ideal solution for a majority of users. On the other hand, a number of groups are using packages or stable production codes (recompilation only required every 50 runs or more), which will therefore be good candidates for a basic portal. This fact is only partially reflected by the data though, as the basic and advanced portals were only judged 'useful' or 'very useful' by 45% and 60% for this category of users - compared with 33% and 43% for the other respondents (see " Section 2: Awareness," page 125).

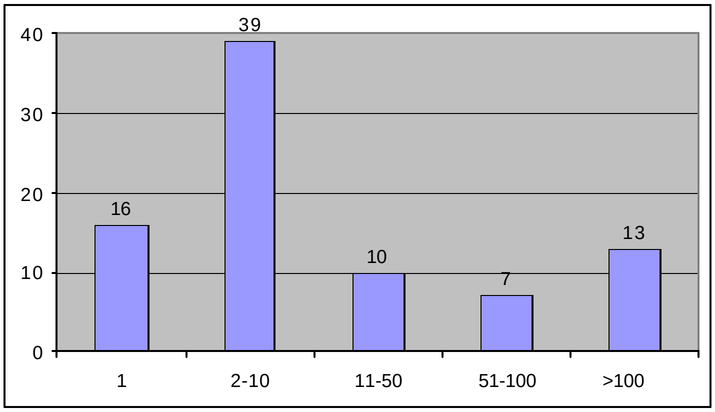


Figure 50: number of runs before recompilations

70% of the respondents stated that they used applications that they did not write themselves, although in 85% of these cases the code had been developed within their group. More than half of the respondents (49) require access to special resources. These include (the numbers in brackets give the number of respondents):

- MPI (24);
- numerical libraries, e.g., NAG, BLAS, fftw, Lapack, etc. (18);
- F77/F90/F95/HPF (10);
- OpenMP (8);
- graphical libraries (2);
- PVM (1);
- Cactus (1); and
- SHMEM (1).

Some other dependencies are platform specific, such as SHMEM or Fortran compilers with Cray pointer extensions, and basic UNIX utilities such as Python, Perl, lex, Yack, etc.

Based on the results in Figure 41, page 157 and Figure 42, page 158, many more of the respondents will require 'standard' programming languages (such as C/Fortan/Java) and parallel libraries (MPI). Their availability was probably assumed!

Some package users have provided further details. The most frequently mentioned codes are Gaussian (11), Gromacs (5), Gamess (4), Amber (3), Castep (3), Crystal (3), Dmol (2), ADF (2), Jaguar (2), and Dalton (2). It is worth pointing out that most of these packages are chemistry packages, and that a package like Gaussian is used by a majority of users within this community (69% of the chemists polled, 11 out 16 respondents in this community). Other packages getting a mention are Cpmd, Molcas, Molecule, Molpro, Grid, Mopac, Dirac, Dot, Lautrec, Sybyl, Fluent, Vasp, nsmb, avbp, Numeca, Arpege, Opa, Unity, SAS, NWChem, FHI, Aces2, Lodyc, Oasis, Insight2. Some of these packages are available as source code, whilst some others are pre-compiled applications. Many other applications not listed here (such as Matlab) are also used routinely as part of the computational cycle. Their omission by the respondents suggests that post-processing is expected to be a process still taking place on their local system rather than directly on the Grid.

III.5.7 Application portability

The portability of application codes will be a pre-requisite to take real advantage of the Grid. Thus, codes which rely on the availability of platform-specific or even site-specific features will only benefit from a few of the advantages offered by Grid computing. The concept of portability is here taken in its wider context to encompass access to resources (e.g., through a portal), reliance on libraries and dependencies.

	Yes	*No*	*Don't know*	*No reply*
Ability to run application through a Web interface	40	22	23	
Other resources required to make the application run and/or compile	49	22	8	6
Consider possible port of code to Java	13	64		8

Table 15: portability issues

The perceived adequacy of Java for scientific codes was then briefly discussed by the respondents. The first step was to assess if users would consider running their applications through a web interface (typically, a portal). The responses to this question were very mixed. Nearly half of the respondents felt that it would be possible (40), a quarter (22) said it was not, and the remaining quarter (23) did not know. Additional comments were received from 25 respondents. The main features arising from these comments were

- too much data, jobs too long, or communications problems (6):

> *"Long batch jobs (days) are unsuitable for a Web interface."*

> *"Input options required are difficult to standardise, hence control input should come from a plain-text file."*

> *"Problems that require interaction via graphical interfaces mostly have high data volume to be transported/analysed… There are no solutions for this via web interfaces so far."*

In fact, many of these (negative) issues arise from a misunderstanding of the purpose of these portals (i.e. providing a uniform access point to distributed heterogeneous resources) or are not a real problem for the portal approach. For instance, the issue of input (parameters) files is worth considering. The real benefits of using a smart Java-based portal (as opposed to a simple CGI Web service) is to perform sanity checking of the input files. Many input files include dependencies between their parameters and using such systems could warn a user whenever one of the parameter values causes other simulation parameters to go out of bounds for the desired simulation. In addition to this, using a plug-in architecture would ensure a fast and easy development of customised input file generators as part of the portal. It was reported following the ENACTS presentation at the Applications Working Group (GGF3) that these capabilities were very popular amongst the user community who had been given the opportunity to benefit from them. Most of the issues related to Grids portals are being investigated by the GCE working group [63].

- Not needed (6):

> *"We are happy with the current environment."*

> *"This is usually not allowed on most HPC centre. In fact, it is also not necessary."*

> *"I can perceive of no advantage. The codes are 'research codes'."*

> *"Compiling through the web sounds ridiculously complicated compared to just opening a few xterms."*

- Nothing available / not thought about it (6):

> *"It will be highly desirable but at present no one of the application I use have a web interface available."*

- Working on it (3):

> *"Not yet available but we will implemented this."*

- Too much development work needed (2).

It cannot be denied that integration of application codes in a portal environment will require a certain amount of work, technical expertise, and administrative privileges which cannot be met by most user groups. Thus, initiatives should be encouraged to incite resource providers (especially HPC centres) to offer their services to carry out this integration and support it whenever required. The most logical approach is to start by supporting the most popular packages (e.g. Gaussian), 'and stable' application codes used by the large groups. Such initiatives are already underway, e.g. EPCC is currently involved in the development of a portal for the Magneto-Hydrodynamics (MHD) consortium in the UK. Both national research councils and the EU should consider providing funding for this type of activities in the same way as it did in the early days of parallel computing (e.g., through the HPCI [39], Parallel Applications Programme (PAP) or Europort initiatives).

However, a significant amount of work still remains to be done to demonstrate the benefits of such an approach to the user community. In the same way that many regarded Email or the World-wide Web as a 'funny concept' which would not last, the user community will need to be convinced of the benefits of Grid computing and be reassured both about its long-term future and how the overheads incurred by this transition will be made acceptable with the assistance of service providers.

A good strategy would be to *demonstrate* the benefits of Grid computing at user conferences (chemistry, physics etc.) by setting up an application demonstrator which could be accessed from the conference. Obviously, this demonstrator must be based on mainstream user application codes or packages such as Gaussian. End users will not embrace Grid technology until the case for supporting it has been clearly established.

The most urgent and important action is to finalise the standardisation/recommendation process undertaken by the GCE Working Group to allow support centres to develop future-proof and re-usable solutions for their users (componentisation). Guidelines must also be drawn quickly to encourage the design of interfaces with a common look and feel. This factor was one of the main wishes expressed by the respondents (see "Perception and benefits of computational Grids," page 130). It is also worth mentioning that at more than one respondent was confused between the Internet / peer-to-peer computing models and access to the Grid through portals:

> *"If I understand it correctly, to efficiently run an application through a web interface, the application needs to consists of several tasks which do not need to communicate with each other too often. In our application a lot of internal communication is needed between the sub-processes."*

A lot of effort	*Some effort*	*No effort*
17	45	23

Table 16: effort required to improve code portability

The results from Table 16 are rather encouraging as they suggest that most application codes could be made truly portable with minimum effort. Furthermore, nearly three quarters of the users said that they would be prepared to spend the time to make their applications more portable. Many users felt that the changes needed were relatively minor and only two users commented that they would need to make major changes to the code. User comments can be classified as follows (numbers in brackets):

- minor changes/easy to change (26):

> *"The code is easily portable to any Unix with Fortran and MPI."*

> *"Not much since we try to make the code portable, and have tested it on several computer systems."*

> *"Very little. Mainly things like changing the scripts used to run the compiled programs (they are mainly stand-alone commercial packages that need parallelisation programs like MPI of OpenMP)."*

> *"Code has been run on a wider variety of machines. There were some changes for Cray-computers due to incompatibilities in Cray-MPI. Those are basically all the changes we have to make to the code in order to run on different environments."*

- major changes (2):

> *"In some cases I would need to fully rewrite the interface and replace for a command line or WEB based interface."*

> *"The codes are tuned for a specific hardware. If you change the platform the code will run but probably not very efficiently. How can I implement this in an 'easy portable way'?"*

- don't know (7);
- need MPI/OpenMP/other parallelisation (6).

Overall, the situation regarding code portability was better than expected and should improve as new codes progressively replace old legacy codes. These encouraging results can be explained by a combination of factors:

- The new generation of computational scientists are more likely to have been taught programming through a series of formal courses. Thus, good software engineering practices are increasingly common amongst Physics and Engineering graduates. It is worth pointing out that Java is often taught as a first programming language. For example, the MSc in High-Performance Computing run by EPCC [11] includes Java as part of its core modules.
- The standardisation process has made possible the creation of standard libraries (such as MPI) which guarantee portability of parallel programs. The definition of Fortran 90 has been equally important to prevent the appearance of additional non-standard features in Fortran 77 codes.
- As compiler technology has evolved, the gain obtained by using low-level programming languages (such as assembler) and platform-specific calls is now hardly worth compromising portability.
- HPC centres have played an instrumental role in making legacy codes portable (e.g., by porting SHMEM codes to MPI) and developing new (portable) production codes.
- Legacy codes are progressively being phased out and replaced by new portable codes.

Portability often implies Java as it is the safest way to ensure true portability (if one avoids Microsoft extensions). Numerical libraries such as Visual Numerics IMSL JNL [64] are now developed using this language whilst Java bindings are being developed for many others. So, would users consider developing their new applications using Java, or even port their existing production codes? Unfortunately, the vast majority of respondents (64) do not consider this to be a viable option. The major problems foreseen by respondents are summarised below:

- speed/performance too slow (26):

> *"We require high speed and my impression is that Java is not necessarily very efficient for large scale numerical simulations."*

> *"I need high performance and I do not expect to obtain so from an interpreted language. I will not rewrite my code unless forced to do so. Much on the contrary I expect other people to give me reasons (and good ones!) for rewriting code"*

> *"Bloody stupid idea."*

> *"MUCH TOO SLOW! There are many people in our community who even believe C is too slow, and Fortran is the only acceptable language. We are not that strict, and allow C and Fortran - but nothing else. C++ is only permitted for applications which are not time-critical, like evaluations.*
>
> *Java IS conceivable as a tool to LAUNCH an application, but I guess standard scripts (shell, perl, tcl,...) will do a better job there."*

> *"Performance issues. I did not tried it by myself, but all what I have heard about JAVA is that it is comparable very slow programming language. May be this will change with time."*

- too much/large amount of effort involved (24):

> *"The code we are working on is a results of 15 years of research. It is just too expensive to re-write it."*

> *"The existing code comprises several 100,000 program lines."*

> *"Too much effort; we have to concentrate on the physical problem."*

> *"It seems that too much time will be wasted. My aim is to get scientific results as fast and effectively as possible, not to invest in major programming efforts"*

- dangerous floating-point arithmetic (1).

These comments raise two important issues. Firstly, it is true that porting large Fortran codes to Java (or any other language for that matter) is not cost effective. However, it has been demonstrated that syntactical similarities allow quick conversions of large C simulation codes to Java. As an example, the port of a cut down version of the Grand Challenge simulation code Ludwig [65] (which comprised over 15,000 lines of code) to Java was successfully undertaken by a student at EPCC in only two weeks [66].

On the other hand, new codes for which portability is essential are suitable candidates to be written in Java. However, effective dissemination of the work undertaken by the Java Grande Forum (JGF) [67] is essential. The JGF advocates that Java has the potential to be a better environment for Grande application development than any previous languages such as Fortran and C++. Their goal is to develop community consensus and recommendations for either changes to Java or establishment of standards (frameworks) for Grande libraries and services. The work of the JGF Benchmarking Group [68] has for instance established that on certain platforms (e.g. Sparc, Intel) Java can now deliver performance between 50% and 90% of that obtained using C or Fortran. The lack of awareness of these results in the user community once again raises questions about the effectiveness of dissemination activities. Just as for Grid computing, the only way to ensure an effective dissemination within the user community is to report these benchmarks (some based on mainstream academic research codes) at *user conferences* rather than at computing conferences from which users are typically absent.

III.5.8 Summary

Fortran is still by far the most popular language for computational codes, although object oriented languages such as C++ or Java are gaining popularity in the computing, astrophysics, and physics communities. The importance of legacy codes in the chemistry and engineering communities is outlined by the predominance of Fortran77 codes over their more modern Fortran90 counterparts. The vast majority of parallel applications rely on MPI, although SHMEM and HPF production codes are still in use, in particular in the astrophysics community.

Most applications are CPU-bound and cause 'CPU power' to be singled out as the main bottleneck. Although network performance, and to a lesser extent disk I/O performance, is not regarded as a significant bottleneck, the emergence of new techniques such as computational steering, job migration, and access to large distributed datasets is likely to re-position it as a major bottleneck in the near future.

Users' requirements in term of storage space and global memory were significantly lower than expected. Indeed, with memory capacity being reported in previous sections as being one of the most important features in a Grid environment, the actual requirements specified by the respondents fail to explain their worry. More interesting was the users' requirements for maximum run-times. The distribution of these requirements clearly allowed the separation of users of national facilities from those using their own resources, although it was shown that some of the groups requiring the longest run-times were running large parallel applications. The predominance of serial and small parallel jobs is also a factor worth considering seriously in a Grid context!

The typical 'stability' of application codes (i.e., number of runs before requiring a recompilation) proved lower than expected, a sufficient fact in itself to explain the disaffection for basic portal reported in the previous section. Apart from these many codes in development there were also a number of stable production codes (e.g., Gaussian) and packages.

Portability does not seem to be an issue as the vast majority of application codes are either already fully portable, or would only require minor modifications. This is a good omen considering that it will be a pre-requisite for generic Grid computing. However, when questioned on the likelihood of porting their codes or developing new applications in Java, most of the respondents dismissed this possibility. This was mainly on due to the perceived poor performance of this language and the large overheads incurred by the porting of large codes.

III.6 Section 5: Infrastructure

III.6.1 Resources available locally

It was apparent that users had access to a wide variety of systems, both in type (custom-built, Beowolf, MPP, shared memory system, etc.) and size (from single processor desktop system to hundreds of PEs). Besides desktop systems, the most common architectures were shared memory systems (24, mostly SGI Origins), Beowulf clusters (18, mainly based on AMD Athlon processors), and distributed memory systems (combination of Cray T3E and IBM SP). Note that these numbers are purely indicative as some of the details provided by the respondents were incomplete.

A full listing of these systems is given in Appendix D, "Part 5 – Infrastructure".

III.6.2 Resources available within national borders

A total of 62 respondents answered this question. Out of these, 54 provided details of remote systems available to them within their national borders. One respondent explicitly stated that they used to have access to their national facilities but that support for this latter had been discontinued:

> *"At the moment we do not have access to external computer resources. Previously we had access to an IBM SP installed at UNI-C."*

This situation is likely to have arisen for many of the respondents located in a country with no national computing centre. This effect was particularly noticeable for Denmark with only a single respondent reporting access to remote resources. To a lesser extent Greek users also seemed to experience difficulties securing access to remote resources.

The systems themselves are mostly large SGI Origins (33), Cray T3Es (24), and IBM SPs (16). A handful of HPs, Hitachis, Fujitsus, custom-made (GRAPE), and large Beowulf clusters were also mentioned. A full listing of these resources is given in Appendix D, "Part 5 – Infrastructure".

III.6.3 Resources available worldwide

Only 18 respondents reported using computing resources outside of their national borders. Access to these resources is typically secured through collaboration with a foreign group who has access to its national facilities or participation in a visitor programme such as those funded under the EC's IHP's Access to Research Infrastructure programme [69]. Typically, these resources were not primary resources but complete or top-up local ones. These systems were mostly large systems found in national facilities and European Large-scale Facilities (LSFs).

III.6.4 Network infrastructure

The quality of the connectivity to the outside world is often described as the main bottleneck for Grid computing. However, following the deployment of multi-Gbps links across Europe (for instance as part of the GÉANT project [3]), the bottlenecks for end-to-end bandwidth are now being displaced by the link between the organisations' gateway and the end-users' desktop.

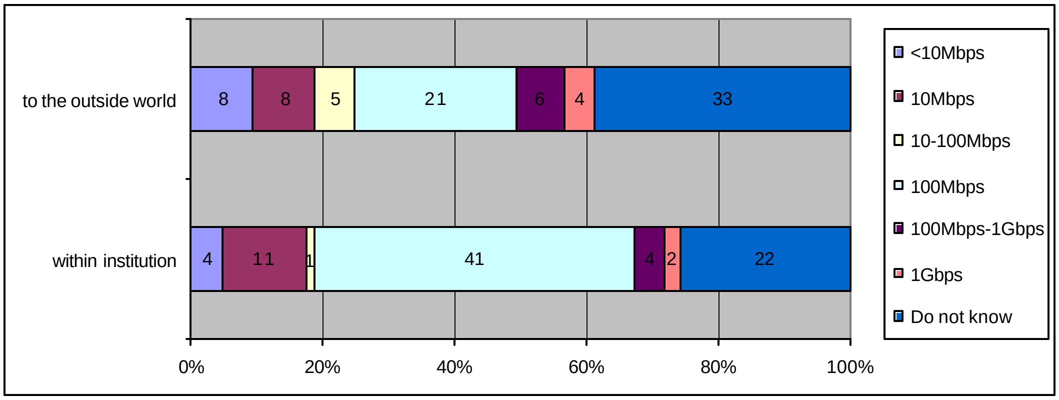


Figure 51: network performance (bandwidth)

Recent years have seen the creation of a number of very successful national and multi-national advanced high-speed research networks. The most successful of these at the national level, following the success of the earlier very high-speed Backbone Network Service (vBNS), are probably the Internet2 Abilene network and the Canadian CANET-3 network. Multinationally, Europe created the pan-European network TEN-155 [112], which is currently being superseded by the multi-gigabit GÉANT [3] network. The period between 23 October 2001, when the first national research and education network connected to a GÉANT access port to send and receive its international traffic, until 30 November 2001 when the TEN-155 network was due to be closed down, saw a complex migration process in which other National Research and Education Network (NRENs) progressively moved their traffic from one network to the other.

From 1 December 2001, GÉANT is in full production service. A four-year project set up by a Consortium of 27 European national research and education networks (NRENs), with DANTE as its co-ordinating partner, GÉANT was launched at the IST 2000 event in Nice in November 2000. Co-funded by the European Commission as part of its Framework V Programme, its goal was the improvement of the previous generation of pan-European research network, TEN-155, by creating a backbone at Gigabit speeds.

GÉANT supplies the European research community with a number of services, including testbed and guaranteed QoS features. Reaching over 3,000 research and education institutions in over 30 countries, GÉANT provides the highest capacity and offers the greatest geographic coverage of any network of its kind in the world.

Networking infrastructure is crucial for Grid applications. However, the bandwidth demands vary from application to application and between usage scenarios. As an example we can give the GridLab project [5], where the Cactus simulations are used on the application side.

While it is beneficial to have high performance links between the individual sites in the GridLab testbed, the whole system needs to be open (i.e. not built using private dedicated links) in order to allow easy incorporation of new sites, changes in topology etc. Also, user access to the Grid infrastructure should not use any dedicated private links, as this severely limits the number of potential users (admitting only those with such special arrangements), as well as user mobility. Thus, the software developed by GridLab should be able to run on any Grid platform, even without specific network provision.

Various data movement schemes will be exploited by the GridLab scenarios. They vary in their demands on latency and throughput, as shown in Table 17, page 172. In fact, the application toolkit (GAT) developed as part of the GridLab project will provide support for adaptive behaviour, in order to dynamically react to changes in the underlying network performance. Also, research and development in the area of mobile computing will emphasise the adaptability to the different network characteristics. This means that instead of relying on guaranteed network characteristics (like a particular throughput or latency), the GridLab solution is to provide a software infrastructure that is able to adapt to changing network conditions.

Scheme	*Min throughput (MB/s)*	*Max latency (msec)*	*Max jitter (%)*
Remote checkpointing	50	X000	100
Raw data movement	10	X000	100
Remote visualisation	0.5	300	10
Remote steering	0.1	50	10

Table 17: Minimum network requirements

Out of the many different scenarios considered within the GridLab project, the following two will stress the network the most:

1. ***Worm-type applications:*** such applications migrate from site to site within a Grid environment. For example, migration might become necessary when the resources (e.g. available compute time) are exceeded at the computation site that is currently used at a certain moment of the application run. For this purpose, a worm-type application performs application-level checkpointing that transfers the entire status (execution state and data) to another computation site. In this event, large data volumes (up to hundreds of Gigabytes) are transferred from one site to another. However, as the computation does not continue during the migration itself, the actual link quality between the sending and receiving nodes is not critical, as long as the data arrive in reasonable time.

   Meanwhile, means for efficient transfer of large amounts of data across networks are currently being studied and developed within other projects (e.g. GriPhyN, DataGrid, etc.) and will be available to the wider scientific community. Also, such huge data movement requirements will not be necessary before the second half of the project (where more "production-like" runs will be possible).

2. ***Remote visualisation:*** remotely visualising data sets or running simulations can create data streams each of which typically requires 5 MByte/sec or more. Algorithms that are able to adapt to different network bandwidths will be developed to reduce the actual requirements. Also, visualisation may only be required during the parts of the project time, leaving opportunity for co-operation with network operators offering managed bandwidth services that may be asked for the visualisations only.

   With remote visualisation, low latency is less important than small amounts of jitter (the latency variation), which should be kept as low as possible. However, current IP networks are not well-suited to guarantee such a behaviour; this is a topic of ongoing network research world-wide. Accompanying visualisation, remote steering and remote data exploration make greater demands on latency and jitter, while transferring only small amounts of data. Software adaptive to varying network quality will offer great benefits to such scenarios.

As the rapporteur from the workshop on *'Expanding the Grid reach in Europe'* organised under the auspices of the European Commission stated himself [70]:

> *"The links between infrastructure research, particularly networking, and Grid research must be strengthened. Research is required to capture the full infrastructure requirements of the Grid applications domain and to ensure networking advances are fashioned to provide the infrastructure the Grid domain requires. The GÉANT project means that for the first time in Europe international circuit capacity is no longer a bottleneck.*
>
> *However, increasing core network bandwidth is not sufficient to meet the needs of e-Science on the desktop. Research institutes will have to invest over the next few years in faster connections to the desktop of they are to take full advantage of the opportunities afforded by e-Science."*

However, increasing the raw performance of the network infrastructure will not be sufficient to implement Grid services relying on network QoS. Fortunately, the networking research community has been actively involved in the development of mechanisms capable of delivering network QoS. The need for QoS is well-documented [71], and dedicated Working Groups have been set up within the framework of the Internet Engineering Task Force (IETF) [72] to look specifically at this issue. The most common approaches are probably the Differentiated Services (diffserv) [73] and Integrated Services (intserv) [74] models. Research in this field is exemplified by Intersim-DiffServ [75], a collaborative project of EPCC and Cisco Systems Inc, aiming to produce a tool built around the OPNET network simulator to model the traffic and performance of differentiated services networks.

The discussion of these models would extend beyond the scope of the current study, so interested readers should refer to the references provided for further information on these topics. Suffice to say that great progress has been achieved in these fields, resulting in strengthening the concept of Virtual Private Networks (VPN) [76,77]. VPNs are defined as follows [76]:

> *"Although some vendors and service providers might disagree, in common usage a virtual private network is a group of two or more computer systems, typically connected to a private network (a network built and maintained by an organisation solely for its own use) with limited public-network access, that communicates 'securely' over a public network. VPNs may exist between an individual machine and a private network (client-to-server) or a remote LAN and a private network (server-to-server). Security features differ from product to product, but most security experts agree that VPNs include encryption, strong authentication of remote users or hosts, and mechanisms for hiding or masking information about the private network topology from potential attackers on the public network."*

The relevance of this research to Grid computing cannot escape anyone.

III.6.5 Summary

The infrastructure demands and possibilities have been discussed in this chapter. As the networking infrastructure in Europe is consistently improving, no bottlenecks are foreseen for Grid applications.

What is seriously lacking is a true, persistent, production-quality global research network –one that eventually will be capable of terabits per second data rates. These will be needed as part of major multinational scientific collaborations in areas such as high-energy physics, radio and optical astronomy, weather forecasting and climatology, biological sciences and earth sciences.

The collaboration between the European Commission and NSF continues to develop; as a result we foresee that the lack of a global research network will soon be remedied, at least for transatlantic connectivity. The DataTag project [113], for example, plans to develop just such a connectivity.

III.7 Section 6: Security and services

III.7.1 Introduction

The ‘Security and Services’ section of the questionnaire had two purposes. The first was to gain some understanding of the level and type of service that respondents would expect from a grid service provider. Secondly, questions were asked relating to both users current levels of security and those anticipated if they were to join a Grid environment.

This document summarises the user responses to the questions in this section.

III.7.2 Quality of service

To try and determine the quality of service that Grid users might expect, respondents were asked to rate a range of added services. The results are summarised in Figure 52 where the factors are sorted by order of importance, using a simple (but arbitrary) weighing scheme:

$$W = 1.5 \times n_{very\_impor\tan t} + n_{impor\tan t} + 0.5 \times n_{useful}$$

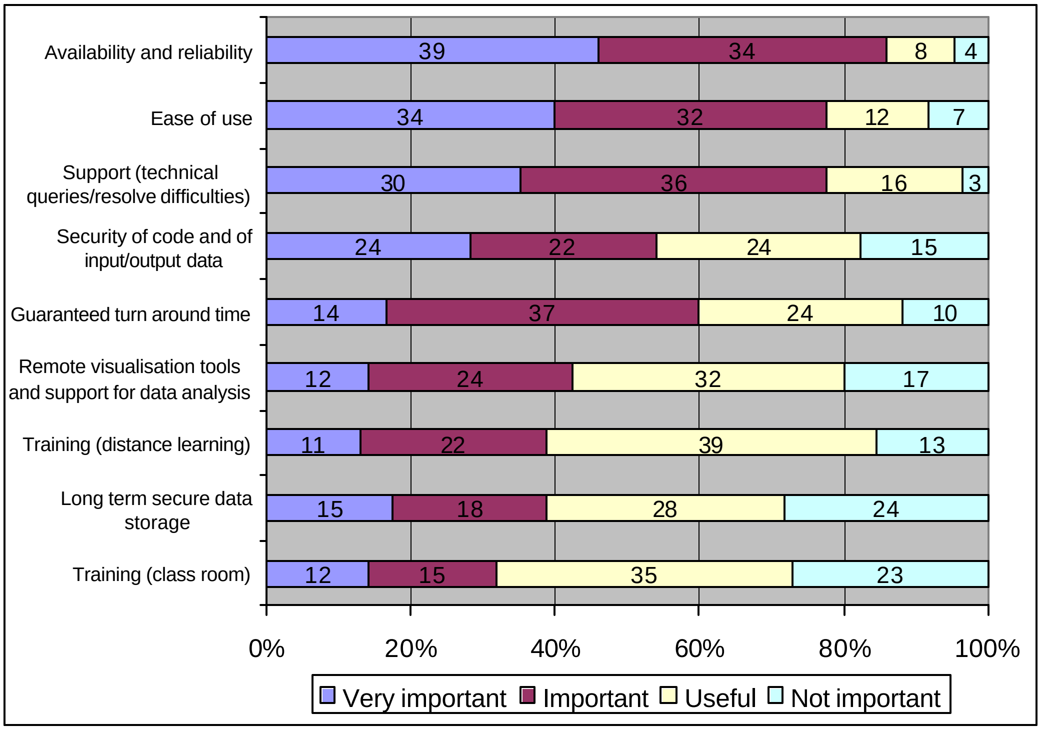


Figure 52: relative importance of added services from grid service providers

The importance of these services can be grouped into five categories with, as might be expected, availability, reliability and ease of use coming top.

1. Availability and reliability

These were by far the most important requirements. This highlights the need to avoid presenting early Grids aimed at users as 'experimental', or 'at-risk' services. As was emphasised in the previous sections, researchers want to focus on what they do best and are known not to be keen on investing time and resources to experiment with new technologies. It is therefore important to ensure that the first open Grids are sufficiently mature and offer a sufficiently stable and predictable environment (i.e., similar to that currently offered by HPC centres) before attempting to migrate grass root users to this type of resources. Ignoring this warning may have long lasting consequences resulting in discrediting the idea of Grid computing itself within the user community for a few years.

2. Ease of use and support

The importance of these factors has already been highlighted several times throughout this survey. However, ease of use can mean different things depending on the level of experience and working practices of the users concerned (see section "Section 2: Awareness," page 125, "Section 3: User profile," page 146, and "Section 4: Application profile," page 156 for a fuller discussion of this topic). The consensus is that Grids must *not* be any more complicated to use than current HPC systems. The case for the generalisation of Grid portals could not be established satisfactorily earlier, but one fact which arose is that users will be reluctant to learn yet more new syntax and mechanisms for submitting their jobs. The standardisation efforts undertaken by the Working Groups from the Scheduling and Resource Management area must therefore be finalised – and implemented – before any Grid infrastructure becomes really suitable for end-users. Hopefully, this issue has been flagged by the relevant GGF Working Groups who are making good progress. Ease of use also implies the availability of easy to operate tools to manage and query the status of user jobs and data and handle data migration seamlessly.

The issue of delivering adequate support to Grid users returns to the consideration discussed in "Section 2: Awareness," page 125: how should the support environment be set up in a distributed framework? Furthermore, support will be required not only to users of Grid technology but to those who attempt to deploy a Grid within their institution. For the latter, one could advocate the creation of dedicated national and regional centres following the model set in the UK [78]. As far as users are concerned though the situation is not so clear-cut. Should the users be taken care of by the site where the job has run (which may not be known, e.g. in the case of general technical queries asked independently of a particular run), or should a particular user/group be assigned to a dedicated centre according to their geographical location, institution, research area, or supporting research council? The possibilities are numerous. Drawing on the considerable experience gained through supporting users of HPC centres, one suggestion is that a suitable mechanism must include features such as ease of use (i.e., one generic E-mail address), guaranteed response time (SLA) and continuity of service. This latter point is particularly important as being aware of and understanding the problems previously experienced by a user will often suggest the solution to their future problems. Should support be offered by more than one centre, great care will need to be taken to ensure a high-level of co-ordination between participating centres, in particular with respect to specifying standard procedures and the sharing of information. In a national context, one of the best approaches may be that adopted in the UK with several Regional centres co-ordinated by a National centre [2].

Another aspect related to user support is liability: somebody has to assume responsibility for a potential malfunctioning of this distributed infrastructure. Regardless of its structure, users will want a single point of contact!

Finally, it should be pointed out that this service is viewed as much more important than both distance learning and formal training. This suggests that users do not expect to have to learn yet more new computing matters to use the infrastructure but instead need a structure capable of solving specific problems, especially when confronted with new technologies.

3. Security and code and data and guaranteed turn-around time

It is worth pointing out that the security of the code and data was considered to be progressively more important as respondents progressed through the questionnaire. In "Key features of computational Grids," page 140, only 43% reported this factor as being 'important' or 'very important' in a Grid environment compared to 52% in the present section. The concept of 'security' actually encompasses two different things:

- protection of codes and data from malicious activities (typically hacking) and protection of sensitive data against from thieves; and
- protection against accidental corruption of these files whilst on the Grid (e.g. during automatic file transfers, job migr ation, etc.).

This latter point emphasises the need for fault tolerant/error recovery mechanisms which will preserve the integrity of data on the Grid. Such topics are partially covered by the Grid FTP Working Group and the Data Replication Research Group [79].

Everyone agrees that data migration in a Grid environment should be made seamless. However, there is a natural concern from users that migrating their data without their knowledge means that they may be a risk of data being lost, corrupted or even accessed by some unauthorised person. Providing users with basic monitoring tools which keep them informed of the whereabouts of their files is a pre-requisite to enable them to preserve a certain level of control.

Note that the issue of security is particularly acute for industrial users (listed as one of their top 3 requirements) and new user communities from the biological area, especially when their research has medical or pharmaceutical applications. For all these categories of users, data security, int egrity, and confidentiality is a must!

The turn around time was rated highly as would be expected from the previous findings. The concept of guarantee is intrinsically associated to QoS and SLAs. However, this service will require potentially lengthy polit ical agreements for non-local Grids as the issue of liabilities, compensations and penalties will need to be introduced (see discussion on this topic in "Section 2: Awareness," page 125). This aspect is particularly important for industrial users - who will expect contractual obligations which will have to cover QoS – and users of time-critical applications. The work undertaken as part of the Gridlab project will actually cover the topic of guaranteed turn around time, although within a more focussed context. The final aspect linked to QoS is of course Grid economics for which very little progress has been made with the exception of the work of Rajkumar Buyya [80].

4. Remote visualisation and data analysis, distance learning and long term data storage

Remote visualisation and computational steering are concepts which have been around for almost a decade but have failed so far to make an impact in the user community despite their undeniable benefits. With the noticeable improvements brought to the network infrastructure (see "Section 5: Infrastructure," page 170) and the advent of integrated solutions such as those provided by the Cactus Visualisation Tools [81] or currently developed as part of the UK RealityGrid project [9], remote visualisation and computational steering are expected to see their popularity increase considerably over the next 3 years. The moderate support for this service can also be explained by its limited relevance to part of the user community such as computer scientists.

The preference of distance learning over formal training is logical in a distributed environment. Distance learning can encompass both self-paced formal training (e.g. MPI on-line [82]), and technical documentation such as FAQs, introductory documents and tutorials etc. The wider scope of distance learning over formal training, its ability to deliver the information required to solve common (specific) problems, and its potential to address a much wider audience means that it will become an indispensable service within any Grid environment. However, operators of non-local Grids will need to adopt a concerted action to produce relevant material and avoid unnecessary duplication of work. In a European context, the lack of availability of this material in a given language may become an issue. The overall lukewarm support for this service again sends the signal that the user community would rather avoid having to spend any time learning about this new technology in the first instance. Computers (and by extension Grids) are simply regarded as tools by the user community, and any time spent learning how to use them is regarded as overhead.

The final service of this fourth category is "Long-term secure data storage". Although only moderately rated, this service is considered as essential by prospective industrial users. It should also be pointed out that the main academic research groups (especially in climate modelling) have generated large amounts of data that they have accumulated over time! The administration of such huge archives will need to be taken care of by Grid infrastructures. Bearing this in mind together with the fact that many of the respondents expressed their intention to get involved in collaborations which may involve sharing data (see "Section 2: Awareness," page 125), the case for establishing topical centres and topical Grids for such disciplines is strengthened further.

5. Formal training

The relative lack of interest for conventional training (provided in classrooms) is further evidence that users would rather avoid having to learn more technologies peripheral to their research interest. Distance learning and support are preferred, presumably for their ability to address specific problems whenever required.

It was not surprising to see that different groups of users have somewhat different perceptions of added-value services that they considered important. Table 18 shows the top-ranked added service for different user communities. The number in brackets indicates the result of the weighing scheme described earlier normalised by the number of respondent for each category.

Physics	*Chemistry*	*Computing*	*Engineering*	*Astrophysics*
Availability and reliability (1.18)	Ease of use (1.06)	Ease of use (1.18)	Availability and reliability (1.09)	Remote visualisation (1.29)
Support (1.06)	Support (1.03)	Availability and reliability (1.07)	Support (1.09)	Availability and reliability (1.21)
Ease of use (1.00)	Availability and reliability (1.03)	Support (1.00)	Security of code and data (0.91)	Ease of use (1.07)
Guaranteed turn-around time (0.94)	Guaranteed turn-around time (0.81)	Security of code and data (0.89)	Ease of use (0.86)	Support (1.07)
Security of code and data (0.79)	Security of code and data (0.75)	Remote visualisation (0.79)	Guaranteed turn-around time (0.77)	Distance learning (0.79)

Table 18: top 5 services for various user communities

Whilst the top 3 categories ("Availability and reliability", "Ease of use", and "Support") appear amongst the four more important services for all disciplines, their relative importance and the importance of group specific services both vary noticeably. Thus, whilst ease of use is seen as essential by the chemistry and computer science communities, availability and reliability are the major factors for the physics and engineering communities. This is not surprising considering that chemistry user groups are more likely to be composed of black box users or mainstream users, whereas physicists and engineers may be more ready to compromise on user-friendliness if this allows them to increase their usage of the resources. Although computational scientists rank ease of use as a key attribute, this should probably be interpreted as meaning ease of implementation of a Grid service rather than from a Grid users' point of view. These findings corroborate those found in "Section 3: User profile," page 146, and "Section 4: Application profile," page 156. On the other hand it was unexpected to find that "Remote visualisation" was the most valued service for the astrophysics community (with 1.29, the most highly rated of all services across). This may be due to exposure to solutions such as those provided by the Cactus environment, leading to a better understanding of the benefits offered by remote visualisation and computational steering.

Other services such as "Guaranteed turn-around time", "Security of code and data", and (to a lesser extent) "Distance learning" were also popular across all disciplines.

A summary of these results in terms of experience of using Grid Computing, favourite Grid model, and key added service is given in Table 19, page 178.

Other comments regarding user services included their cost:

> *"The administration, accounting etc must not be too onerous"*

… and the need for *"private databases"* and a more detailed description of what user-friendliness entailed:

> *"The support should be a high level support and not a secretary.*
>
> *The complete system (which supercomputers are connected, which shares are available, how long are the queues,...) should be transparent for the end user"*

Users field of work	*Physics*	*Chemistry*	*Computing*	*Engineering*	*Astrophysics*
Previous practical grid experience.	12%	0%	71%	0%	29%
Grid model considered most useful.	Toolkit and small Grids	Advanced and Basic portals	Toolkit approach	Small Grids and Toolkit approach	Advanced portals
Highest ranked added service from grid service provider.	Availability & reliability	Ease of use	Ease of use	Availability & reliability, Support	Remote visualisation

Table 19: previous Grid experience, preferred grid model and key added services for some key user groups

The Global Grid Forum 'Grid User Services' Working Group [109] has proposed a number of recommendations for the implementation of Grid services in a document entitled *'Grid User Services Best Practices'* [110].

III.7.3 Security issues

Trustworthy and reliable security is essential for many of the components discussed above. While a useful Grid environment demands convenient and even automatic access to available resources, the need for security makes this difficult. There are required appropriate mechanisms and tools which provide dynamic Grid applications with the ability to access the resources they require when they require them, and which also allow users and their collaborators to monitor and steer their applications wherever they are on the Grid, while at the same time maintaining security to prevent unauthorised access to running applications or resources.

Ideally, users should be able to log on (or authenticate) just once and then have access to all their resources in the Grid, without the need for further user intervention. It is necessary to ensure that a program can run on that user's behalf in such a way that the program itself can access the resources for which the user is authorised. Optionally, the program should also be able to delegate further, to another program. The user must be able to control the level of delegation.

Complicating this picture, each site or resource provider may employ any of a variety of local security solutions, including Kerberos, Unix security, etc. The Grid security solution must be able to operate in conjunction with these various local solutions. It cannot require wholesale replacement of local security solutions, but rather must allow mapping into the local environment. These security requirements impact almost all the Grid elements, including user authentication, resource management, monitoring, information services and portals.

Systems are installed behind a firewall for 61% of respondents, while 7% of the respondents did not know whether they had this protection. Institutions are getting increasingly aware of their vulnerability to hacker attacks and have been taking steps to protect themselves, so this number is expected to rise further over the next couple of years. On the other hand, firewalls are known to affect the end-to-end bandwidth of Grid applications running outside their local domain. To put it simply, the high level of security causes a decrease in the overall performance of the system. Thus, the user should be provided with the tools that will allow her to control the level of grid security. Firewalls are just one option of dealing with security issues, and they are still a problem for the Grid middleware. Of course, we are not able to list here all the varieties of firewall and all the grid systems installed behind them; instead, let us take as an example the Network Address Translation firewall and Globus. The major issue for those setting up Globus services behind a Network Address Translation (NAT) firewall is the selection of the name to use in the host or gatekeeper certificate. Clients expect to see the same name in the certificate that they get from doing a reverse lookup on the IP address they are connecting to.

If they are going through a NAT firewall the IP address they will be contacting will probably be that of the firewall instead of the actual host. Accordingly they will expect to see the firewall's name in the certificate.

Putting the firewall's name in the certificate works for external users, but users inside the firewall will then get mutual authentication errors, as they will expect to see the actual hostname in the certificate. There is no good solution to this problem at present.

Being able to provide high-performance solutions without compromising the security of the end-users' local domain will pose serious technical challenges to Grid middleware developers and providers. This issue has been raised on numerous occasions during GGF3.

Access permission control		
Yes	43	51%
No (I want security issues like proxy Credentials to be transparent to me)	8	10%
Don't know	33	39%

Table 20: control over the level of permission user grant to other Grid components (schedulers, resource brokers, etc.)

Somewhat surprisingly, only 51% of the respondents stated that they would like to have control over the level of permissions granted to other Grid components (see Table 20). However, only 10% felt that they would like security issues to be transparent to them, while 39% did not know.

Several respondents expressed reservations about granting access to remote users, principally on the basis of security, risk of interfering with the work undertaken locally, and the risk of introducing administrative overheads (systems administration). Amongst those respondents who were open to the idea, a significant number also commented on the need to be able to reserve resources for local use and close down external access whenever required. This requirement implies that Grid applications (or rather the underlying Grid middleware) must be capable of migrating (or re-starting) on a different system whenever they get suspended. This is an interesting topic which touches the on the areas of resource discovery, data migration, etc. The framework for addressing this issue has been put partially in place as part of the GGF groups (e.g. as part of the "Grid Information Services" area [83]).

The following points summarise the views of the respondents (numbers in brackets indicates the number of related comments):

1. The need to preserve access control of external users and discontinue it whenever required (12);

> *"We would like to be able to reserve local resources for exclusive use. That should be dynamically changeable."*

> *"There can be bottlenecks in the department's accesses to the outside world. We are fed through University college's link to Janet and there can be periods of high use currently. It might be useful to restrict grid access during such times."*

> *"Again not able to give a straightforward answer - some of applications need transparent security others I would like to control access. In particular bandwidth reservation for some applications"*

> *"Our system people would, most probably, not be willing to allow access from a remote system without them carefully monitoring and knowing what is going on, in any respect. We had quite some problems with unauthorised*

> *breaking into our systems. Furthermore, my feeling of our machines is in essence that of private property, so I must be given the right to turn off external access, if, for instance, the load becomes too heavy, etc. But maybe I misunderstood the question?"*

> *"There is a need for controlling what we give to other users. In other words: we would like to be able to use our resources for ourselves in the moment we need it."*

> *"We sometimes need to run experiments without any external disturbances."*

The key issue here is that user groups will only consider making some of their local resources available to the Grid if the middleware allows them to exert dynamic control over the amount of resources they are making available for Grid use. Access to such resources should be discontinued in period of heavy use by local users, or whenever a code needs to be run without interference from external applications. This type of feature has already been implemented in small-Grid middleware such as Condor [37]. The final two comments related to the importance of monitoring usage of their resources for security purposes (to prevent or circumvent malicious use), and the need for a system implementing QoS, i.e bandwidth reservation.

2. Serious security concerns (8);

> *"I am suspectfull to any application from outside my group that access my computers. There is a lot of valuable data on them and It is my responsibility to look after that and protect it from potential damages"*

> *"Security and grid computing is very hard to combine. If you increase security you decrease accessibility and have another source of unexpected bugs. My opinion is to have as much security as necessary but do not annoy the user with too many security checks."*

> *"There is no secure binaries, daemons, brokers, whatever..."*

The issue of security on the Grid is very complex and most users (and system administrators) are naturally quite wary about increased risks to their local domain. Judging from the responses in Table 20, page 179 and the comments made in this section, group leaders will require concrete evidence that the level of security provided by Grid middleware is adequate before committing any of their resources to a non-local Grid. This may therefore be regarded as *the* major obstacle to the emergence of Grid computing.

On a related note, the Working Groups from the GGF "Grid Security" Area [84] should be much more pro-active in making system administrators aware of their progress and, whenever possible, in getting more of them involved in the work of this group. System administrators are responsible for the security at their site and are therefore unlikely to support the deployment of a local Grid or integration of some of their local systems to external Grids unless they are satisfied that the security measures are adequate. As stated earlier in relation to users, dissemination activities on security matters should be considerably widened to target system administrators fora, mailing lists, and workshops.

Intrusion Detection Systems (IDS) seem to be a future solution for proactive Grid security. In general no system is ever completely secure; this will probably continue to be the case, as a result of errors in design, implementation or configuration. Moreover, for a variety of reasons human error will always be a factor. In such situations hackers find a chance to attack. The real threat is silent and precise attacks initiated by external organisations intent upon monetary or status gains. Attacks on the Grid through a single node may provide direct and privileged access to data on other systems. IDS may help here; its purpose is to detect in real time all kinds of inappropriate user activities, such as attempts to breach system integrity or to gain unauthorised access to information. However, there is a need to extend this concept from mere to detection to active protection, defined as a means of preventing attacks from succeeding. Further information on IDS may be found at http://www.man.poznan.pl/security.

3. Easy to use, seamless environment (3);

> *"Simplicity"*

This says it all. Some users expect setting up the middleware and configuring the files implementing the site policy to be user-friendly. This comes back to the same concept of avoiding overheads of any kind, whether by complex software installation or complex use of this system.

4. Need for complex hierarchical system (2).

> *"not sure of the answer since we're likely to have a complex situation involving users with different access privileges"*

> *"Resources at the University are shared and administered by very different agencies (Department, University, HEFC, Research Councils) and so there needs to be some control to satisfy these broad and possibly conflicting requirements."*

These points are perfectly valid and pertinent to many other respondents. They highlight the need to design a middleware capable of handling a differentiated hierarchy of users and systems and implementing complex policies over the Grid. This situation is typically more complex in Europe than in North America as the utilisation level of computing resources is significantly higher and the variations on policies and interested parties are more numerous. It raises once again the need for European centres to reflect on these matters and make their voice heard by participating actively at the relevant GGF groups (see "Section 2: Awareness," page 125 for a fuller discussion of this topic).

At Finally, this remark illustrates the views of many who consider that there will be no such thing as 'free cycles' at their site:

> *"Own resources are expensive; we are not a service provider."*

This highlights the need to make further progress with Grid economics and to standardise ways of making participation to a Grid economically sensible.

III.7.4 Summary

Overall, respondents reported that availability, reliability and ease of use were the most important added services needed from a grid service provider. Of course, it might that be argued that these are core services required from the Grid rather than added extras. However, it was also apparent that different groups of users – with different requirements, knowledge of and expectations from the grid – had different perceived Grid 'added services'. A good example of these differences is for instance the importance given to remote visualisation by the astrophysics community.

This whole area is one that would merit further investigation, as it is central to any attempt to define the user requirements for future Grid services. In particular, as Grid tools continue to be developed and become more widely available and as users become more familiar with the range of services being offered it will become even more important to integrate usability into system development. This in turn will necessitate the active involvement of users throughout the development lifecycle.

Security issues also appear central to Grid computing, and respondents expressed concerns both as prospective users of Grids and as prospective resource providers to a Grid. In addition to the usual worries (hacker's intrusion), these concerns were also motivated by the prospect of 'unknown' (i.e., non-local) users running on a provider's system and lack of control on the whereabouts of code and data.

Whilst a significant number of respondents were not opposed in principle to providing some of their own resources to a Grid, virtually all of them specified that they would only do so if they could withdraw access to Grid users whenever local users require it. However, this would be difficult to reconcile with high availability and reliability.

III.8 Section 7: Future needs

III.8.1 Foreword

This section of the questionnaire was initially included for the second ENACTS study ("The HPC Technology Roadmap") jointly undertaken by NSC and CSCISM [19]. The key issue was to elucidate how users' computational requirements are likely to evolve over the next ten years. Since these data will be analysed and discussed at great length in NSC's and CSCISM's report, they will not be discussed in detail in this report. However, all the basic data will be presented for the sake of completeness, and any point relevant to Grid user services requirements will be highlighted.

III.8.2 Influence of computational constraints on research activities

Respondents were asked to describe in what way the problems they would choose to address would differ from those they are currently working on if the current computational constraints were eliminated. The main objective was to establish if these would be scaled up versions of current problems, or completely new problems altogether.

About two third of the respondents stated that they would rather consider scaled up versions of their current research rather than tackle new problems, although a large proportion of them also specified that they may consider a different approach for this. A handful made it clear that they would use this opportunity to 'globalise' their effort by sharing both their work and data with their research community world-wide, e.g.:

> *"We would globalise our current simulation effort and the distribution of simulation products to the astronomical community.*
>
> *We would work within the Global Virtual Observatory"*

A few others were also specific about the finality of the degree of accuracy required:

> *"More realistic computer 'experiments', closely matching true experimental conditions. Mostly, scaled-up versions of current problems, but applied to systems that are unattainable at the present"*

> *"More realistic problems, closer to the chemical industry"*

Note that these results can be compared to those presented in the section 'Grid computing and Grand Challenge problems," page 143. They clearly establish the case for increased resources, and hence the case for the emergence of Grid infrastructures.

See "Appendix D: Future Needs" and N SC and CSCISM's report for a discussion of this topic.

III.8.3 Application codes

Respondents were then asked to speculate on how the set of application codes they use today will have changed 5 and 10 years from now. The data shown in Figure 53, page 184 correspond to the estimated fraction of what users estimate the codes to be (they were asked to try to get the columns to add up roughly to 100% in so far as this is possible):

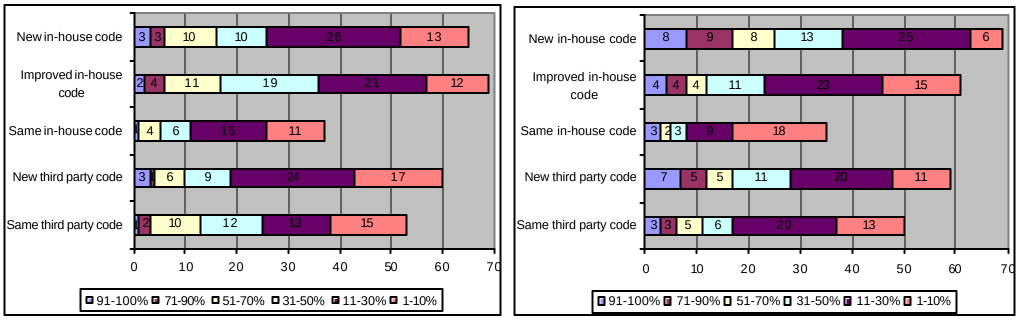


Figure 53: future application codes in 5 (left) and 10 years (right)

III.8.4 Future performance and Moore's law

The respondents were presented with Moore's law and its implications as follows:

> *"Moore's law stipulates a doubling of transistors on microprocessors every 18 months. Processor peak performance, memory density and disk storage capacity are all on a very similar curve as Moore's law. This would give an increase by a factor of about 10 fiveyears from now and a factor of 100 ten years from now."*

They were then asked to comment on how this potential performance increase would compare to their future computational needs, assuming that financial funding will be on the same level as today.

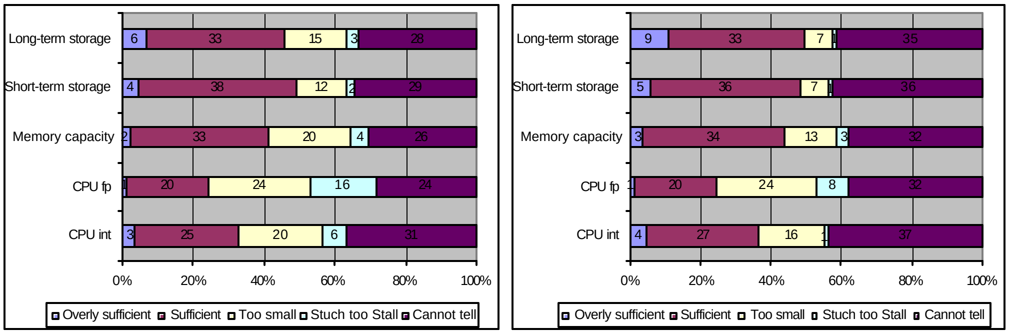


Figure 54: adequacy of Moore's law to predict requirements in 5 years (left), and in 10 years (right)

These results are particularly interesting one considers them in relation with the data presented in Figure 29, page 141, and Figure 43, page 159. Thus, CPU power (and to a lesser extent memory capacity) is anticipated to remain the major problem.

The bandwidth and latency performance seem to grow on a slower-than-Moore's law curve i.e. not keeping pace with the processor and memory performance development. Respondents were asked to comment about how this might affect their application's performance in the future.

A number of comments can be made from these data (see Figure 55, page 185). Firstly, it is obvious that limited memory bandwidth and latency may represent a problem bearing in mind that it will impact on the overall CPU performance, and the scalability of parallel systems (especially for shared memory systems). Secondly, the awareness of the importance of network performance appear much more clearly than at any point during the previous sections. A very good sign indeed! Finally, it is worth pointing out that about a third of the respondents simply could not predict the influence of any of these factors. This high proportion probably stems from the limited knowledge some respondents may have of their applications' requirements, combined with equally uncertain requirements brought by operating within a Grid infrastructure.

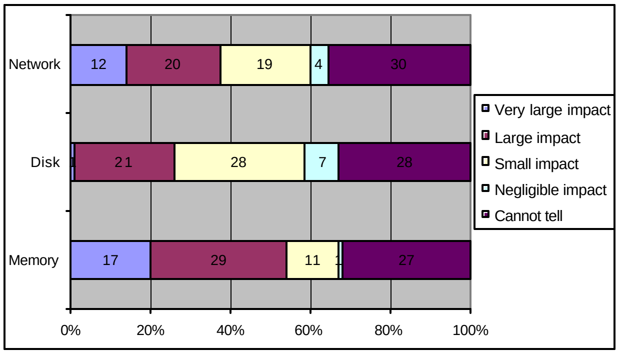


Figure 55: influence of limited bandwidth and latency

III.8.5 Predicted use of computing resources

Respondents were then asked to speculate on the importance of various computing resources for their future work.

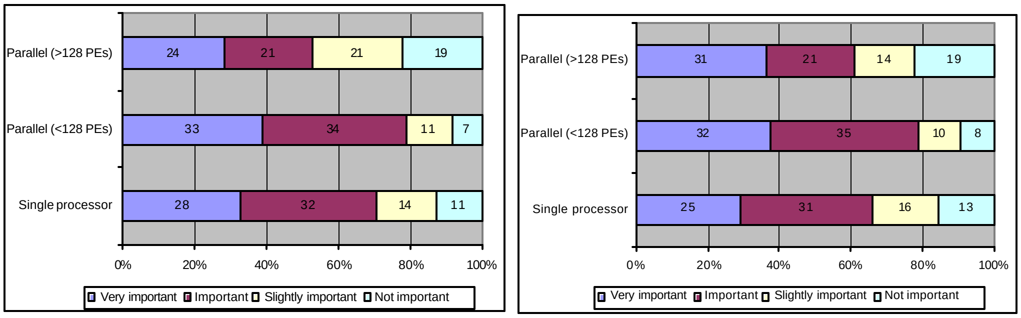


Figure 56: importance of given architecture in 5 years (left) and in 10 years (right)

The most interesting factor emerging from the data of Figure 56 is without a doubt the sustained importance of serial computing through time. Clearly, a large number of respondents regard this type of architecture as "very important", even in 10 year time. As was emphasised throughout this study, it is therefore important to devise adequate access to such resources within a Grid infrastructure. Note that models such as Internet Computing and small Grids may prove to be the most suitable models for this purpose.

The second point worth mentioning is that provision for medium parallel jobs (< 128PEs) is consistently regarded as the most important. Bearing in mind the data from Figure 48, page 162, the most popular job size seems to lie in the region of 32-64 PEs. However, it is equally important to remember that massively parallel jobs (128 = PEs) are still considered essential by a significant fraction of the respondents (especially those concerned with Grand Challenge applications).

III.8.6 Future importance of common parallel programming models

Finally, respondents were asked to comment on the kind of parallel programming models that they think will be of interest for their research work in the future, 5-10 years from now.

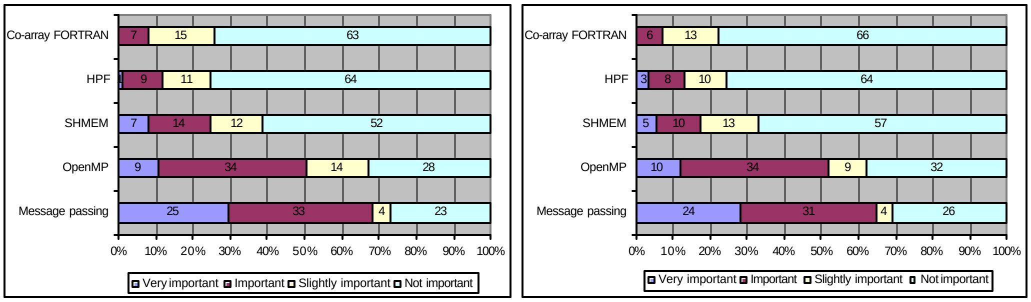


Figure 57: usefulness of common parallel programming models in 5 years (left) and in 10 years (right)

The data from Figure 57 should be considered in relation with the results from the section 'Languages and parallel environments," page 156. The most striking observation is that these data appear surprisingly stable through time. For instance, the importance still given to HPF, SHMEM, and Co-array FORTRAN in 10-year time is rather unexpected considering that most HPC centres no longer teach any of these programming models and that the latter two are proprietary. The limited uptake of OpenMP (stabilised near the 50% mark) is certainly linked the incompatibility of its programming model with the increasingly popular Beowulf platforms.

III.8.7 Summary

Based on the data presented in this section, future requirements are perceived to be highly correlated with current requirements presented earlier in this study. Thus for instance, CPU power and memory performance are expected to remain crucial bottlenecks over the next 10 years. The need for serial computers (first identified in "Most wanted resources in a Grid environment," page 142) is also confirmed.

A more details discussion of these data will be included in the second ENACTS study ("The HPC Technology Roadmap") jointly undertaken by NSC and CSCISM [19].

III.9 Section 8: Closing remarks

This section contains a summary of the closing remarks made by some respondents.

III.9.1 Further comments

One of the most common issues raised was the need for more support for the design of parallel codes. It appeared as if the Grid was one step too far for some researchers who have not yet negotiated the transition from serial to parallel programming. This was confirmed by the importance given to serial applications throughout the survey:

> *"From my point of view I see no clear transition way to migrate from the single-processor philosophy to the distributed/parallel processing.*
>
> *I would have asked for how easy would be for me to access resources that would help on this migration"*

and

> *"Difficulty/simplicity of parallel program development.*
>
> *Availability of parallel problem solving environments."*

Clearly such environments exist and could be put to good use for researchers unable to invest too much time learning about parallel programming. For instance:

> *"On the previous question I believe that other concepts that allow users to write parallel programs directly which will then use any of the above communication packages will be much more important. This concept is the basis of ZPL (http://www.cs.washington.edu/research/zpl/) and this or something like it will have more future than any of the above communication packages because of its ease of use."*

One crucial aspect is the fact that many of the respondents expressed scepticism about the need for Grid computing at all. Thus:

> *"In general I see no advantage of grid computing for end users with a need of program development and plenty of production runs at the moment. Neither the technical nor the political questions (access to resources abroad) have been solved yet. I hope that this will change in the future.*
>
> *This questionnaire took me 100 minutes instead of 20 minutes promised!"*

and

> *"It is not obvious that there will be a widespread need for Grid resources, unless of course they are made available for free (meaning that someone else is paying for it)."*

In addition to this, it appears that several groups – including large ones – are composed of researchers with only a limited understanding of the issues linked to the use of HPC facilities and Grid infrastructures. This factor already emerged in "Section 2: Awareness," page 125 and "Section 3: User profile," page 146 where it could be seen that in some disciplines like chemistry or biological sciences, the groups are principally composed of black box users who focus entirely on the science together with a handful of highly computer literate developers. These comments emphasise the need for more pro-active dissemination targeted at these research groups. Relying purely on HPC centres to carry out these activities will not be sufficient as the links between these groups and the centres can be tenuous:

> *"I don't consider mys elf as particularly literate, smart, or visionary on these issues. Some of my answers may be quite stupid, or off the point. You are most welcome to contact me again. I have tried to answer the questionnaire for the whole Theory Group which comprises some 30-40 researchers, whose computational needs vary very strongly. Usually we have only one or two truly high-end people at the same time."*

Along similar lines:

> *"Please take into account that as of currently the group has relatively limited experience in using high performance computing solutions. This means that it is hard to give very accurate answers to many of the questions posed. We do see a need for hpc in the future since the amount of data in bioinformatics is growing very quickly and since we want to take on problems on a larger scale then earlier."*

and

> *"Some of my answer are most certainly affected by lack of knowledge in the specific topic."*

Another aspect worth mentioning is the re-emergence of purpose built systems (which not long ago were thought to be on the way out) and Beowulf systems.

> *"You haven't inquired about the importance of locally maintained, in-house Beowulf clusters. We believe that this technology is becoming a major trend in supercomputing. The price/performance is unbeatable, and local control over resources is perhaps the most crucial factor in maintaining a productive computing environment. Many colleagues around the world are moving into this direction, bypassing computer centers along the way."*

This comment is important as it emphasises the multiplication of departmental resources such as Beowulf clusters and shared memory departmental servers in the research community. This trend may not only slow down the emergence of the Grid, but may also result in cutting or weakening the links between the research community and their national HPC centre - a fact worth bearing in mind when considering dissemination and consultation activities.

Another common topic in this final section was the difficulty for respondents to predict their resources beyond a few years ahead. The reasons for this difficulty are varied, ranging from the reliance on successful funding:

> *"Research funding is often on a short cycle 2-3 years and success rates are 30% so it is very difficult to predict future requirements.*
>
> *In such a large research group there are a large mixture of codes used so it is extremely difficult to give one general response."*

to simply the difficulty on predicting the future impact of emerging technologies:

> *"Really impossible to project to 10 years from now. The answers will be strongly modified by how Grid computing develops."*

and also related to the difficulty in summarising the needs and strategy of a group as a whole when it is made of independent sub-groups:

> *"Some of the questions in the 'Future Needs' section do have different answers for different sub-groups. So these should be considered more a 'weighted average'."*

Another comment emphasised the potential of Grid computing for their discipline (electronic design), although the pre-requisite seems to be the ability to run standard packages over the Grid:

> *"Please try to add to the current HPCN trends the area of providing services, running the FPGA design packages:*
>
> *-simulators,*
>
> *-VHDL compilers*
>
> *-Place and route tools*
>
> *for large multi-million gate FPGAs*
>
> *This will be one of the major CPU time consumers in the future.*
>
> *It would be great enabler for people from the east if they could run these tools (which are in general offered by Europractice... IST KA 4) remotely over the GRID..."*

This point is particularly pertinent. Unlike HPC facilities which have only devoted modest efforts to accommodate package users, the Grid ought to cater for this category of users if it really aims to build a wider user base. The technical implications of such a support are not yet fully understood.

The final comment makes an interesting point which is seldom considered within the Grid community. The presence of venture capitalists at GGF3 and their suggestion that today's Grid middleware developers have the potential to become tomorrow's successful entrepreneurs rang some alarm bells for many attendees. Would the research community at large benefit from a commercialisation of the Grid? The answer is probably not:

> *"Risk of dramatically increasing costs due to commercialisation of software on The Grid."*

III.9.2 Prize draw

All respondents were automatically entered into a prize draw for a bottle of single malt Scottish whisky. For the sake of completeness, the respondents' choice has been reproduced in Figure 58. We do not expect any correlation between these preferences and the adequacy of any particular type of Grid model…

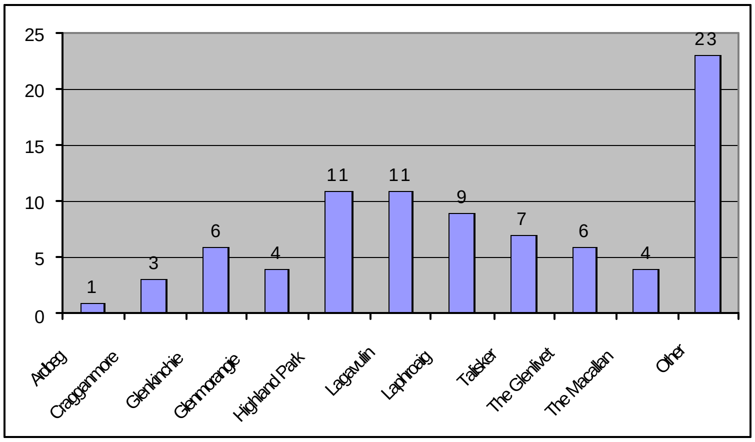


Figure 58: whisky preferences

The lucky winner of the prize draw is Prof. Dr. Ing. Ernst von Lavante from the University of Essen (Germany) who has won a bottle of Lagavulin. Well done Ernst!

… and thanks to everyone who has accepted our invitation to participate to the survey. We hope your contribution will help to make computational Grids meet your expectations and fulfil your requirements.

Part IV Executive summary and Recommendations

This section comprises three main parts:

1. a summary of the features offered by leading-edge Grid middleware technology and higher level components, based on the detailed material presented in Part II;

2. a summary of the key service requirements identified by the user community in the preceding survey; and

3. a proposed strategy to increase a successful uptake of Grid-based computational resources within the user community. This strategy will lead to recommendations aimed at both HPC resource providers and funding agencies.

IV.1 Summary of features available within a Grid environment: comparative study of different Grid models

This study aims to help users of Grid systems that can exploit heterogeneous networked computing resources by discovering under-utilised remote computers and deploying jobs to them. As many of the systems which are currently available tend to have significantly similar functions, a study of the relative suitability of such systems for being extended and used for this purpose is a logical first step. This report presents the findings of such a study that considered the systems presented in the previous section. These are LSF, Globus, LEGION, Condor, Unicore and Entropia.

The main features of the systems analysed are divided into four major categories, which are presented in Tables 1–4.

System, flexibility and interface: See Table 1. These data lead to the following conclusions.

- The majority of the systems are available in public domain. The only system that does not have a public domain version is LSF.
- Documentation is excellent for LSF, and good for the majority of the other systems. The majority of the systems are constantly being updated and extended with new capabilities.
- The majority of the systems operate under several versions of UNIX, including Linux.
- Windows NT versions are under development for Condor and Globus.
- Impact on the owners of computational nodes is typically small and configurable by the users themselves.
- The majority of the systems have both a graphical and a command-line interface. The only system without a graphical interface is Legion.

System, Flexibility and Interface						
Name	LSF	Globus	Legion	Unicore	Condor	Entropia
Version	4.1	1.1.3	1.7	3.0	6.1.17	N/A
Distribution	Commercial	Public	Public and commercial	Not finalised (part of it will probably be commercial)	Public	Public
Open source	No	Yes	Yes	No	Upon request	No
Documentation	Excellent	Good/Poor	Good	Poor	Good	Poor (good for Client side)
Roadmap	Yes	Yes	Yes	Yes	Yes	No
Linux support	Yes	Yes	Yes	Yes	Yes	Yes
NT support	Yes	Planned	Yes	Yes	Not stable	Yes
Solaris support	Yes	Yes	Yes	Yes	Yes	Yes
Tru64 support	Yes	Yes	Yes	Yes	No	Yes
Other OS support	Unix-like	Unix-like	Unix-like	All Java	Unix-like	Windows (UNIX planned)
Impact on owners of computational resource	Small	Variable (interfaces to the local resource management system)	Small	Small	Small (user-defined)	Small (user-defined)
Dynamic system reconfiguration	Yes	Yes	Yes	No	Yes	Yes
User interface	Client and GUI	Client and GUI	Client	GUI	Client and GUI	Client GUI
API	Yes	Yes	Yes	No	Yes	No

Table 21: Grid systems, their flexibility and interfaces

Scheduling and Resource Management: See Table 2. We draw the following conclusions.

- Batch jobs are supported by all systems, whether centralised or distributed. Interactive jobs are not supported by Condor and Unicore.
- Parallel jobs are fully supported only by LSF, although limited support is provided by Globus and Legion.
- Resource requests from users are supported by all the job management systems except Legion. In Globus, a special specification language, RSL, is used to specify the job requirements.
- The most flexible scheduling is provided by LSF. In Condor and Entropia, one of several predefined scheduling policies can be selected by a global administrator.
- All centralised job management systems support job priorities assigned by users.
- All centralised job management systems and Globus support job monitoring.
- Accounting is available in all centralised job management systems.

Scheduling and resource management

Name	LSF	Globus	Legion	Unicore	Condor	Entropia
Batch jobs	Yes	Yes	Yes	Yes	Yes	Yes
Interactive jobs	Yes	Yes	Yes	No	No	No
Parallel jobs	Yes	Limited	Limited	Limited	No	No
Resource requests	Yes	RSL	No	Job preparation tool	Yes	Matching
Limits on resources	Yes	Yes	No	Yes (?)	Yes	Yes
Flexible scheduling	Yes	Depends on the lower level systems	Local scheduling	Depends on the lower level systems	Defined user attributes only	Yes
Job priorities	User assigned and automatic	Depends on the lower level systems	Depends on the lower level system	Depends on the lower level system	Assigned by user and administrator	No
Job monitoring	Monitoring tools and logs	Monitoring tools and logs	?	Yes	Monitoring tools and logs	Yes
Accounting	Yes	No	?	Yes	Yes	Yes

Table 22: Resource management aspects of the systems

Efficiency and utilisation: See Table 3. These data lead to the following conclusions.

- Stage-in and stage-out are supported by LSF, Globus and Legion, and to a limited extent by Condor.
- Time-sharing of jobs is supported by LSF, Globus, Legion, and Unicore.
- Time-sharing of processes is available in Condor.
- Process migration is supported by all the centralised job management systems.
- Dynamic load balancing is supported only by LSF.
- Scalability is high for LSF and all distributed job management systems.
- Scalability in Condor is limited by a central scheduler.

Efficiency and utilisation

Name	LSF	Globus	Legion	Unicore	Condor	Entropia
Stage-in, stage-out	Yes	Yes	Yes	Yes	No transfer of subdirectories	No
Timesharing	Jobs	Jobs	Jobs	Jobs	Processes	Processes
Process migration	Yes	No, although some work is being done as part of other projects (e.g. GridLab [5])	No	No	Yes	Yes
Static load balancing	Yes	Yes	Yes	No	Yes	No
Dynamic load balancing	Yes	No	No	No	No	No
Scalability	High	High	High	?	Medium	High

Table 23: efficiency and utilisation

Fault tolerance and security: See Table 4. We draw the following conclusions.

- System level checkpointing is supported only by LSF and for only a small subset of the operating systems. Run-time library checkpointing is supported by LSF and Condor. User level checkpointing is supported by LSF and Condor.
- Fault tolerance is the best in LSF and Condor.
- Authentication based on Kerberos is provided in LSF, Globus, Legion and Unicore. Strong authentication based on SSL and X.509 certificates is provided in Condor, Globus, and Unicore. Additionally, Globus supports hardware authentication tokens.
- Strong authorisation is available in LSF, Condor and Legion, DCE.
- Encryption is clearly documented in Globus, Legion, and Unicore, but may also be present in several other systems using Kerberos and SSL.

Fault tolerance and security

Name	LSF	Globus	Legion	Unicore	Condor	Entropia
Checkpointing	User level, limited OS	No	No	No	User level and run-time library	No
Suspending, resuming, killing jobs	Yes	User only	User only	User only	Yes	Processes
Fault tolerance	Yes	Limited	Limited	No	Yes	Yes
Authentication	Kerberos	Kerberos + SSL + HW tokens	Yes	Kerberos + SSL	SSL	No
Authorization	Yes	Yes (User map files)	Yes (ACL)	Yes	Yes (different access levels)	No
Encryption	Upper level mechanisms	DES	Yes	Yes	?	High

Table 24: efficiency and utilisation

IV.2 Summary of key user requirements within a Grid infrastructure

IV.2.1 Background

Advocating the use of Grid computing by the user community invariably yields one reaction:

> *"Why? Can you demonstrate how the Grid would benefit my work?"*

This has to constitute the core preoccupation for people involved in the uptake of Grid computing *before* any of the technical aspects presented in Part III are considered.

The survey presented in Part III has provided a relatively detailed account of user groups characteristics (such as group size and scope, level of co-ordination and composition, etc.), of their expectations of Grid computing and of their current requirements. The most remarkable finding was how much these factors varied across research areas. A summary of these characteristic features is given in Table 25, page 195.

Field of work	***Physics***	***Chemistry***	***Computing***	***Engineering***	***Astrophysics***
Awareness level	88%	38%	100%	64%	100%
Practical Grid experience	12%	0%	71%	0%	29%
Scope of group (collaborations)	International and Departmental (both 29%)	Departmental (38%)	Departmental (36%)	Depart mental (64%)	International (86%)
Co-ordination level	Independently co-ordinated (53%)	One person co-ordinating (69%)	One person co-ordinating (64%)	One person co-ordinating (55%)	Independently co-ordinated (86%)
Preferred Grid model	Toolkit and small-scale Grids	Advanced and basic portals	Toolkit	Small-scale Grids and toolkit	Advanced portals
Need for interactive use	53%	38%	43%	27%	100%
Five most important factors in a Grid environment (by decreasing importance)	Faster CPUs (1.32)	Faster CPUs (1.34)	More PEs (1.18)	Faster CPUs (1.14)	Greater total memory (1.29)
	Greater total memory (1.00)	More PEs (1.03)	Faster CPUs (1.04)	More PEs (1.09)	Faster CPUs (1.21)
	More PEs (1.00)	Greater total memory (1.00)	Ease of use (1.00)	Longer run-times (1.00)	More PEs (1.21)
	Longer run-times (1.00)	Longer run-times (0.94)	Best machine for the job (0.96)	Greater total memory (0.95)	Best machine for the job (1.14)
	Better throughput (0.88)	More memory per PE (0.91)	Test scalability on more Pes (0.82)	Better turn-around time (0.73)	Resources not locally available (1.14)
Five most important added services in a Grid environment (by decreasing importance)	Availability and reliability (1.18)	Ease of use (1.06)	Ease of use (1.18)	Availability and reliability (1.09)	Remote visualisation (1.29)
	Support (1.06)	Support (1.03)	Availability and reliability (1.07)	Support (1.09)	Availability and reliability (1.21)
	Ease of use (1.00)	Availability and reliability (1.03)	Support (1.00)	Security of code and data (0.91)	Ease of use (1.07)
	Guaranteed turn-around time (0.94)	Guaranteed turn-around time (0.81)	Security of code and data (0.89)	Ease of use (0.86)	Support (1.07)
	Security of code and data (0.79)	Security of code and data (0.75)	Remote visualisation (0.79)	Guaranteed turn-around time (0.77)	Distance learning (0.79)
Two most important architectures in a Grid environment	Parallel, distributed memory (76%) and fast single CPU (53%)	Parallel, shared memory (69%) and distributed memory (44%)	Parallel, distributed (79%) and shared (71%) memory	Parallel, distributed (64%) and shared (64%) memory	Parallel, distributed memory (86%) and shared memory (57%)

Table 25: characteristic features from some key user groups

This diversity across the user spectrum suggests that there will be no such thing as an ideal Grid infrastructure. Grid user service requirements must be considered from two different angles to get a better understanding of how best to address them:

1. Consider users' aims and expectations: What do they plan to use computational Grids for? What are the expected benefits for the user community from their own point of view?

2. Consider users' requirements: What do users identify as the key features and services within a Grid environment, and why?

IV.2.2 What do users expect from a computational Grid?

One of the most reassuring aspects of the survey was the high level of support from the user community for the concept of Grid computing. Indeed, a stunning 91% of the group leaders participating in this survey believe that their group will benefit from accessing computational Grids. Beyond this though, it is important to highlight exactly what users expect from this new technology.

Summarising the discussion from "Perception and benefits of computational Grids," page 130, in order of importance users expect Grid computing to:

1. provide more cycles;
2. enable the study of larger scale problems;
3. increase opportunities to share application codes and packages;
4. provide access to sophisticated visualisation and data analysis tools; and
5. share distributed data.

IV.2.2.1 Increasing computing cycles

The most common expectation is that Grid computing will increase the level of resources, either through a better utilisation of existing resources (e.g., by setting up a local Grid based on LSF [36] or Condor [37]), or through access to 'free' cycles. In this context, 'more' has different meanings as it encompasses access to more resources within a confined period (i.e., increased throughput to cope with burst of activities) and shorter turn-around times.

The former can be easily met by deploying a local Grid within the group, department or organisation. At least three mature pieces of software, namely Condor, Sun's GridEngine [85] and LSF, are already providing an efficient and reliable solution to this problem. Whilst it can be argued that the cost of LSF has deterred many prospective users from adopting this solution, the major stumbling block seems to be caused by the lack of awareness that such solutions exist, and more crucially, the lack of in-house expertise to deploy and configure such systems. Experienced systems administrators are notoriously too expensive for many small academic groups, so a large-scale (systematic) development of such expertise is not economically realistic. Instead, the onus could be placed on HPC centres to provide 'starter kits' with adequate support to assist users with the installation and configuration of systems like Condor and to provide follow-up support for specific problems whenever required. Such a strategy is currently being put in place in the UK through its network of eScience regional centres and helpdesks [2,53,78]. Of course, it is also essential to devise an effective strategy to publicise these services within the user communities in order to raise the level of awareness. With grass root end-users in mind, the only effective means will be through presentations at national user conferences.

Increased throughput is more difficult to address as it implies the ability to increase the finite amount of resources a group has access to over a given period. The only solution is to secure access to external resources. Although this could be based on the systems presented earlier, e.g. by integrating remote resources in a Condor pool, or by more sophisticated Grid middleware such as Globus [34], the key hurdle is political. User groups have to reach agreement with other resource-owning groups and define rules for sharing access to their systems. Such agreements could for instance be established at a Faculty or University level to provide increased throughput, whilst keeping administrative and security overheads to an acceptable level. This approach (large-scale internal Grids) can also be applied to organisations and companies; indeed, successful examples of this approach have already been widely publicised [56,57,86,87].

The possibility of setting up 'topical Grids' within and/or between large structured groups is also an appealing one, although the overheads in setting these up can hardly be justified if they are only used for occasional access. Their relevance would obviously increase with the frequency of use, number of systems included and participating sites, although the administrative, political, and security overheads would grow accordingly. Topical Grids are becoming more pertinent for those groups which own their own HPC systems rather than a share on national facilities, as they have the authority to negotiate and define access policies with external (sometimes foreign) user groups.

IV.2.2.2 Tackling larger-scale problems

71% of the respondents indicated that they expected Grid computing to enable them to tackle larger or more complex problems. The usefulness of Grid computing for this purpose is not straightforward, as it requires both a net increase in the resources available to the group and access to systems which can deal with the increased requirements of total aggregate memory, storage space, better scalability, etc. Both these factors incur a significant additional cost which only a few groups will be able to afford. Although metacomputing is often presented as an affordable solution to this problem, its relevance to end-users for production purposes is still three to four years away at best. This technology has been proven to be effective for one-off demonstrators such as those presented by the Cactus team at the Supercomputing conferences [28] or for Grid testbeds [21], but the underlying technology is not yet mature enough for routine use by non-experts. It has to be emphasised that the main challenges are both technological *and* political, as site operators would need to introduce significant changes in their policy to support advanced reservations and network QoS. On the purely technological front, there are challenges both at the middleware level and at the application level, as most applications will require the inclusion of smart algorithms to Grid-enable them - typically by making them latency tolerant and bandwidth parsimonious (see [27] and [88]).

Metacomputing is particularly interesting for countries which have no high-end HPC systems but have a number of large enterprise-class servers spread across their country. Provided their network infrastructure is capable of sustaining bandwidth of 155Mbps or higher [111], metacomputing may indeed be the enabling technology which allows them to tackle Grand-Challenge problems for the first time. It should be mentioned that efforts are currently being made to develop this technology through initiatives such as the EC-funded IST Damien project [22] and as part of the GGF Working Groups and Research Groups [89], so the situation will need to be re-assessed on a yearly basis.

IV.2.2.3 Sharing application codes and packages

Creating opportunities to share application codes is in itself independent of Grid computing. Such initiatives for communities to develop and share application codes are already in place and should be encouraged further. This survey gave the impression that some users expected access through a Grid portal to 'magically' make codes and packages available, leaving the responsibility for making them available to the community to the Grid operators. A higher level of collaboration and co-ordination between groups with related research interests is, of course, a highly desirable trend. The role played by HPC centres to initiate and catalyse the formation of large consortia of users could be extended further to meet this aim. There is evidence that such mechanisms are already in existence, e.g. through initiatives such as the UK Collaborative Computational Projects (CCP) [45] spearheaded by Daresbury laboratory [90].

One point which is pertinent to Grid computing is the need to exert pressure on the commercial software companies to issue more innovative and flexible licensing schemes to allow their packages to be used on demand (on-the-fly) across computational Grids. In such a scheme, temporary access to a software could be charged as a Grid resource in the same way CPU power or network bandwidth would be charged.

IV.2.2.4 Visualisation and data analysis

This is an emerging topic: 'advanced' visualisation and data analysis. It relates principally to stream-based remote visualisation during the simulation run and 'live' interaction through computational steering (which aims to combine simulation, data analysis, and visualisation into a single package [91]), but also to standardised and user-friendly - typically web-based - tools for data analysis.

The lack of success of computational steering to date has not been caused so much by the lack of software solutions or HPC resources, as by the lack of QoS from HPC sites and the limited support from site operators. QoS is required to ensure a guaranteed minimum end-to-end bandwidth for remote visualisation (bandwidth reservation) as well as guaranteed starting time for batch jobs (advanced reservation). This latter functionality is a pre-requisite for any interactive process evolving within a batch environment. Although computational steering software supporting parallel applications has been in existence for over six years, few user groups actually use this technology on a regular basis. As with most previous points, this limited success is due mainly to the lack of awareness in the user community about the existence of this technology and what it can offer. Once again, HPC centres should be instrumental in both demonstrating the benefits of this technology to the user community and in supporting them, either through implementing the required modifications to the user codes, or through formal courses and distance learning.

Conditions are not so favourable for the emergence of standardised, user-friendly (post-mortem) data analysis tools or environments. Whilst some research groups are known to follow strict standards for their I/O operations in order to ensure portability across systems and interoperability between user groups, most smaller groups sadly do not. Thus, whilst research groups from the medical sciences [92], biological sciences [93], or climate research [94] areas typically produce interoperable data, this is not generally the case for small groups in physics. As a direct consequence, while sophisticated tools and problem solving environments (workbenches) start proliferating for the former, few attempts have been made to develop such environments for the latter. Measures should be taken to promote data interoperability within these groups e.g. by promoting the use of formats such as HDF [95] or initiatives such as netCDF (network Common Data Form [107]). Such initiatives are currently undertaken through drawing the attention of researchers to this issue but this type of action will take time to come to fruition.

The next generation of problem-solving environments will be capable of handling distributed data. Although prototypes implementing core functionalities are already available, more progress needs to be made at the middleware level to make them fully functional. To provide middleware for these data-intensive applications requires a lot of research in the areas of proper data grid architecture and basic virtual data grid technologies, such as scheduling, execution management, and virtual data instantiation. Architecture development should be a focused activity whose goal is to define a virtual data grid reference architecture that can guide the development of the virtual data toolkit as well as provide a framework for performing initial application experiments. To date, research activities on the various specific virtual data technology areas have been largely decoupled. The reference architecture will provide an integrating framework for many of these efforts, and will form the basis for cross-project collaboration. Specifically, during the next year, it is anticipated that more mature research activities will be integrated into the overall system architecture and be deployed as part of our testbed environment for evaluation within the context of application experiments. We highlight some of these ongoing research activities below[1].

Agent-based scheduling. Request management in virtual data grid environments is complicated by the complexity of the requests, the potential duration of request execution, and the dynamics of the underlying execution environment, including component failure. For this reason, request planning and execution management for virtual data grids must be very flexible, and capable of preplanning request execution in the case of failure. A promising approach is to exploit Intelligent Agents as a technology for implementing request management.

Improving Query Performance. Research at UC Berkeley has been focused on developing techniques to reduce latency and improve the interactive performance of access to virtual data products. The work performed to date has focused on two promising directions: 1) Query relaxation and associated caching techniques, and 2) dynamic prefix caching for large files.

XML-based data manipulation. The virtual data grid requires mechanisms for the creation of derived data products. At the same time, the virtual data grid requires the ability to query the collections of derived data products. An emerging standard for querying and manipulating XML files is Quilt. A standard version of Quilt is being developed called XQuery.

Data Management. There are a number of research activities that are investigating various aspects of location transparency, specifically with respect to creating and managing replicas of materialised data.

[1] More info can be found at the project web pages: DataGrid [20] and GriPhyn [96] projects.

Work at the University of Chicago by Kavitha Ranganathan has used simulation studies to examine alternative replication and caching strategies on application response time and bandwidth requirements.

Flexible Storage Technology. At the core of virtual data grid technology is the need for efficient and manageable physical data storage and movement. This work is linked to initiatives such as the DataGrid [20] and GriPhyN [96] projects and is therefore expected to come to maturity within the next three years.

IV.2.2.5 Sharing distributed data

The issue of sharing data over the Grid is intimately linked to the topic of data analysis. Major developments have been achieved in this field, mostly due to the impetus of CERN's DataGrid project [20] and its American counterpart, the GriPhyN project [96]. The need for systems capable of managing large distributed datasets and providing user-friendly, secure and high-performance mechanisms to retrieve information is not only limited to particle physics, but also applies to climate modelling, earth observation (processing of satellite imagery) [97], medical imaging, genomics etc [104]. The functionality of the DataGrid will also cover these particular disciplines. Great progress has been made as part of the DataGrid project. It is worth pointing out that the DataGrid software – obviously this does not cover the data– will be released as open source and made available through the release mechanism of the GGF (open source licence) as will any other Grid software produced by other projects of this kind. The first release is expected in December 2001, once validation is complete.

IV.2.2.6 Other aspects

Four central issues emerged from the discussion in section "Perception and benefits of computational Grids," page 130:

1. the need for standardisation of access and operation methods (mostly user interfaces and functionalities);
2. the need for a user-friendly system offering an integrated and seamless access to heterogeneous resources across the grid (this implies middleware taking care of portability and interoperability, but also access to fully-featured tools, etc.);
3. the need for collaborative environments and sharing of resources, including both hardware, data and application codes; and
4. the need to optimise the usage of existing resources.

Interested readers should refer to section "Section 2: Awareness," page 125 for a fuller discussion of all these points.

IV.2.3 Which services will user require within a Grid environment?

"Section 2: Awareness," page 125 and "Section 6: Security and services," page 174 considered the key features users would expect to find within a Grid environment. A summary of these findings is presented in Table 25, page 195 under the categories *'most important factors in a Grid environment'* and *'most important added services in a Grid environment'*.

A detailed discussion of these findings, and in particular an analysis of discipline specific variation and requirements is available in Part III. The core requirements essentially consist of increased levels of conventional hardware resources such as faster CPUs, more processors or greater total memory. Whilst the need for greater total memory and access to more PEs is a clear indication of the users' intention to tackle larger or more complex problems (see IV.2.2.2), the need for faster CPUs also encompasses the issues raised in IV.2.2.1, i.e. more cycles and increased throughput.

It is not surprising that these were the most highly-rated factors, as these are the ones typically put forward during procurements for HPC systems. However, they are not linked to Grid computing per se.

In this respect, the following categories are much more relevant i.e. the need for (in decreasing order of importance):

1. longer run-times;
2. ease of use;
3. better throughput;
4. best machine for the job; and
5. access to resources not locally available.

In addition to these requirements, users also regarded the following services to be important within a Grid environment (again, in decreasing order of importance):

1. availability and reliability;
2. ease of use;
3. support (e.g., Technical queries);
4. security of code and data;
5. guaranteed turn-around time; and
6. remote visualisation tools.

IV.2.3.1 Longer run-times

The need for longer run-times was flagged by a significant number of respondents (see 'Section 4: Application profile," page 156). This suggests that there is a need for systems within a Grid to implement complementary policies rather than duplicate the same access restrictions. Thus, a generic Grid may include systems configured to provide a better environment for large-scale production runs (e.g. with large maximum run-times and storage areas), systems configured with short express queues providing fast turn-around time for testing and benchmarking purposes and systems configured for interactive tasks such as code development and visualisation. One of the immediate benefits of Grid computing for the user community is its potential to offer more flexible policies without compromising the infrastructure's overall level of utilisation.

IV.2.3.2 Ease of use

It was emphasised in 'Section 2: Awareness," page 125 and "Section 6: Security and services," page 174 that ease of use is one of the main requirements put forward by the user community. The concept of "ease of use" however is not restricted to an easy to operate interface between the Grid middleware and the users, but also encompasses concepts such as portability, interoperability and standardised interfaces. The user community made it clear that it would reject any system which would oblige them to learn yet more complicated scheduling syntax. Grids *must* be simple to access and operate, and if possible even more user-friendly than typical HPC centres.

The relatively low level of support for Grid portals (only 46%) was unexpected, as this might be assumed to be the most obvious way to enable users to interact with Grids in an intuitive way. The reasons for this lack of support have been established in "Application portability," page 164. These are:

- fear of being forced to use 'standard' application codes;
- assumptions that only limited capabilities can be implemented within a portal (typically, inadequacy for non-stable codes and applications requiring pre-/post-processing); and
- fear of cost for getting custom application code integrated within a portal (lack of expertise, financial cost, etc.).

IV.2.3.3 Better throughput

This issue has been already discussed in section IV.2.2.1.

IV.2.3.4 Best machine for the job and access to remote systems

These factors are two of the main selling points for many users in the astrophysics community. They are also important more generally for users in other research areas. Examples include users who would rather develop a parallel code with OpenMP but do not have access to shared memory systems, or users who would find it economically more sensible to run on custom architectures (such as APE1000, or GRAPE) but cannot afford to buy such a system for their own group.

Two possible approaches could be made. Firstly, topical Grids are the most suited to user groups with specific requirements (typically, those for whom custom-built systems exist). For more general use, Grids should be constituted of a mix of architectures rather than offering more of the same. This is particularly relevant to the concept of using the best machine for the job. The advent of Grid computing is seen by many as an excellent opportunity to remove applications with modest requirements (typically those not too sensitive to latency, or only requiring a small number of PEs) from expensive high-end systems. By integrating cheaper hardware such as Beowulf systems or small shared memory systems within a computational Grid, it would be possible to onfigure the middleware to re-direct this class of jobs away from high-end systems. This type of policy would hopefully result in increasing the resources available to tackle Grand-Challenge problems. Of course, this will only be achievable when HPC sites include smaller/cheaper systems to migrate jobs onto and when further progress has been made in the field of Grid economics. Indeed, the most effective mechanism available to operators to optimise use of their resources will be to tune the cost model for access to their Grid. This relies on the availability of a single currency with conversion factors between the various resource types. A successful cost model must incorporate the concept of supply and demand (i.e. be dynamic) but must also take into account the underlying cost of the equipment and combination of resources requested. In this way there would be a strong incentive to run jobs which do not require high-end systems on cheaper alternatives.

IV.2.3.5 Availability and reliability

The respondents regarded availability and reliability as the most desirable characteristic of a Grid infrastructure. This highlights the need to wait for the technology to be mature (i.e. stable) before migrating users onto this type of infrastructure. Although Grid testbeds are invaluable in making significant progress in developing Grid middleware and in tuning configurations and validating policies, they will not provide an adequate environment for end-users. It is particularly important to avoid migrating grass root users to 'at risk' or experimental services. Bad experiences of Grid computing are very likely to discredit this technology in the eyes of the user community.

The issue of availability is also particularly pertinent considering the terms under which user groups would be willing to share part of their resources by integrating them to a Grid. As shown in "Section 6: Security and services," page 174, most groups who are open to this idea do indeed want to preserve the right to withdraw access to their resources whenever it interferes with he work undertaken by their local groups. The only way to reconcile this requirement with high availability is to develop job migration of running processes in a seamless fashion. Although promising results have been obtained in this area (see Part II), the transition from Grid testbeds to production environments is still a couple of years away.

IV.2.3.6 Support

Effective user support proved to be instrumental in making HPC accessible to a wide audience. The same will be true in the context of Grid computing. Users expect support groups or centres to help them to get started on their Grid. This support encompasses not only information on how to get started with this new type of infrastructure, but also how to deploy and configure small Grids locally and how to Grid-enable their applications to exploit fully the potential of this new technology. The most effective approach will be to provide a combination of starter kits, documentations and skilled support staff accessible via a helpdesk for solving specific technical problems, as well as support from the Grid operators to help with Grid-enabling applications whenever required. Such structure and services is currently being put in place in the UK [2,53,78]. Note that although formal training and distance learning were not rated so highly by users, these may prove effective, for example for training support or specialist staff responsible for their own local group.

IV.2.3.7 Security of code and data

Concerns for the security of codes and data were relatively high. This arose from the idea of having codes and data transparently migrated to remote systems without their owners being aware of their location. Although Grids purport to offer a seamless environment, it is important to enable users to track the whereabouts of their code and data through user friendly monitoring tools should they wish to do so. Using these tools, users should be able to "switch off" the seamless mode, and regain control of their application and data. Building user confidence during the early days of Grid computing will not be easy, and possible breaches of security (whether they are simple intrusion, corruption or theft of data, etc.) would quickly deter them from using this type of infrastructure. This also applies to system administrators who will oppose vehemently the integration of any system they are responsible for into a Grid unless they are completely satisfied that the security mechanisms in place are sufficiently robust and reliable. Assuming that security will be based on public key certificates, one of the remaining points of discussion concerns the trustworthiness of the authentication methods used by external certificate agencies (CA). The default Globus method (E-mail based) will not satisfy many systems administrators! The alternative for local Grids is to set up their own CA.

Hopefully, security rightly figures as the most active Working group within GGF.

IV.2.3.8 Guaranteed turn-around time

The demand for guaranteed turn-around time is not specific to Grid computing. The same need already arises with HPC facilities. This requirement is part of a wider need for the implementation of Quality of Service (QoS) taken in its wider context. In addition to the traditional meaning of minimum guaranteed (network) bandwidth, the concept also extends to include other types of resource (e.g., minimum temporary storage space, minimum Gflops/s, etc.) and policy and scheduling issues such as minimum guaranteed throughput or turn around time.

The implementation of a guaranteed turn around time could in principle be already in place in HPC sites. The main hurdle is the adoption of a new approach to scheduling which would include priority queues (say, high, normal, low) with associated guaranteed turn around time and differentiated costing. The second, more technically challenging issue is how to incorporate the ability to migrate a job to another system on the Grid whenever this is required to meet the guaranteed turn-around time. This capacity should also support running applications to implement fault tolerance.

Guaranteed turn-around time is also one of the critical services requested by industrial users (together with security of data and code, and access to long-term secure storage), as this is an absolute essential requirement for time-critical applications.

IV.2.3.9 Remote visualisation tools

The demand for remote visualisation and computational steering environments varies considerably across user groups (see "Section 6: Security and services," page 174). Indeed, the relevance and usefulness of visualisation is highly application dependant. However, it was clearly established that user groups who are already using this technology routinely, namely the astrophysicists, rate it as *the* most important service in a Grid environment. No doubt many other research areas could also benefit from remote visualisation and computational steering, especially for parameter steering. The main reasons for this limited success are the lack of awareness about the benefits (and availability) of this technology, a lack of understanding about how to incorporate such capabilities within existing codes, and finally, sometimes a lack of a suitable environment (queues) to run such jobs. Just as for many of the above points, the HPC centres could play a central role in promoting the emergence of standard solutions, demonstrating the benefits of this technology to grass root users and using their technical expertise to produce distance learning materials on how to implement these capabilities.

IV.2.4 Miscellaneous considerations

A number of other important issues also emerged throughout the questionnaire. Although they did not fit into any the categories of sections IV.2.2 and IV.2.3, these issues have nonetheless a definite relevance to this study. They are:

1. The difference in awareness level across research areas is quite noticeable and will have implications in terms of strategy through funding and dissemination activities. Different research groups will adopt the technology earlier than others, based on the following combination of factors:

   - the benefits they expect from computational Grids (see "Section 2: Awareness," page 125);
   - the type of services they will require (see "Section 2: Awareness," page 125, and "Section 6: Security and services," page 174);
   - their structure, level of co-ordination, and existing collaborations at a national and international level (see "Section 1: Start," page 121, and "Section 3: User profile," page 146);
   - the type of resources they own and their willingness to integrate them within a Grid (see "Section 5: Infrastructure," page 170 and "Section 6: Security and services," page 174); and
   - their existing links with HPC centres.

2. There is a much greater diversity of working practices, size, structure collaborative links and applications' characteristics and requirements across research areas than within any single area (see "Section 3: User profile," page 146). This suggests the suitability of topical Grids for large, well co-ordinated groups involved in international collaborations.

3. The type of architecture deemed essential within a Grid infrastructure also varies considerably across research areas (see Table 25, page 195 and "Most wanted resources in a Grid environment," page 142). In particular, the reliance on large MPP systems has to be better understood as such systems will progressively be superseded by cluster of shared memory systems.

4. The preference for a particular Grid concept varies not only across disciplines but also depends on the level of experience of the users concerned (see Figure 35, page 150 and "Features of mainstream research areas," page 147). Thus, portals are clearly favoured by the less Grid literate whilst toolkits (and in particular Globus) are put forward as the best solution by those at the other end of the spectrum. This suggests that the development of portals for mainstream application codes and packages (such as Gaussian) is the most effective strategy for the introducing the concept of Grid computing to groups from the chemistry and bioinformatics areas (see "Section 3: User profile," page 146). It is likely that the additional features of portals, such as the ability to generate configuration files and perform sanity checks on these latter, will also prove important to these communities.

5. The preferred scope for computational Grids (local, national, topical, etc.) correlated directly with both the type of collaborations groups were involved in and the amount of resources they directly control.

6. The composition or structure of the research groups in term of proportion of black box users and user developers i.e. the distribution of their users' level of computing expertise, suggest that many groups in physics and engineering are often making a low re-use of their codes (see "Section 3: User profile," page 146). These practices result in a plethora of codes 'in development' which are inadequate to be integrated within a portal or problem solving environment. This was illustrated by the high frequency of recompilations required between successive runs (see "Section 4: Application profile," page 156). Steps should be taken to encourage community led initiatives to develop standard skeleton codes following the examples of say, Cactus [27] or DL_POLY [98].

7. Application bottlenecks appear roughly uniform across applications areas: CPU performance, memory to CPU bandwidth, and memory capacity were reported as the top 3 bottlenecks across all areas. However, network performance starts to become an issue for the computing, engineering, and astrophysics communities, whilst data storage capacity is of concern to the chemistry and astrophysics communities (see "Section 4: Application profile," page 156). Note that network

performance will be one of the main bottlenecks to any user who makes routine use of computational steering or metacomputing. Mechanisms providing job migration (e.g, for fault tolerance or guaranteed turn around time) and access to distributed datasets will be equally affected by poor network performance.

8. Following recent investments in the national and transnational network infrastructures, the bottleneck for end-to-end bandwidth has now been displaced to within institutions and universities (see "Section 5: Infrastructure," page 170). A successful uptake of services such as remote visualisation will require upgrades to local networks for which funding can prove difficult to secure.

9. Security is a matter for researchers both as users of Grids and as prospective resource providers. Indeed, should a group decide to set up their own local Grid or even join an existing Grid, it is important that the security framework can be fully understood, operated, and trusted by the group's systems administrators and support teams (see "Section 6: Security and services," page 174).

10. Although it was shown that most application codes could be made fully portable with few modifications, it is important to keep on promoting code portability within the research community (see "Section 4: Application profile," page 156). The same remark also applies to data interoperability.

IV.3 A proposed strategy for an effective uptake of Grid computing within the user community

IV.3.1 Preliminary considerations

The amount of effort and investment in Grid related research and infrastructures has reached an unprecedented level in computing and this trend is expected to even accelerate over the next three years under the impetus of the EC's framework programme 6 (FP6). This is exemplified by large programmes funded by the EC's FP5 IST programme and complementary national programmes such as the UK E-science initiative (£118M). However, the real objective of this effort is to get users to adopt this technology rapidly once it becomes mature. As has been emphasised throughout this report, this requires the ability to demonstrate to grass root users the direct benefits of this technology for their research. Unlike parallel computing for which the benefits were obvious and easy to understand with no 'hidden catches', the situation for Grid computing is rather more complex. So, what is the most effective approach to ensure a smooth uptake of Grid computing within the user community? The key factor is to adopt a progressive, staged approach demonstrating actual benefits for specific problems and placing less emphasis on the 'visionary' global Grid.

IV.3.1.1 Stage 1: Solutions technologically ready for deployment (but requiring political action)

1. Deployment of local Grids within universities and organisations using Condor, Sun GridEngine, LSF, or other equivalent systems:

Benefits:

- Optimise use of existing resources (both serial and parallel).
- Improve overall throughput for bursts of activity (depends on existing average load).
- Potential to increase flexibility of batch operated resources by implementing complementary policies across these systems ⇒ improved support for a wider range of applications (production, testing, visualisation, etc.).
- Support for the implementation of sophisticated access restriction rules taking into account the systems load (system dependant), etc..
- Keeping the grid internal to an institution will make it more secure and will require a lower administrative overhead.

- Get users, support and systems staff used to the concept of grid computing ⇒ start building trust in the technology.

Requires:

- Support staff capable of installing the system and defining the access restriction and policy rules and/or support from external expert staff.
- Adequate level of computing resources to meet the expected demand (typically three enterprise class servers or more).
- Modern local network infrastructure (fast Ethernet or better).
- Political willingness within large departments (or even between different departments).

2. Deployment of national topical Grids based on Globus and Condor-G:

Benefits:

- Optimise use of existing resources (typically enterprise-class servers and small MPPs).
- Improved overall throughput for bursts of activity (dependant on existing usage level).
- Potential to increase flexibility of batch operated resources by implementing complementary policies across these systems ⇒ improved support for a wider range of applications (production, testing, visualisation, etc.).
- Increase opportunities to get access to "the best machine for the job".
- Keeping the Grid available to a small number of known users makes it more secure and will require a lower administrative overhead.
- Small number of (typically) stable production codes makes those latter ideal candidates to be integrated within a single portal system or problem solving environment.
- Get users, support and systems staff used to the concept of Grid computing ⇒ start building trust in the technology.
- Promote resource sharing (code, data, hardware and expertise) and co-ordination at a higher level.

Requires:

- Support staff capable of installing the system and defining the access restriction and policy rules (and/or support from external specialist staff).
- Some of the participating sub-groups must own a suitable level of resources (say, at least three medium to large parallel systems).
- Modern network infrastructure, i.e., Fast Ethernet (or better) locally, and 155Mbit/s (or better) links between participating sites.
- Political willingness across user groups concerned, and more formal structure to define the policies and access restrictions (existence of sub-groups with their own PIs).

IV.3.1.2 Solutions ready within the next two to five years

1. Deployment of national Grids linking HPC national and regional facilities. Note that the typical load of such systems is so high so the only way to improve the management of these resources is by migrating small jobs to more adequate systems.

Benefits:

- Increase opportunities to get access to "the best machine for the job".
- Improve throughput and turn around times of grand challenge applications by diverting small jobs and jobs with low requirements on cheaper systems.
- Potential to increase flexibility of batch operated resources by implementing complementary policies across these systems ⇒ improved support for a wider range of applications (production, testing, visualisation, etc.).

Requires:

- Reliable and user-friendly tools for accounting on the Grid (both for the Grid operators and for the groups' PIs).

- More sophisticated economic model for charging, including support for supply and demand and differentiated services (guaranteed turn-around times).
- Modern network infrastructure, i.e., Fast Ethernet (or better) locally, and 155Mbit/s links (or better) between participating sites.
- Adequate level of resources to meet the demand (benefits will be fewer and less obvious if the systems integrated are already reaching average utilisation levels of 80%).
- Integration of cheap Beowolf and Enterprise class shared memory servers to re-direct small jobs off high-end systems.
- Political motivation from the resource owners (research councils, universities, third party private company, etc.) To participate and abide by a code of conduct (e.g., regarding SLAs and to prevent economic dumping).
- Overall acceptance of the concept by users.
- Support staff within existing infrastructure and policy to assign responsibilities and liabilities (compensation and penalty schemes).
- Development of course material and documentation on how to get started with Grid computing.
- Initiatives to Grid-enable some suitable applications codes.
- More reliable and tested security models.
- Centralised Certificate Authority (CA).

2. Growth and merging of University Grids described in section IV.3.1.1 (merging or two or three University Grids):

<u>Benefits</u>:

- All the benefits described for the simple local Grids (most of them scaling up with the number of systems).
- Increased opportunity for getting access to “the best machine for the job”.

<u>Requires</u>:

- More scalable tools and middleware (depending on number of systems).
- Political determination to make it happen: co-ordination between groups of support and system administration staff as well as between resource owners/decision makers, arbitration mechanisms, etc.
- Common Grid currency (*aka* Grid token or credit) and ubiquitous accounting mechanisms.
- Policies including basic Grid economics concepts (supply and demand).

3. Data Grids: as the EC’s DataGrid project is completed and its software is made available under Open Source, opportunities will arise to deploy (and extend if required) some of its components for other data intensive applications:

<u>Benefits</u>:

- Access to reliable, scalable, and highly tested software components at no cost.
- Guaranteed long-term support for this software (bug fixes, improvements, etc.)

<u>Requires</u>:

- Expertise to deploy and configure the software.
- Adequate level of resources.
- Many other factors not yet fully understood (will become clearer as the DataGrid project progresses).

IV.3.1.3 Accompanying measures

The above course of action has an increasing number of pre-requisites as the solutions become sophisticated. It is therefore important to initiate the following accompanying measures at least a year before the planned start of a service:

1. Infrastructures and technological issues.

    a) The most common networking bottleneck for end-to-end bandwidth has been displaced within the users' local networks. It therefore important to set up appropriate funding lines to improve local infrastructures. A scheme favouring sites committing themselves to deploy a small Grid within their department or university would be strategically judicious.

    b) Setting up a network of regional support centres helping the deployment of local Grids by supplying "starter kits" and related documents and training local support staff and system administrators. The onus could be placed on the groups' support staff to deliver user support for their users' specific technical queries.

    c) The success of a computational Grid, regardless of its scope, will depend on the availability of sufficient hardware resources. As prospective providers are only expected to grant a lukewarm support to this idea, initiatives must be devised to "encourage" them to make these resources available. An effective scheme could follow the approach taken by the Canada Foundation for Innovation (CFI) [99]. This organisation funds strategic research infrastructure to help strengthen research training across Canada, with a strong emphasis on collaborative and multidisciplinary projects. However, one of the clauses attached to their funding is that access must also be provided to external users; for computing equipment, the level of access for external users has been set to 20%. It is perfectly conceivable to adapt existing funding schemes for equipment (such as the UK research councils' JREI programme [100]) accordingly. With such adjustments, the grant for HPC hardware would become conditional on making a fraction of the resources funded available for Grid use.

    d) On a purely technical level, support for seamless job migration is probably the most important feature currently missing from Grid middleware. Indeed, this feature will be a pre-requisite for increasing the throughput of Grand Challenge applications and guaranteeing the success of time critical applications (by enabling fault tolerance and guaranteed turn around time). Initiatives like the IST-funded Gridlab project [5] are excellent starting points but further research will be required in this area.

2. Policy issues.

    a) Promotion of community led code development, code re-use and portability, as well as data interoperability through the generalisation of structures similar to that of the UK Collaborative Computational Projects (CCP) [45] or DIRECT 'Data int er-operability group' [105].

    b) Promotion of the formation of larger and more structured consortia of users in view of establishing a concerted strategy for resource sharing: pre-requisite for topical Grids.

    c) Building up users' trust in the technology: demonstrate immediate benefits for *their* work (more efficient use of existing resources, increased flexibility, e.g., maximum run-times, and potential for higher throughput). This activity must be undertaken at user conferences (*not* at HPC/Grid meetings from which grass-root users are absent) using demonstrators and mainstream application codes used by the target community.

    d) The priority is probably to accelerate the deployment of (University-wide) local Grids for the reasons outlined earlier. However, the main pre-requisite will be the willingness from the various research groups and departments to include (part of) their resources to this Grid. This implies high-level decisions, and as such will depend on discussions held between high-ranking university staff issued from multidisciplinary background.

    e) Establishing a link between the Grid research community and the user community at large. This may require the creation of a dedicated GGF Working Group composed of experienced HPC user support staff. HPC facilities such as those from the ENACTS network would constitute an ideal link between these two communities. The role of this Working Group (in consultation with the Applications Working Group) would be to set up demonstrators to deploy at user conferences and ensure an effective (and relevant!) dissemination within the user community. The constitution of this group is also essential to ensure that the information obtained from the Grid developer community is more honest and specific about the true functionality and level of development of their systems. Well-known examples have been reported in the past of mainstream middleware cultivating ambiguity about this!

f) Grid economics is one of the key pre-requisites for the deployment of large-scale Grids. It is particularly important for Grid operators to understand how to tune their policies to maximise (adequate) usage of their resources and implement a fair charging scheme for their users. This will require the creation of Grid currencies and economical models. Some progress has been achieved recently (see [51] and [80] respectively) but much remains to be done. The GGF may consider forming a Research Group focussing on this specific issue, possibly inviting mathematicians working in related areas (optimisation) to take an active part?

g) The GGF GCE Working Group [63] should be pressed to standardise (or at least make recommendations) for the development of computational Grid portals. Many user groups –not to mention the industry – will be reluctant to invest any resources in such an effort as long as standards will not have been finalised. Note that portals will be essential to users from emerging disciplines such as bioinformatics and medical research which expect to drive their application tomorrow on the Grid the same way they do it today on their desktop machine.

h) As security concerns are often put forward as the main stumbling block for the deployment of computational Grids, it is particularly important to keep system administrators informed and interested in Grid security. This means both undertaking pro-active dissemination activities at system administrator conferences, and getting more of them actively involved in the GGF. Suitable funding lines should be set up to this effect.

i) In the long term, Grid computing will have a considerable impact on HPC as we know it. As HPC centres are ideally placed to play a central role in this revolution, it is important to discuss the future role that they could play in this revolution and re-define their role within a Grid-centric role.

IV.3.2 A new role for HPC centres?

HPC centres are in a particularly favourable position to become the key players for the uptake of Grid computing. This is due to a combination of skilled and experienced staff and their long running relationship with the user community. Their proposed strategic role in this context can take the form of one or more of the following actions:

1. Promoting best practice:

HPC centres are best placed to promote best practice, particularly in the field of code portability (with performance hit) and data interoperability through a combination of training (formal courses, degrees, distance learning, etc.), and initiatives to improve legacy codes. They may also teach the concept and basics of Grid-enabling applications to the user community the same way they taught MPI five years ago.

Another important role in this area is to put in place and provide support to structures similar to the UK Collaborative Computational Projects (CCP). Such structures are invaluable to improve user groups' working practices, in particular with respect to promoting the idea of code re-use.

Another ideal mechanism for HPC facilities is through EC-funding for an initiative similar to the IHP-funded TRACS [7], (see also the Access and Minos programmes [69]), but with an emphasis on Grid computing instead of HPC. Its remit would be to teach European researchers how to Grid-enable their application, thus demonstrating the benefits of Grid computing for their own research. Ideally, the scheme would allow participating centres to set up their own Grid across Europe as part of the programme.

2. Support to set up local Grids:

HPC centres are ideally placed to develop "Grid Starter Kits" and associated supporting documents, and advise external support and systems groups to deploy their own local Grid. This role could be extended further by setting up Grid centres of excellence which could act as mediator and facilitator to sort out political deadlocks within research groups and universities.

3. Training and education:

In addition to basic support for Grid deployment, funding will also be required to develop new training material covering the Grid-enabling of HPC applications. This area will cover both the field of advanced programming models and technologies like computational steering. Another important area for user groups without a strong HPC background is the integration of their production codes within a portal environment. Obviously, such an initiative will only be supported on a large-scale *after* the framework for portal development has been standardised by the GGF GCE Working Group [63]. This is another reason why HPC centres must be much more pro-active in such groups. In the meantime, funding from national research councils could be provided to integrate a few, carefully selected, mainstream application codes such as Gaussian.

4. Liaison between the Grid research and user communities:

The importance of targeted dissemination at user conferences has been established, especially the formation of a User-liaison Working Group as part of the Applications Research Group. This is a role which is by definition tailored for HPC centres' applications consultants and support staff. This group will be 'the missing link' between the user and Grid communities. Its duty will be fulfilled by reaching the users at their own conferences where they will disseminate the benefits of Grid computing by exhibiting application demonstrators based on relevant problems and solutions.

5. Practical Grid research for user services and participation to testbeds:

Many European HPC centres have made only a timid contribution to the Global Grid Forum to date. As policy issues for European and US sites are likely to be quite different (see "Section 2: Awareness," page 125), it is important for European HPC centres to ensure an adequate representation at this forum and improve their contribution to groups working on both policy-related matters and the underlying enabling technology. European HPC centres should also be encouraged to participate more actively in Grid testbeds (even for one-off demonstrators such as those set up for the Supercomputing conferences) and form more active links with the networking community.

It is important that HPC centres start supporting new classes of requirements which have so far been ignored within the HPC context. These include serial applications, computational steering and package users. There is also an urgent need for HPC centres to start thinking about more sophisticated charging schemes with a view to implementing differentiated services. Small Grid testbeds could be deployed to this effect within HPC centres to experiment with Grid economics and its impact on local services.

6. Widening the reach of Grid computing

This covers two distinct areas: industry, and the section of the research community with no previous encounters with scientific computing. HPC centres have proved to be very efficient technology providers to industry as part of the EC's technology transfer network [101]. However, Grid based solutions will not be relevant to industry as long as the underpinning technology is not sufficiently stable and has not undergone standardisation. This provides yet another motivation for European HPC centres to take a more pro-active stance in GGF groups and catalyse the standardisation process. For non-computational scientists Grid computing has far reaching benefits. This includes improving human-to-human interactions through problem solving environments, not only for conventional users of HPC, but also for researchers with no previous links with computational sciences such as social scientists. For example, Access Grid technology should be publicised to *all* researchers, thus creating new opportunities for HPC centres to become involved in the development of Access-Grid enabled problem solving environments. This expertise will be directly relevant to both the academic HPC community and to industry alike.

7. Miscellaneous considerations

This survey highlighted the fact that there has been a considerable expansion of local resources owned and operated by research groups, accompanied by a relative shrinkage of the HPC national facilities user base (see “Section 5: Infrastructure,” page 170). It is important to point out that modifying the balance of capability vs. capacity computing in favour of the latter will not only make Grand Challenge problems unreachable, but it will also sever the link between the user community at large and the HPC community. HPC centres are not only about providing computing resources but play an important part in the overall infrastructure.

The second issue worth raising is the need for HPC centres to develop further expertise in networking and data management. This can be achieved in a variety of ways, for instance becoming a tiers-2 centre. However, this role will not only require the development of expertise in data replication and caching, but will also require a broader understanding of technologies such as Corba and investment in data storage and networking infrastructure.

The final point related to HPC centres is the need for research councils to make SLAs more flexible during the transition period of their national centres in a national Grid node. Indeed, time will be required learn how to tune policies to reach the same level of effectiveness as in a non-Grid environment.

IV.3.3 Closing remarks

Grid computing is perceived by most users as a solution in search of a problem rather than a solution to any specific problem. This state of mind is remarkably well illustrated by an article published in the Economist on June 21st 2001 [102]:

> *“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. There is some truth in this. Many of the Grid projects running today resemble solutions in search of problems.”*

This is a very cynical and critical statement indeed. However, the present study has clearly established that Grid computing does have the potential to bring major benefits to the research community at large. For example, the benefits of Grid computing are obvious for applications such as the LHC experiment or the ESA’s earth observation programme - for which it represents a true solution to a true problem. On the other hand, it is inevitable that some research groups and organisations will try to ‘tap’ into Grid-oriented funding lines with the intention of pursuing only their current activities with minimum effort into the advancement of Grid computing at large. Of course, this can be observed for most strategic funding lines and is in no way Grid-specific.

Above all, the main challenges Grid computing will have to face will be political. This statement has been echoed by Bob Aiken (Cisco University Research Director) at GGF1 in Amsterdam, when he warned that the biggest challenges to the successful deployment of the Grid will be social and political rather than technical.

Ultimately, the acceptance of Grid computing will be accepted by the user community will be facilitated if one bears in mind its long-term objective. As the rapporteur from the EC’s workshop on “Expanding the Grid reach in Europe” judiciously pointed out [70]:

"Crucially, it [E-Science] is not just about connectivity it is also about knowledge and understanding… Grids are not just about running computational jobs and moving data about. They are much more fundamental than that. The key is to enable human-to-human interaction in a distributed collaborative environment worldwide."

Appendix A Bibliography

[1] Development of an Inter-Disciplinary Round Table for Emerging Computer Technologies (DIRECT). See http://www.epcc.ed.ac.uk/DIRECT

[2] UK National e-Science Centre (NeSC). See http://www.nesc.ac.uk

[3] The Next Generation of European Research Networking (GÉANT). See http://www.dante.net/geant

[4] Global Grid Forum (GGF). See http://www.gridforum.org

[5] Gridlab: a Grid Application Toolkit and Testbed. See http://www.gridlab.org

[6] CrossGrid. See http://www.crossgrid.org

[7] Training and Research on Advanced Computing Systems (TRACS). See http://www.epcc.ed.ac.uk/tracs

[8] 'High-Performance Computing for Complex Fluids' consortium (UK EPSRC-funded, grant GRM/56234). See http://www.ph.ed.ac.uk/cmatter/cgi-bin/archive/show.cgi?db=projects&id=19

[9] Reality Grid. See http://www.realitygrid.org

[10] Socrates Artificial Inteligence and Neural Network Tools for Innovative ODL (Socrates AINN). See http://www.dsp.pub.ro/SocratesAINN

[11] EPCC MSc in High-Performance Computing. See http://www.epcc.ed.ac.uk/msc

[12] MyGrid: directly supporting the E-Scientist. See http://www.mygrid.org.uk

[13] In 1973, the computing pioneer and 3Com Corp. founder Robert Metcalfe made the observation that, *"the power of a network increases exponentially by the number of computers connected to it. Therefore, every computer added to the network both uses it as a resource while adding resources in a spiral of value and choice."* The law has been used to describe the rapid growth of the Internet and the World Wide Web, but also as way say that the value of the network increasing according to the square of its users' increase.

[14] Supercomputing 2001 (SC2001). See http://www.sc2001.org

[15] UK High-end Computing (UKHEC). See http://www.ukhec.ac.uk

[16] BBC *'Shape of things to come'*, Wednesday, September 22, 1999. See http://news.bbc.co.uk/hi/english/sci/tech/newsid_452000/452554.stm

[17] Development of an Inter-Disciplinary Round Table for Emerging Computer Technologies (DIRECT): *'Whither HPC in Europe - A Strategic Review of High-Performance Computing from a European Prospective'*, by Dr Rob Baxter. See http://www.epcc.ed.ac.uk/direct/newsletter5/direct.html

[18] Ian Foster, Carl Kesselman, and Steven Tuecke, *'The Anatomy of the Grid: Enabling Scalable Virtual Organisations'*. See http://www.globus.org/research/papers/anatomy.pdf

[19] ENACTS Study 2: *'The HPC Technology Roadmap'* by NSC and CSCISM. See http://www.enacts.org/activities/roadmap

[20] DataGrid project. See http://www.eu-datagrid.org

[21] Eurogrid project. See http://www.eurogrid.org

[22] Damien project (Distributed Applications and Middleware for Industrial use of European Networks). See http://www.hlrs.de/organization/pds/projects/damien

[23] European Commission Information Society Technologies programme (IST), Cross-Programme Actions 9 (CPA 9): *'Grid test beds, deployment and technologies'*. See http://www.cordis.lu/ist/cpt/cpa9.htm

[24] Access Grid homepage. See http://www-fp.mcs.anl.gov/fl/accessgrid/default.htm

[25] The UK eScience programme: in November 2000 the Director General of Research Councils, Dr John Taylor, announced £98M funding for a new UK eScience programme. The allocations were £3M to the ESRC, £7M to the NERC, £8M each to the BBSRC and the MRC, £17M to EPSRC and £26M to PPARC. In addition, £5M was awarded to CLRC to 'Grid Enable' their experimental facilities and £9M was allocated towards the purchase of a new Teraflop scale HPC system. A sum of £15M was allocated to a Core eScience Programme, a cross-Council activity to develop and broker generic technology solutions and generic middleware to enable e-Science and form the basis for new commercial ebusiness software. The £15M funding from the OST for the core e Science Programme has been enhanced by an allocation of a further £20M from the CII Directorate of the DTI which will be matched by a further £15M from industry. See http://www.research-councils.ac.uk/escience

[26] EPSRC's e-Science Pilot Projects. See http://www.research-councils.ac.uk/escience/epsrcpilots.shtml

[27] Cactus problem solving environment. See http://www.cactuscode.org

[28] Supercomputing '98 HPC Challenge: 'Colliding black holes and neutron stars across the Atlantic Ocean'. See http://jean-luc.ncsa.uiuc.edu/Projects/SC98

[29] GGF Applications and Testbeds Research Group. See http://www.zib.de/ggf/apps

[30] The Astrophysics Simulation Collaboratory. See http://www.ascportal.org

[31] Virgo consortium. See http://www.mpa-garching.mpg.de/Virgo

[32] PACX-MPI: 'Extending MPI for Distributed Computing'. See http://www.hlrs.de/organization/pds/projects/pacx-mpi

[33] GGF Scheduling & Resource Management Area. See http://www.cs.nwu.edu/~jms/sched-wg

[34] Globus toolkit. See http://www.globus.org

[35] First Euroglobus workshop (Marina di Ugento, Lecce, Italy). See http://www.euroglobus.unile.it

[36] Platform Computing's Load Sharing Facility (LSF). At Berkley, Songnian Zhou developed Load Sharing Facility (LSF) software, which determines which computers and servers are needed for specific technical computing jobs and then manages application resources across an entire network. After building a prototype that was quickly adopted by Northern Telecom, Zhou and two partners launched Platform Computing in 1992. See http://www.platform.com

[37] Condor, high-throughput computing. The Condor project started in 1988, building on the results of the Remote-Unix (RU) project and as a continuation of the work in the area of Distribute Resource Management (DRM) by a group directed by Professor M. Livny (University of Wisconsin). See http://www.cs.wisc.edu/condor

[38] Ian Foster & Carl Kesselman: *'The Grid: Blueprint for a New Infrastructure'*, Morgan Kaufmann Eds (July 1998). See http://www.mkp.com/books_catalog/catalog.asp?ISBN=1-55860-475-8

[39] UK National High-Performance Computing Initiative, supported by the UK Engineering and Physical Sciences Research Council (EPSRC). See http://www.epsrc.ac.uk/documents/hpc/hpc_national_programme/hpci/hpci_main.htm

[40] UNICORE portal at CSAR.

[41] CCLRC data portal portals. See http://www.dl.ac.uk/TCSC/UKHEC/GridWG/portals.html; ERCIM News, No. 45 - April 2001. See http://www.ercim.org/publication/Ercim_News/enw45/ashby.html

[42] EPCC's ePortal. See http://www.epcc.ed.ac.uk/grid/eportal

[43] Gaussian portal. See http://lexus.physics.indiana.edu/griphyn/grappa; and Grid Access Portal for Physics Applications (GRAPPA)'s Science Portal Project. See http://www.extreme.indiana.edu/an

[44] Portable, Extensible Toolkit for Scientific Computation (PETSc). See http://www-fp.mcs.anl.gov/petsc

[45] Collaborative Computational Projects (CCP). See http://www.cse.clrc.ac.uk/ListActivities/CLASS=5;CLASSTYPE=21;

[46] Computer Physics Communications program library. See http://www.cpc.cs.qub.ac.uk/cpc

[47] DisCo Grid archive. See http://esc.dl.ac.uk/DisCo; and UK Grid Support Centre Grid Starter Kit: GSC/GSK. See http://esc.dl.ac.uk/StarterKit

[48] OCCAM data selector. See http://www.soc.soton.ac.uk/JRD/OCCAM/welcome.html

[49] GridPP. See http://www.gridpp.ac.uk; AstroGrid. See http://www.astrogrid.ac.uk

[50] CSAR PI on-line system. See http://www.csar.cfs.ac.uk/admin/PI_guide.shtml

[51] GGF Distributed Accounting Working. See http://www.gridforum.org/5_ARCH/ACCT.htm; See also forthcoming document on 'Grid Credits' by B. Rosenthal

[52] GGF Grid Security Infrastructure (GSI). See http://www.gridforum.org/2_SEC/GSI.htm; and Grid Certificate Policy (GCP) Working Groups. See http://www.gridforum.org/2_SEC/GCP.htm

[53] UK GSC. See http://www.nesc.ac.uk/esi/talks/RobAlan+SteveBoothPresentation/GSC_22-10-01.pdf

[54] CSAR resource tokens and trading pool. See http://www.csar.cfs.ac.uk/admin/rtokens.shtml

[55] GRISK project. See http://www.ii.uib.no/grisk

[56] INFN Grid. See http://server11.infn.it/globus

[57] ENEA Grid. See http://www.eu-datagrid.org/DGW3/sponsors.htm

[58] GGF3, *'Industry and Research Forum: How to involve more Industry in grid developments and early exploitation'*.

[59] APE SIMD machines. See http://chimera.roma1.infn.it/ape.html

[60] GRAPE project. See http://grape.c.u-tokyo.ac.jp/~makino/grape6.html

[61] GenoGrid: project funded by the French ministry of research *("Action Concertée Incitative GRID: Globalisation des Ressources Informatiques et des Données")*

[62] Virgo consortium's production code *'hydra'*. See http://www.epcc.ed.ac.uk/t3e/virgo

[63] GGF Grid Computing Environments (GCE). See http://www.computingportals.org

[64] Java numerical libraries (JNL) from Visual Numerics. See http://www.vni.com/products/wpd/jnl

[65] J.-C. Desplat, I. Pagonabarraga, and P. Bladon, *'LUDWIG: A parallel Lattice-Boltzmann code for complex fluids'*, Computer Physics Communications **134** (3), pp273-290 (2001)

[66] EPCC Summer Scholarship Programme 2001: *'Portable Lattice-Boltzmann in Java'* by Rubén Jesús García Hernández. See http://www.epcc.ed.ac.uk/ssp/2001/reports/ruben.pdf

[67] The Java Grande Forum. See http://www.javagrande.org

[68] Java Grande Forum Benchmarking Group. See http://www.epcc.ed.ac.uk/javagrande

[69] European Commission: Improving Human Potential, Enhancing Access to Research Infrastructures scheme. See http://www.cordis.lu/improving/infrastructure/home.htm; see also the TRACS programme at EPCC (http://www.epcc.ed.ac.uk/tracs), the MINOS programme at CINECA (http://minos.cineca.it/presentation.html), the ACCESS programme at CEPBA (http://www.cepba.upc.es/mobility.htm) and the visitor programme at UiB (http://www.fi.uib.no/~bcpl)

[70] M. Parsons, *"Workshop Report: Expanding the Grid reach in Europe"*, Workshop organised and hosted by the European Commission, 23/03/01, Brussels. See http://www.cordis.lu/ist/rn/grid_ws.htm

[71] The Quality of Service Forum contains a number of white papers on QoS and QoS enabling technology. See for instance *'The need for QoS'*, *'Introduction to QoS policies'*, *'QoS Protocols and Architectures'*, etc. See http://www.qosforum.com/tech_resources.htm

[72] The Internet Engineering Task Force (IETF). See http://www.ietf.org

[73] IETF Working Group: Differentiated Services (diffserv). See http://www.ietf.org/html.charters/diffserv-charter.html

[74] IETF Working Group: Integrated Services (intserv). See http://www.ietf.org/html.charters/intserv-charter.html

[75] Intersim-Diffserv project. See http://www.epcc.ed.ac.uk/intersim/diffserv

[76] Virtual Private Network (VPN) FAQ. See http://kubarb.phsx.ukans.edu/~tbird/vpn/FAQ.html

[77] Virtual Private Networks Consortium (VPNC). See http://www.vpnc.org

[78] UK Grid Support Centre. See http://www.grid-support.ac.uk

[79] Global Grid Forum Grid FTP Working Group and Data Replication Research Group. See http://www.zib.de/ggf/data

[80] The Economy Grid (EcoGrid) project: Economics Paradigm for *"Resource Management and Scheduling for Service-Oriented P2P/Grid Computing"*, by Rajkumar Buyya. See http://www.csse.monash.edu.au/~rajkumar/ecogrid

[81] Cactus Visualisation Tools. See http://www.cactuscode.org/VizTools.html

[82] MPI on-line. See http://www.epcc.ed.ac.uk/epcc-tec/MPIonLine

[83] Global Grid Forum Grid Information Services research area (GIS). See http://www-unix.mcs.anl.gov/gridforum/gis

[84] Global Grid Forum Grid Security area. See http://www.gridforum.org/ security

[85] SUN GridEngine. See http://www.sun.com/gridware

[86] NASA Information Power Grid (IPG). See http://www.ipg.nasa.gov

[87] Federal Computer Week: *'Plug and play: Power computing grids'*, 06/11/2000. See http://www.fcw.com/fcw/articles/2000/1106/tec-grid-11-06-00.asp

[88] GGF Advanced Programming Models (APM) Research Group. See
http://www.eece.unm.edu/~apm

[89] GGF Working Groups and Research Groups. See http://www.gridforum.org/L_WG/wg.htm

[90] Daresbury laboratory. See http://www.dl.ac.uk

[91] S.G. Parker and C.R. Johnson. *'SCIRun: A Scientific Programming Environment for Computational Steering'* in Supercomputing '95, IEEE Computer Society, (1995). See http://www.supercomp.org/sc95/proceedings/499_SPAR/SC95.HTM; J.A. Kohl, P. M. Papadopoulos. *'A Library for Visualization and Steering of Distributed Simulations Using PVM and AVS'* in High Performance Computing Symposium '95, Montreal, Canada (1995). See http://www.epm.ornl.gov/~phil/pvmavs/pvmavs.html; David M. Beazley Peter S. Lomdahl. *'Lightweight Computational Steering of Very Large Scale Molecular Dynamics Simulations'* in Proceedings of Supercomputing '96. See http://www.supercomp.org/sc96/proceedings/SC96PROC/BEAZLEY/INDEX.HTM; Balint Joo. *'DIVE: Distributed Interface to Visualisation Environment'*. See http://www.epcc.ed.ac.uk/ssp/1996/ss9612.html

[92] Digital Imaging and Communications in Medicine (DICOM). See http://medical.nema.org/dicom.html

[93] PDB: structural genomics. See http://www.rcsb.org/pdb/strucgen.html

[94] Kirstin Kleese, *'Data Management in Climate Research: from Acquisition to Visualisation'*. See http://www.epcc.ed.ac.uk/DIRECT/kkleese.html

[95] The Hierarchical Data Format (HDF). See http://hdf.ncsa.uiuc.edu

[96] GriPhyN: the Grid Physics Network. See http://www.griphyn.org

[97] DataGrid WP9: *'Earth Observation Science Application'*. See http://tempest.esrin.esa.it/~datagrid

[98] CCP5's DL_POLY. See http://www.dl.ac.uk/TCSC/Software/DL_POLY/main.html

[99] Canada Foundation for Innovation (CFI). See http://www.innovation.ca

[100] UK *'Joint Research Equipment Initiative'* (JREI) funding scheme.

[101] HPCN-TTN Network: *'European Commission brings Supercomputing to Small Companies (SMEs)'*. See http://www.hpcn-ttn.org

[102] The Economist, *'Computing power on tap'*, Jun 21st 2001. See http://www.economist.com

[103] Mario Antonioletti. *'Helping the trains to run on time'*. EPCC News **44**, November 2001. See also http://www.jardine.co.uk

[104] Data Intensive Distributed Computing Research Group (DIDC). See http://www-didc.lbl.gov

[105] DIRECT 'Data inter-operability group'. See http://www.epcc.ed.ac.uk/DIRECT/dio.html

[106] SDSC's Biology workbench. See http://workbench.sdsc.edu

[107] NERSC annual report: NetCDF. See http://www.nersc.gov/resear ch/annrep00/04RD.html

[108] The University of Lecce's *'Grid Resource Broker' (GRB).* See http://sara.unile.it/grb

[109] GGF Working Group: 'Grid User Services'. See http://www.gridforum.org/7_APM/GUS.htm

[110] Towns, J. Ferguson, D. Fredrick, G. Myers. *'Grid User Services Best Practice'*. Grid Forum Working Draft, Grid User Services Working Group, J (2001). See http://dast.nlanr.net/GridForum/UserServ-WG/Documents/GridUserServicesBestPractices-Final_Draft.htm

[111] Metacomputing applications, as shown in the paper by T. Dramlitsch, G. Allen et al. (*'Supporting Efficient Execution in Heterogeneous Distributed Computing Environments with Cactus and Globus'*, Proceedings of Supercomputing 2001, Gordon Bell Prize) can even achieve a high performance with a network of lower bandwidth, such as 24Mbps. The efficiency achieved using such a slow connection (based however on the OC12 network) for the tightly coupled Cactus simulation was up to 88%.

[112] The Ten-155 Network by Dante. See http://www.dante.net/ten-155.

[113] The EU DataTAG Project: 'Research and Technological Development for a Transatlantic Grid'. See http://www.datatag.org

Appendix B ENACTS Questionnaire

ENACTS User Grid Questionnaire

Thank you for having agreed to take part in this ENACTS grid user questionnaire. We hope that by your contribution we may be able to help and direct the development of grid environments. The questionnaire is divided into eight parts. No question is compulsory and you have the right to remain anonymous if you so chose. We would be grateful though if you could answer as many questions as possible though. In all instances you should try to answer the questions on behalf of the group you work with as opposed to yourself. If you supply your email address you will be emailed back a copy of your response. All responses will be treated as anonymous unless we explicitly ask otherwise. If you have any questions regarding any part of the questionnaire then please feel free to get in touch with us at enacts@epcc.ed.ac.uk. As a sign of our gratitude you will be entered into a free prize draw for a bottle of whisky - you may select the type of whisky at the end of the questionnaire.

A PDF version of the questionnaire is available here in case you want to preview/print the questionnaire before answering. Please do not use this to fill the questionnaire though.

Thank you for your time and contribution.

The ENACTS Team.

Warning: pressing the back button on your browser will result in loss of data if you use the submit buttons at the bottom of each page. Using the forward button on the browser should be ok. If you would like to modify, add or get a copy of your answers (if you forgot to add your email address) then email us at enacts@epcc.ed.ac.uk and we will do this for you.

Start: user details

In this section we would like to find out about you and the group you work with. If youwouldlike to get a copy of your answers to this survey then you must supply your email address.

Start – Awareness – User Profile – Application Profile – Infrastructure – Security & Services – Future Needs – End

1. Name: □

2. Email: □

3. Position: □ ▾

   if other then please specify: □

4. Department: □

5. Institution: □

6. Please specify the country where you are working: □ ▾

7. What is the composition of your group:

   ○ Only people from my department.

   ○ Only people from my institution.

   ○ People from several institutions but within my home country.

   ○ International collaborative group but within Europe.

   ○ International including transatlantic collaborative groups.

8. Approximately how many researchers work in your group: □ ▾

9. Could you give an indication of the level of collaboration within your group:

   ○ There is no overall coordination.

   ○ Coordinated performed by one person.

   ○ Independently coordinated subgroups within the group.

   Other, please specify: □

10. Can we get in touch with you to follow up any of your comments in this questionnaire?

    ● Yes

    ○ No

Go the Awareness Section

If you have any questions please email: enacts@epcc.ed.ac.uk. *$Revision: 1.28 $.*

Awareness

In this section we would like to find out how much you know about the grid and present you with some grid models, ask you which you find most attractive and try to determine what you would like to get out of grid environments.

Start – Awareness – User Profile – Application Profile – Infrastructure – Security & Services – Future Needs – End

1. Before looking at the proposed grid models below could you say whether you have heard about *grid technology*?

   ○ Yes, heard about and used

   ○ Yes, heard about but not used

   ○ No

   If you answered no to this question you can find a little background about the grid here.

2. For the purposes of this questionnaire we would like to propose the grid models outlined below. After each description please indicate how useful the given model would be to your group:

   Basic portal computing (workbenches)

   This model is suitable for groups using a set of *standard* (*i.e.* stable in time) executables. Examples of this approach are already in use (*e.g.* CFDNet) or are currently being developed (*e.g.* the EPCC MHD portal).

   Level of usefulness: ▾

   Advanced portal computing

   An evolution of the previous model which includes support for remote compilation and user options (*e.g.*, users are given control of which features will be built into the binary, using which code revision, *etc.*). This model is particularly relevant to groups with codes requiring compile-time customisation (*e.g. pick-and-mix package* such as `DL_POLY`, custom codes with user-selectable options, *etc.*).

   Level of usefulness: ▾

   Small-scale Grids (load sharing systems), *e.g.*, Condor, LSF, *etc*

High throughput computing environments can be used to manage *spare cycles* or evenly distribute the load across large numbers of distributed, privately owned workstations. More details are available here.

Level of usefulness: ▾

The toolkit approach, *e.g.* Globus, Legion and Unicore, *etc*.

These are much more sophisticated approaches that allow you to create *meta batch* systems, providing a common uniform access point to a variety of resources. Some prominent examples adopting this mode of operation are described here.

Level of usefulness: ▾

Internet computing (capacity computing) and peer-to-peer computing

Peer-to-peer computing (as implemented, for example, in the Napster, Gnutella, and Freenet file sharing systems) and Internet computing (as implemented, for example by the SETI@home, Parabon, and Entropia systems) is an example of the more general ("beyond client-server") sharing modalities and computational structures that could be exploited to process large amounts of distributed data or to exploit spare computational capacity provided by voluntarily by users and as such, have much in common with Grid technologies.

Level of usefulness: ▾

3. Do these models correspond to what you had previously heard about grid technology?

○ Yes

○ No

If not could you say where the differences lie:

4. Could your research group benefit from a grid network?

   ○ No

   ○ Yes

   If yes could you give some reasons?

5. What type of grid network would best serve your group?

   ○ One exclusively restricted to my group.

   ○ One that only worked at a local departmental level.

○ One that only worked at a local institutional level.

○ One that only served researchers doing similar type of work.

○ One that only worked at a national level.

○ No geographical restrictions.

Could you comment on why you have made the above choice:

6. Will you require *interactive* access to any of the target machine you would use within a grid environment?

○ No

○ Do not know

○ Yes

If yes could you please briefly say why:

7. Have you any experience with any of the following *grid enabling technologies*?

   □ Condor

   □ LSF

   □ Codine

   □ Nimrod

   □ Globus

   □ Unicore

   □ Legion

   Other, please specify

   If any could you comment on your experiences with them, *e.g.* ease of use, effectiveness

... *etc*.

8. Could you judge the importance of the following factors that you think are important for you to work within a grid environment?

- □ Access to faster CPUs.
- □ Access to greater total memory.
- □ Access to more processors (parallel machine).
- □ More memory per processors on a distributed memory machine.
- □ Production run environment.
- □ Testing scalability on more processors.
- □ Access to longer run times.
- □ Access to larger storage devices.
- □ Access to fast Input/Output.
- □ Better throughput.
- □ Better turn around time.
- □ Ease of use.
- □ Security of your source code/data files.
- □ Access to the best machine for the job.
- □ Access to resources not locally available.

Other important points not mentioned above, please specify: □

9. If you are seeking access to resources not locally available would this be for:

   ○ Use only once (to evaluate or test).

   ○ To do a performance evaluation.

   ○ To test portability

   Other, please specify: □

   What resource you would be looking for

10. Would access to a grid encourage you to tackle larger or more complex problems?

    ○ No

    ○ Do not know

    ○ Yes

    If yes could you give some examples of wha t you might wish to try:

11. What kind of computing architectures would you like to be available to you?

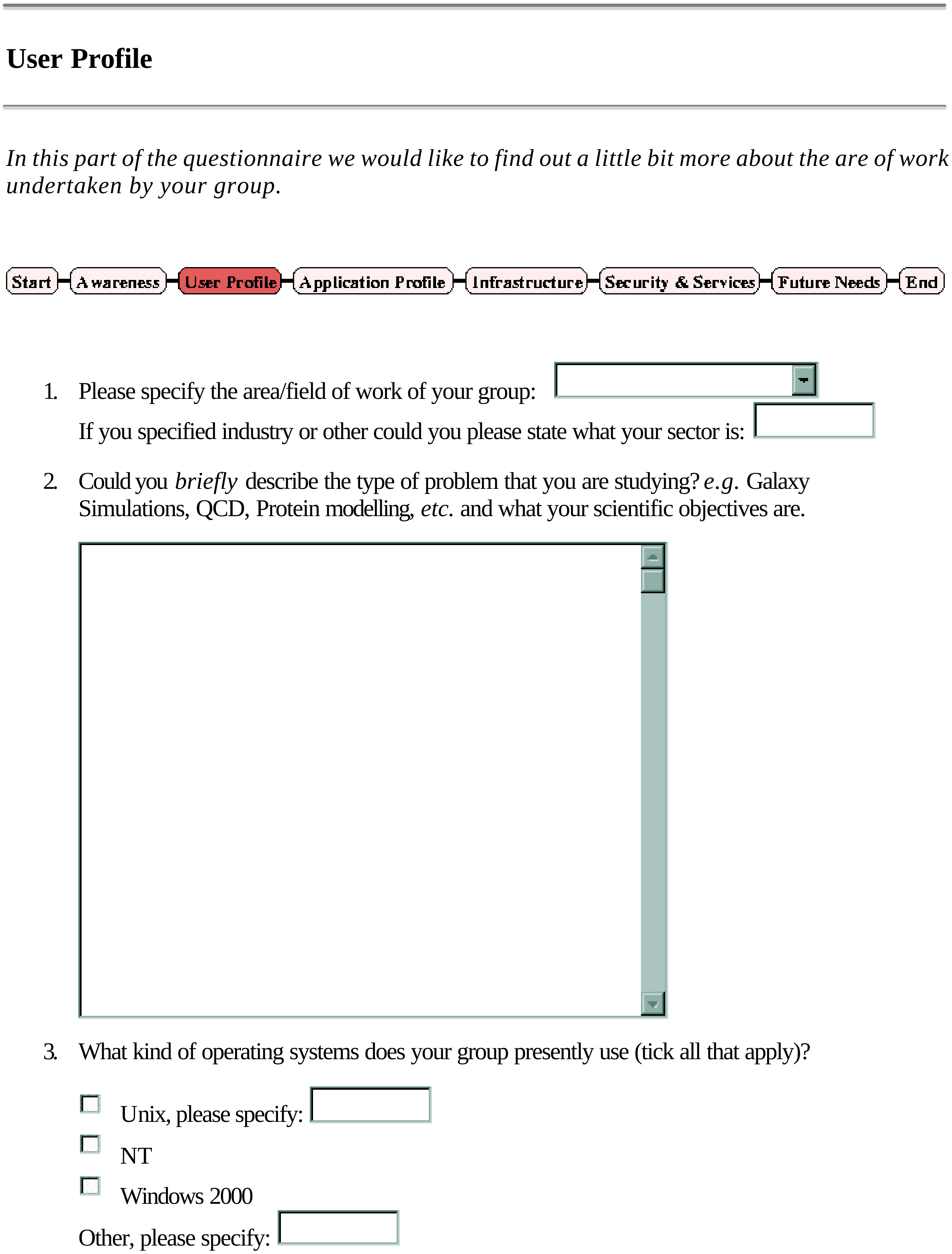
   - □ Fast single CPU.
   - □ Multiple processors, shared memory.
   - □ Multiple processors, distributed memory.
   - □ Vector processing.
   - □ SIMD
   - □ System with large memory or large secondary storage

   Specific machine type/brand, please specify: ▭

   Other architecture, please specify: ▭

[Go to the User Profile]

If you have any questions please email: enacts@epcc.ed.ac.uk. *$Revision: 1.28 $.*



User Profile

In this part of the questionnaire we would like to find out a little bit more about the are of work undertaken by your group.

1. Please specify the area/field of work of your group:

   If you specified industry or other could you please state what your sector is:

2. Could you *briefly* describe the type of problem that you are studying? *e.g.* Galaxy Simulations, QCD, Protein modelling, *etc.* and what your scientific objectives are.

3. What kind of operating systems does your group presently use (tick all that apply)?

   □ Unix, please specify:

   □ NT

   □ Windows 2000

   Other, please specify:

4. Looking at the following definitions could you indicate under which of these models your group operates (tick all that apply), and roughly the percentage of your group that operates under this model:

□ Black box user:

You run purely pre-compiled applications. You specify input parameters and extract results. You do not modify source code and/or do not have access to it.

Approximate percentage of group that employ this model: ▾

□ User:

You have access to the source code of your application and make occasional modification to the source and/or re-compile the code or have to re-compile to use a different set of modules. You do not develop the code.

Approximate percentage of group that employ this model: ▾

□ User Developer

You are developing code as well as using it. Performing new runs often involves re-compilation or modification of the original source.

Approximate percentage of group that employ this model: ▾

Got to the Application Profile Section

If you have any questions please email enacts@epcc.ed.ac.uk. *$Revision: 1.28 $.*

Application Profile

In this part of the questionnaire we would like to know something about the applications that your group runs. This may be important as you will have to specify your job requirements in some manner. Do not worry if you cannot answer any of these questions as this is useful to know too.

Start – Awareness – User Profile – **Application Profile** – Infrastructure – Security & Services – Future Needs – End

1. What language is your application written in? (if more than one please select accordingly).

   □ Do not know
   □ C
   □ C++
   □ Java
   □ Fortran 77
   □ Fortran 90
   □ Fortran 95
   □ HPF
   □ OpenMP
   □ MPI
   □ PVM
   □ shmem

   Other, please specify: ▭

2. Please give an indication of the computational bottlenecks in your current research codes by choosing amongst: *Critical* means you can not do your work because of this, *Major* means you can do the work but quality is severely affected, *Minor* means you can work around it but would like it to be better and it is *Not an issue*?

   [▼] CPU power
   [▼] CPU to main memory performance

▾ Disk I/O performance

▾ Memory capacity

▾ Data storage capacity

▾ Network performance

Other, please specify and include a comment on the severity: □

3. How much total <u>input data</u> do you require to run a typical job: ▾

   and what format is this data usually in (tick all that apply):

   □ plain text/ASCII.

   □ binary files.

   □ propriatory formats used by package.

   □ machine independent, *e.g.* HDF, XDR.

   Other, please specify: □

4. How much total <u>output data</u> does your typical job produce: ▾

   and what format is this data usually in (tick all that apply):

   □ plain text/ASCII.

   □ binary files.

   □ propriatory formats used by package.

   □ machine independent, *e.g.* HDF, XDR.

   Other, please specify: □

5. Can you specify your typical job requirements?

   - Memory used by the application:
     ○ Yes ○ No ○ Do not know.
     If yes roughly how much memory: □ Mbs.
     If you use a distributed memory machine put *memory required per processor (number of processors required) e.g. 250(64)* in the text box.

   - Typical run time (approx wall clock time):
     ○ Yes ○ No ○ Do not know.
     If yes roughly how many □ minutes ▾ on a □ (type of machine this value corresponds to) on □ (number of processors).

- Temporary disk storage (files used temporarily during a run but not kept after execution finishes):

  ○ Yes ○ No ○ Do not know.

  If yes roughly how much: ______ Mbs.

- How often do you have to recompile your application: ______ ▾ and typically how many times will you run your application before you have to recompile: ______ ▾

6. If you use an application (or applications) that you did not write what is it (or are they)? *e.g.* DLPoly, Gaussian Amber, *etc* please outline below. If your group wrote your own code then just write *own code* below.

7. Would you be able to run your application through a web interface?

   ○ Yes

   ○ Do not know

   ○ No

   if no could you specify reasons why this would not be desirable/possible?

8. To make your application run and/or compile does it require access to special resources, *e.g.* OpenMP or HPF compilers, compiler support for standard (ISO,ANSI,...) extensions, commercial/free numerical libraries, packages *etc*?

   ○ Yes, please describe

   ○ No.

   ○ Do not know.

9. How much effort would you be prepared to spend making your application more portable (*e.g.* not referencing input/output files via absolute paths, removing dependencies on libraries not universally available, removing non-standard compiler extensions or reliance on tools such as `gmake`, *etc*): ▾
   What changes do you think you would have to make for your code?

10. Java is often referred to as *a platform independent language*. Would you consider re-writing your application to use Java?

    □ Yes

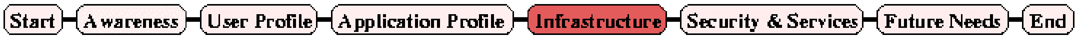
    □ No

    Could you give a couple of reason for making this choice?

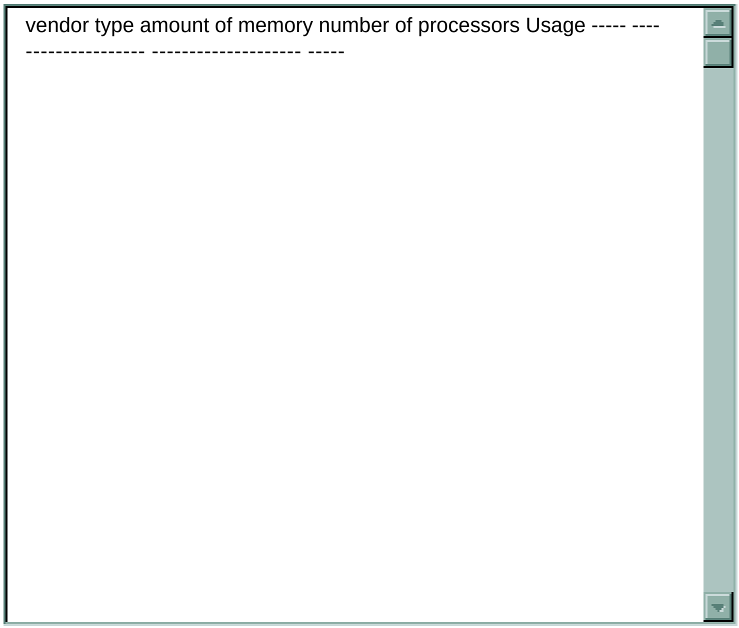
Go to the Infrastructure Section

Infrastructure

In this part of the questionnaire we would like to be able to determine what resources you already have access to.

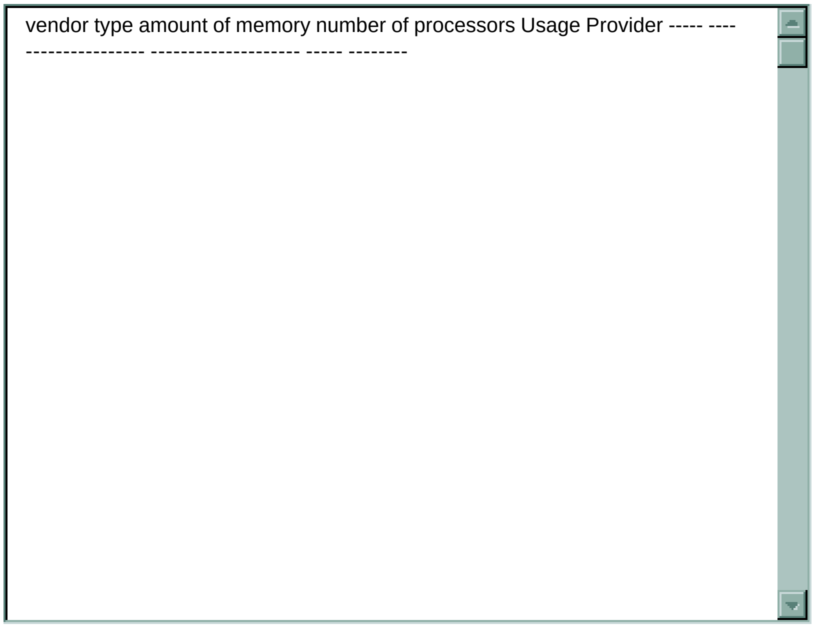


1. What systems do you have access to and use within your home institution and approximately how much time do you use for each of these (processor hours over a year)?



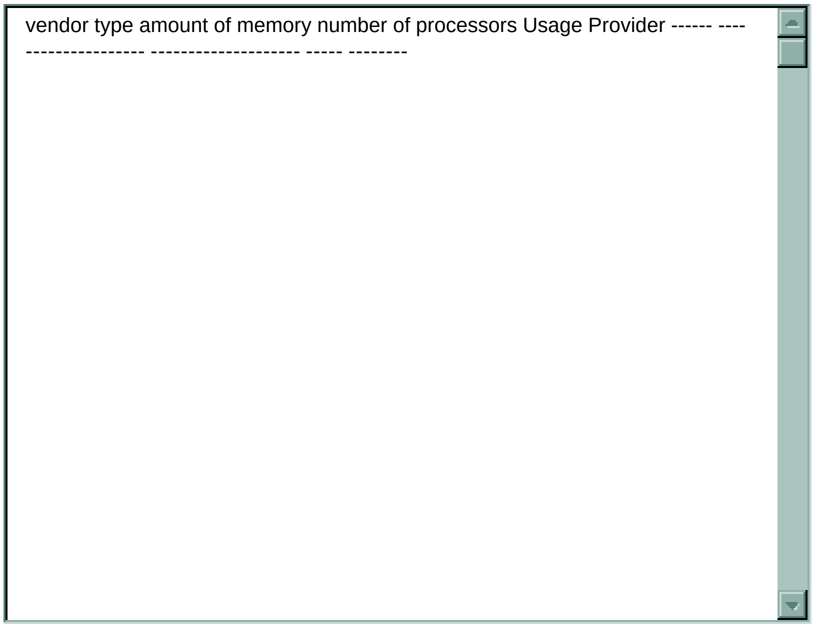
vendor type amount of memory number of processors Usage ----- ----
--------------- -------------------- -----

2. What kind of resources do you have access to <u>outside your own institution but within your home country</u> and approximately how much time do you use for each of these (processor hours over a year) and who provides this resource?



vendor type amount of memory number of processors Usage Provider ----- ----
---------------- -------------------- ----- --------

3. What kind of access do you have access to <u>outside your own home country</u> and approximately how much time do you use for each of these (in terms of processor hours over a year) and who provides this resource?



vendor type amount of memory number of processors Usage Provider ------ ----
---------------- -------------------- ----- --------

4. What kind of access (bandwidth) do you have?

- within your institution ▭ Mbits/s
  or do not know □

- from your machine to the outside world (to the backbone but not the backbone itself) □ Mbits/s

  or do not know □

Go to the Security & Services Section

If you have any questions please email: enacts@epcc.ed.ac.uk. *$Revision: 1.28 $.*

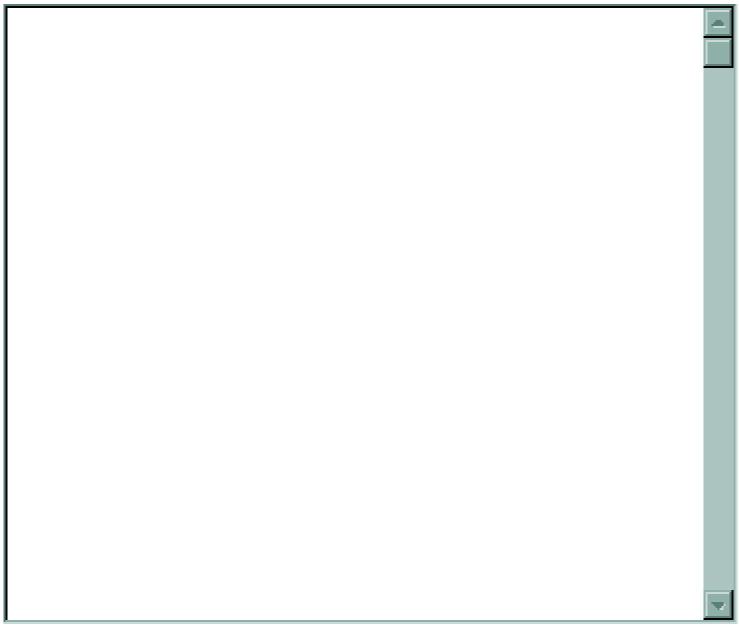
Security and Services

Start - Awareness - User Profile - Application Profile - Infrastructure - **Security & Services** - Future Needs - End

1. What added services from a grid service provider do you think are important to determine the quality of service?

   ▾ Training (in a class room environment).
   ▾ Training (distance learning).
   ▾ Support (answer technical queries or resolve difficulties).
   ▾ Remote visualisation tools and support to analyse data.
   ▾ Long term secure data storage.
   ▾ Ease of use.
   ▾ Guaranteed turn around time.
   ▾ Security of code, input and output data.
   ▾ Availability and reliability

   Is there any other service that you think would be important that is not covered above:



2. Are your systems installed behind a firewall?

○ Yes
○ No
○ Do not know

3. Would you like to have a control over the level of permissions you grant to other Grid components, such as scheduling brokers, bandwidth reservation brokers etc.

○ Yes
○ Do not know
○ No, I want security issues like proxy credentials to be transparent to me.

What motivates you to answer this way:

Go to the Future Needs Section

If you have any questions please email: enacts@epcc.ed.ac.uk. *$Revision: 1.28 $.*

Future Needs

In this part of the questionnaire we would like to know a little bit about your future needs.

Start – Awareness – User Profile – Application Profile – Infrastructure – Security & Services – **Future Needs** – End

1. What new problems would you address if the current computational constraints were eliminated? Describe in what way they differ from the problems you currently work on. Are they new or scaled up versions of current problems?

2. What is your opinion on how the set of application codes that you use today will have changed, 5 years from now and 10 years from now? To what fraction do you estimate that the codes will be (try to get the columns to add up roughly to 100% in so far as this is possible):

	In 5 years	**In 10 years**
The same or new versions of third party codes as today	▾	▾

New third party codes		
The same in house codes as today		
Slightly rewritten or enhanced versions of current in house codes		
New codes written in house from scratch		

3. Moore's law stip ulates a doubling of transistors on microprocessors every 18 months. Processor peak performance, memory density and disk storage capacity are all on a very similar curve as Moore's law. This would give an increase by a factor of about 10 five years from now and a factor of 100 ten years from now. By simply extrapolating performance and capacity into the future and assuming that financial funding will be on the same level as today we would like to get your view on how this potential performance increase would compare to you future computational needs.

	5 years from now		10 years from now
10xCPU integer performance :		100 x CPU integer performance :	
10 x CPU floating point performance:		100 x CPU floating point performance :	
10 x Primary memory capacity :		100 x Primary memory capacity :	
10 x Short term storage (during a calculation):		100 x Short term storage (during a calculation) :	
10 x Long term storage :		100 x Long term storage :	

4. The bandwidth and latency performance seems to grow on a slower-than-Moore's law curve thus not keeping pace with the processor and memory performance development. In a future perspective, what is your opinion as to how this will affect your application's performance:

Memory bandwidth and latency:	
Disk bandwidth and latency:	
Network bandwidth and latency:	

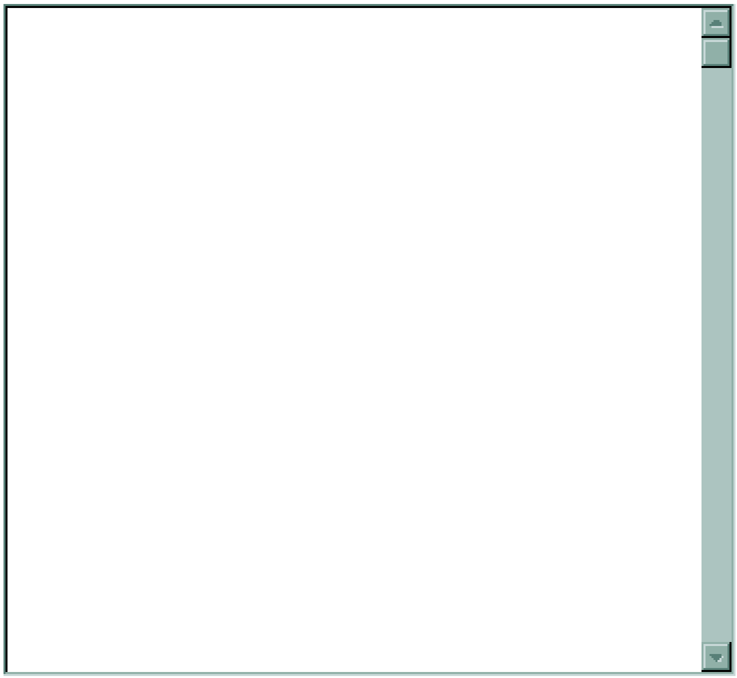
5. Single or parallel processing. Please indicate your view on the importance for you on future needs for the following:

	In 5 years	In 10 years
Single processor performance	▾	▾
Access to moderately parallel systems, less than 128 processors	▾	▾
Access to highly parallel systems, more than 128 processors	▾	▾

6. What kind of parallel programming models do you think you will be interested in the future, 5-10 years from now. Could you indicate how important you think they will be for your future work?:

	In 5 years	In 10 years
OpenMP	▾	▾
Message Passing	▾	▾
SHMEM	▾	▾
Co-array FORTRAN/UPC	▾	▾
HPF	▾	▾

Before finishing are there any other issues, comments, remarks that you would like to be considered by us that you do not think have been addressed by the questionnaire. Please add this below



You may also now select the type of whisky that you would like if you are our lucky winner (if you do not like whisky then leave *Other* and we will get back in touch and get you something of equal value:

Go to the End

If you have any questions please email: enacts@epcc.ed.ac.uk. *$Revision: 1.28 $.*

Thank You

Start – Awareness – User Profile – Application Profile – Infrastructure – Security & Services – Future Needs – End

Thank you very much for having filled in this questionnaire. The results will be useful to provide some user input into future grid developments. If you supplied your email address then you will be emailed back a copy of your answers. If you do not get a copy of your answers it may be that the email address you supplied may not be correct. If you email enacts@epcc.ed.ac.uk we will send you a copy of your replies. If you did not provide your email address but would like a copy then if you email the same address we will send you a copy. You will be informed when the results of the survey come out.

The ENACTS Team.

If you have any questions please email: enacts@epcc.ed.ac.uk. *$Revision: 1.28 $.*

Appendix C Description of various Grid models

This section contains a transcript of the summary Grid models presented to the respondents in Section "Awareness" of the on-line questionnaire.

Before looking at the proposed grid models below could you say whether you have heard about *Grid technology?*

- Yes, heard about and used
- Yes, heard about but not used
- No

If you answered no to this question you can find a little background about 'the Grid':

C.1 What is the Grid?

The term "the Grid" was coined in the mid-1990s to denote a proposed distributed computing infrastructure for advanced science and engineering. Considerable progress has since been made on the construction of such an infrastructure but the term "Grid" has also been changing its meaning a bit, at least in popular perception, to embrace everything from advanced networking to artificial intelligence. The question is whether there are needs for the Grid technologies and, if so, what are these needs? What is the nature of these technologies, and what is their domain of applicability?

One of the simplest ways of implementing a Grid is to us a *Grid portal*- a customisable, personalised web interface for harnessing Grid services and resources. The Grid Portal Development Kit (GPDK) (http://dast.nlanr.net/Features/GridPortal) provides modular, reusable components for accessing common Grid services as well as providing a sample template portal that can be easily modified or extended to support additional services and customised problem solving environments. One of the core portal requirements is to provide users with a common gateway to Grid resources accessible from any system including their home machine with their certificate and private key or a publicly available terminal with a secure web browser. We can also speak of the application specific (grid) portals.

Some of the other current ways of implementing a grid are described in the parent document.

For the purposes of this questionnaire we would like to propose the grid models outlined below. After each description please indicate how useful the given model would be to your group:

C.2 Basic portal computing (workbenches)

This model is suitable for groups using a set of *standard* (*i.e.*, stable in time) executables. Examples of this approach are already in use (*e.g.* CFDNet, see http://www.cfdnet.com) or are currently being developed (*e.g.* the EPCC MHD portal).

C.3 Advanced portals

An evolution of the previous model which includes support for remote compilation and user options (*e.g.*, users are given control of which features will be built into the binary, using which code revision, etc.) This model is particularly relevant to groups with codes requiring compile-time customisation (*e.g. pick-and-mix package* such as DL_POLY, custom codes with user-selectable options, etc.)

C.4 Small-scale Grids

These *high throughput computing environments* can be used to manage *spare cycles* or evenly distribute the load across large numbers of distributed, privately owned workstations. Their development has been motivated by an ever increasing need to harness any spare capacity in such collections, *e.g.* workstations sitting in peoples offices will usually lie idle over night. Special attention is thus paid to the rights and sensitivities of the workstation owners. It is the owner that defines the conditions under which the workstation may be allocated to an external user. These environments usually provide a powerful and flexible suite of *resource management* services for sequential and parallel applications. It is even possible using remote system calls to preserve a large measure of the originating machine's environment on a remote execution machine, even if the originating and execution machines do not share a common file system and/or user ID scheme. Jobs that consist of a single process can be automatically checkpointed and migrated between workstations as needed to ensure eventual completion.

Checkpointing and job migration of parallel jobs is also supported by some of these environments too.

C.5 Toolkits

These are much more sophisticated approaches that allow you to create *meta-batch* systems, providing a common uniform access point to a variety of resources. Some prominent examples adopting this mode of operation are described underneath.

Globus provides a software infrastructure that enables applications to handle distributed, heterogeneous computing resources as a single virtual machine. The Globus project is a U.S. multi-institutional research effort that seeks to enable the construction of computational grids. A computational grid, in this context, is a hardware and software infrastructure that provides dependable, consistent, and pervasive access to high-end computational capabilities, despite the geographical distribution of both resources and users. A central element of the Globus system is the Globus Metacomputing Toolkit (GMT), which defines the basic services and capabilities required to construct a computational grid. The toolkit consists of a set of components that implement Initiative basic services, such as security, resource location, resource management, and communications.

Legion is an object-based metasystem developed at the University of Virginia. Legion provides the software infrastructure so that a system of heterogeneous, geographically distributed, high performance machines can interact seamlessly. Legion attempts to provide users, at their workstations, with a single, coherent, virtual machine. Legion is organised by classes and metaclasses (classes of classes).

Legion takes a different approach to provide a metacomputing environment: it encapsulates all its components as objects. The methodology used has all the normal advantages of an object-oriented approach, such as data abstraction, encapsulation, inheritance, and polymorphism. It can be argued that many aspects of this object-oriented approach potentially make it ideal for designing and implementing a complex environment such as a metacomputer. However, using an object-oriented methodology does not come without a raft of problems, many of these is tied-up with the need for Legion to interact with legacy applications and services. In addition, as Legion is written in Mentat Programming Language (MPL), it is necessary to “port” MPL onto each platform before Legion can be installed.

UNICORE (UNiform Interface to COmputer REsources) is a project funded by the German Ministry of Education and Research. A consortium of people from universities, national research laboratories, software industry, and computer vendors develops UNICORE. Initially, it was a two years project ending in December 1999 but there is a plan to retarget it and extend it for another two/three years. UNICORE main focus is in providing a uniform interface for job preparation and control that offers seamless and secure access to supercomputer resources. The idea behind UNICORE is to support the users by hiding the system and site-specific idiosyncrasies and by helping to develop distributed applications.

C.6 Internet computing (capacity computing) and peer-to-peer computing

Peer-to-peer computing (as implemented, for example, in the Napster (http://www.napster.com), Gnutella (http://www.gnutella.co.uk), and Freenet (http://freenet.sourceforge.net) file sharing systems) and Internet computing (as implemented, for example by SETI@home (http://setiathome.ssl.berkeley.edu), Parabon (http://www.parabon.com), and Entropia (http://www.entropia.com) systems) are examples of the more general ("beyond client-server") sharing modalities and computational structures that could be exploited to process large amounts of distributed data or to exploit spare computational capacity provided by voluntarily by users and as such, have much in common with Grid technologies.

Appendix D Questionnaire raw data

Total number of data sets: 85

Details of the respondents have been removed from this document to preserve their anonymity. However, each comment has been tagged so that all answers from a given respondent can be followed through the whole questionnaire. The tag is made of a number between square brackets ([n]), appended to each comment.

Note also that copies of these answers together with the Perl scripts used to post-process them is available for download from the ENACTS web site:

http://www.epcc.ed.ac.uk/enacts

One of these scripts allows to filter the data according to up to three different parameters (e.g., if one wishes to study separately the answers from German engineers).

Part 1 - Start

Details of all those that answered the Questionnaire

Name, status, E-mail address, department, university/company, country.

People that are ok about us doing a followup: 79

Numbers of Answers by Country

Czech Republic	4
Denmark	5
Finland	6
France	4
Germany	13
Greece	6
Ireland	3
Italy	4
Netherlands	2
Norway	5
Poland	3
Portugal	2
Slovakia	1
Spain	8
Sweden	7
United Kingdom	11
Total	84

Sizes of the Groups

1	2
2-10	53
11-50	25
51-100	2
>100	2
Total	84

Composition of the Group

Departmental	28
Institutional	13
National	12
International-European	13
International	18
Total	84

Level of collaboration in the Group

One person coordinating	37
No overall coordination	4
No response	4
Independently coordinated groups	39
Total	84

Part 2 - Awareness

Previous encounters with the Grid

Yes, heard about and used	17
Yes, heard about but not used	48
No response	1
No	18
Total	84

Usefulness of Basic Portal Computing

Not useful	22
Of some use	28
Useful	24
Very useful	10
Total	84

Usefulness of Advanced Portal Computing

Not useful	16
Of some use	24
Useful	28
Very useful	16
Total	84

Usefulness of small scale Grids

Not useful	16
Of some use	15
Useful	35
Very useful	18
Total	84

Usefulness of the toolkit approach

Not useful	14
Of some use	19
Useful	29
Very useful	22
Total	84

Usefulness of Internet Computing

Not useful	26
Of some use	23
Useful	23
Very useful	12
Total	84

Correspondence to what was previously heard about the grid

Yes	66
No	8
No Response	10

Tot	84

Comments:

I do not really have a good idea of what grid computing is. What I would like to see is a very simple user interface where I can write my programs in a convenient language (say C or ZPL for parallel programs) and then only have to type say

% remote Program.z

and then the system finds an appropriate computer to compile and run my program on (and it should find any local input-files I defined) and after it is finished I will find the resulting files in the directory I called the program from. Ideally this would also work for interactive programs.

There should then also be a simple utility to find the status of submitted applications:

% status remote

Program.z is running on 1024 processors on asci.lanl.gov

That is my vision of a no-hassle grid computing where the user does not have to worry about where the programs will be run on what kind of architecture. [2]

I knew the methods but in my area of science GRID is meaning totally something else... [8]

The models described above correspond, but we also want to be able to do

** very large scale metacomputing, distributing simulations across multiple supercomputers*

** make use of grid services, such as resource brokers, contract negotiators, and information services in portals, and importantly to allow applications themselves to be much more grid-aware, and able to dynamically exploit a changing grid environment*

** also would consider remote visualisation, steering and monitoring of applications, along with other collaborative techniques to be part of grid technology [22]*

I cannot really answer to that part of the questionnaire in a sound basis because of lack of enough knowledge to assess the answers (how do these possibilities can really help my work?). All seem interesting advances, but I am not sure if they will be worth to our research work [35]

Idea of Virtual laboratory or e-Science is not covered [50]

The concept "grid" is simply new to me, and therefore. I have nothing to compare with. [57]

Actually I can't answer to this question because I have never heard before about grid. [65]

So far I understood grid computing more as an initiative to couple high performance computing facilities across institutional structures to provide higher capacities. This provides a 'reliable' environment while I understand the 'grid technology' term used here more as attempts to utilise spare cycles on machines in a rapidly changing environment. [80]

Benefit from a grid network

Yes	77
No	3
No Response	4
Tot	84

Comments:

We are interested both in distributed high end simulation effort - i.e. generating, distributing, post-processing and visualising large theoretical datasets across institutional and national boundaries (i.e. the Virgo Consortium) and in coordinated and transparent access and manipulation of large globally distributed astronomical data archives (i.e. the Global Virtual Observatory). [1]

If something that I have described above would be available this would allow us to access high-end computers in a much simpler way and might eliminate bottlenecks in high-end computing facilities like CSAR.

There will be, of course, many issues about transferring data, compatibility etc., but I don't want to have to think about it. [2]

Easier access to large scale computing facilities increasing need computing power for making large-scale MHD simulations of solar related problem in 3D time depended situations [3]

We use it for research and we are using it to develop high-level middleware for grid applications. [4]

We are dealing with huge biomolecular systems which requires long computations (up to one month using 32 processors). [8]

My research group can benefit from an integration of workstations and supercomputers/high performance computers. We would also strongly benefit from an integration including various supercomputers/high performance computers. [9]

Enable users to submit jobs to one "single virtual machine" instead of having to handle a variety of different machines. [10]

One reason is the fact that we have large amounts of data stored at various sites - EPCC, Manchester and locally.

Another would be to get better computational throughput or solve bigger problems [12]

The basic problem in many our applications is the large amount of input/output data. If this can be solved, our applications could benefit large 'number crunching' possibilities. [13]

For example, It would be important in using graphical interfaces because we have only limited experience implementing such procedures. [14]

To increase our calculation CPU time [15]

With reservations. In computational simulation of viscous ship flows it is highly beneficial to distribute the workload on a network of processors and/or computers. However due to the large amount of data involved in such computations there is a risk that the bandwidth of the network becomes a bottleneck. [16]

Our group as well as the local computing center of the university runs a cluster of unix workstations/CONVEX SPP and the load sharing system is very useful to spread the jobs evenly.

We have also access to two supercomputing centers in Germany, and we would benefit from a more common/uniform environment. [18]

We are participating in a international experiments that are producing big amounts of data that must be analysed [19]

We are working on developing different modules on a very large system of air pollution models; we shall not be forced first to distribute them and then to use them. We shall be able to use a common set of input data (or even input data distributed on several sites). We shall be able to redirect output results to the place where they are to be visualized. [20]

Distributed access to resources. Stable, uniform sources. Distributed analysis and visualisation. [21]

We need a lot of CPU cycles, and are using codes which are platform independent, so we can make use of available machines anywhere. If there isn't enough memory on a single machine, we want to be able to easily form larger virtual machines, we also want easier access to computing resources. [22]

We use similar simulation software in our group [24]

Efficient use of computational resources [25]

Better utilization of computer resources in house, and access to resources not available in house. [26]

---

we share codes, data and machines [27]

Sharing of common models on multiple platforms. Greater use of varied HPC facilities. Reuse of many utilities. Shared data and data analysis tools. Information systems to enhance collaboration [28]

---

to perform extremely large computations and to take advantage of some unused capacities [29]

To offer the possibility of working with common tools (codes, packages) on a distributed computing resource (i.e. part of a "virtual" laboratory whose computing elements could be "delocalized" in a national/continental network). [31]

We develop a new system architecture that uses the grid approach itself and moreover is aimed at fitting into the grid seamlessly. [32]

New experiments at CERN's new hadron collider (LHC) have Petabyte data which must be accessed by many simultaneous users distributed over five continents. Seems to be the basic concept of GRID. [33]

To facilitate access to the emptiest, fastest machine available for massive computations [34]

The same as above [35]

Better performances and optimized working time [36]

We seek the possibility of using other resources available [37]

Giving service to the Scientific Community of our University (Physicists, etc.), Solving larger Combinatorial Optimization Problems, Easing the administration of our Systems (Advanced Portal Computing) [38]

We hope to find contacts and will be able to solve larger problems. [40]

---

load balancing, better resource utilisation in general. [41]

Possibility for computing larger problems. [42]

We could efficiently utilize our computational resources. [43]

I am only using LSF and this is very useful for my work but I do not have any other knowledge about this technology. [47]

Easier access to high-performance computing resources, independent of architecture and physical location. [49]

Shared access to high-throughput computing resources [50]

---

a testbed is needed [51]

Yes in the sense that we are one of the founding nodes of grid and are therefore interested and keen on any developments that promote and/or use grid computing under the different modalities outlined by you above, [52]

Data mining project PRODACTOOL needs computing power. [53]

A common code platform, built from modules and modified during compilation to accommodate individual needs would be beneficial for our group computing, since we are all using the same code system. [54]

More machines of different types. Cooperation with other remote researchers [55]

Have developed software for a embarrassingly parallel problem, and need all the cpu cycles we can get. [56]

We are already using some of the models described above. Batch processing on our cluster of workstations is partly driven by DQS, partly by so-called "farm tools", a suite of routines written by J. Heinen in Juelich which allow to access spare cycles, but at the same time give the user at the screen the right to "lock" the machine if it gets too heavily loaded. I am also wondering if PCs in other groups could be used by a technology similar to SETI@home, but have not been able yet to figure out how. In particular, I am not sure how to handle computer security concerns with such a system.

We are also doing lots of parallel computations on the big Cray T3E of the Max Planck Society in Garching. Most of our applications make good use of the its excellent communication hardware, but some have only little communication. These latter applications in essence waste the high technology, but on the other hand, it is the only machine available to us which has enough power. Access to a BIG (several 100) workstation cluster could solve that problem, and here I could imagine grid technology coming into play. [57]

BCPL hosts a wide variety of research projects in computational physics, particularly in reaction dynamics. For different tasks many of the above mentioned methods are adequate. [58]

Photonics simulation requires increasing computational resources, complexity and the ability to combine several different physical models. [61]

Investigate tools enabling I.T. group to validate grid technology for future use to CERFACS' researchers [63]

---

1. More computational resources.

2. Collaboration with other research groups around EU.

3. Individual benefits [65]

We are working on distributed systems based on the mobile agent paradigm. It could be interesting to investigate the applicability of the paradigm to a grid environment. [66]

Several people at different locations need to access data located at different servers and perform various types of manipulations / calculations [68]

We have about 10 Windows NT workstations, if we could systematically use these at night to solve either large problems in some form of parallelism or simply run the same program with different parameters on the different systems, then we could optimise our use of these machines and tackle some problems that take too much time when run on only one machine. Typical example: tests of irrigation canal control schemes for different canal *maintenance states, sensitivity analysis of such schemes to deviations between actual and predicted canal behaviour, etc. [69]*

It is easy and accelerates research process. [70]

Presently, we have access to very large HPC centres and facilities. So, the use of these tools is low priority, al lest at present. This may change in the future, if we are interested in clustering HPC resources at different centres [71]

We are doing research on grid software, so this is at the heart of our activities. [72]

Cost effective access to large number of CPUs and large memory [73]

Don't really know yet. If yes, it is because of better use of resources and possibility to get more and better calculations done. [74]

To have more CPU resources. To benefit of a seamless computing environment [75]

Some of our projects are very CPU intensive, and in that respect grid computing can be useful. [76]

Access to HPC otherwise impossible to use. The possibility to perform large simulations that require supercomputers. [77]

---

access to distributed (archived and real-time) data sources; access to computing capacity [78]

Research groups intends to build a computing network for Biocomputing purposes. [79]

There are two main directions of research in our department:

a) development of new algorithms for parallel computing and

b) application of these algorithms to problems from engineering sciences and physics.

a) has to rely on a somewhat constant environment to be able to evaluate the results from different algorithms, so this might not benefit from grid technology as I understand it from the previous questions. However, there are some applications in b) that can benefit from peer-to-peer computing and internet computing, as here acquisition of statistical data from numerical simulations is required.

The answer has to be rather a 'yes and no' as there are applications in our group where it could make sense and there are applications where it does not. [80]

In some sense, when grid computing is understood as an unified view on a larger number of high-performance computing devices (submit a parallel job and not caring about where and on what target it will be executed). [81]

We use one already. [82]

If the grid network can enable us run our calculations in reasonable times, with access to better debugging and profiling tools, then we may see some improvements in performance over those provide by our current facilities, e.g Cray T3E [83]

Our work is typically related to collaborative research projects. [84]

Constant need for more computational resources (mainly CPU power). Moreover, the abililty to run more than one image of each program in parallel, makes the grid network an ideal solution. [85]

Type of grid network that would best serve the group

One exclusive to the group	6
No geographical restrictions	38
One exclusive to our institution	7
One that only worked at a national level	9
No response	4
One exclusive to our department	4
One that served only researchers in the same field	16
Total	84

Comments:

Different levels of grid functionality are required for different purposes e.g.

1 Carrying out and analysing our high end simulations

2 Making our simulations available for analysis by the astronomical community world-wide

3 A Global Virtual Observatory requires coordinated action from the whole astronomical community

4 Outreach activities for simulation movies, interactive image manipulation and query response require interfaces to home internet capabilities as well as substantial compute resources at responsible nodes [1]

In my ideal computer world there would be an intelligent server system that would only send jobs to appropriate computers and if anyone would use my workstation via the grid I would earn credits that I could use later when I have a burst of activity and need to get many results fast on computers anywhere in the world.

It might be a nice gadget to see my computing credits being filled up when someone is running a job on my computer. [2]

As long as we can get our experiments done, then the location of the computer has less influence. At least to the extent were we are able to handle the data analysis locally and will not rely on super computers to make 3D data visualisation. In this case we will need a local machine capable of doing this or a nearby computing center with the required facilities. [3]

When system is getting bigger, is will be more problematic to handle. However, if we could co-operate within the discipline, that would be the optimum situation. [8]

People within the group are spread out over the country. [9]

Similar type of work would allow to optimize resources and set-up. [10]

Researchers at other institutions in the UK, Europe and further afield often request some part of our data holdings. At present a locally stored subset subset is available by anonymous ftp from a web interface. This could be extended to include data not stored locally. Grid technology could be used to identify where a particular dataset resides and access it without user intervention. [12]

In research there is no limits. If we think of operational work, the environment must be extremely reliable, giving peak performance at given times. [13]

I do not have a clear answer concerning with this item. In principle I do not see any problem about a grid network without geographical restrictions, but I am thinking about security problems which are very important , obviously. [14]

See answer above. Maximal potential for CFD required. [16]

I expect computational resources within my group to be enough to complete any computational task I will need in the near future [17]

I don't have strong geographical preferences as long as (i) there is a reasonable fast (interactive) access to the machine (ii) if the machine is in a different time zone it allows for interactive testing during "normal" times at my place [18]

See the comments to the previous question [20]

My group has transatlantic collaborators, with compute centres in Edinburgh (UK), Durham (UK), Munich (Germany) and McMaster (Canada). [21]

We collaborate with a lot of different groups, we have computational resources in both the USA and Europe. [22]

I am very sceptical about the profit of the Grid technology for computer scientists. My experience is, that it brings a lot of overhead and is not satisfying due to a lot of restrictions today. [23]

The simulation software we use is tailored for our own needs. Having a small local group will make it easier to make changes/improvements to the software. [24]

If the network works fine, then there are no restrictions [25]

We are interested in getting calculations performed as fast as possible. I don't matter where the results are calculated. [26]

---

our work increasingly involves multinational collaborative projects [27]

There are climate grids being developed in Europe, USA and Japan from which we can learn and use many techniques already being developed. [28]

First we want to learn; we deem it easier to do that in a local environment. [29]

The grid concept itself does not seem useful [30]

At an European-wide level, we should create Consortia or pool of structures to be deployed in the most "transparent" way from the end-user communities. This with a (primary) aim of better using European computing resources and to decrease the impact of the lack of large computing resources in some area of computational science. [31]

For our experiments the indicated level seems sufficient. [32]

See above [33]

Would be useful for the group or department level only if the hardware available were increased an order of magnitude [34]

Is it at this level that we miss the most any powerful and effective computational structure [36]

We would be glad to share the experience with researchers doing similar type of work mainly at national level but not excluding international collaboration [37]

Actually, I would prefer to have the opportunity to select several options (i.e One exclusively restricted to my group, and even to have one at each level :-).

But, since we do not work with private material, and we shared the idea that the best way to exploit resources is to share them, the last seems suitable for us.

Certainly, the choice is for the future, but not for nowadays, since we expect that the quality of services (bandwidth, administration, computational power) of such a geographically system will be better than any of the other options. [38]

I'm LSF nation-wide Multicluster administrator in Poland. [41]

Collaboration mainly with regional counterparts. [42]

People from different departments/universities are involved in our type of computational projects. [43]

Different projects have different needs to which it is hard to find universal solutions [44]

I am using computers for Monte Carlo simulation to analyse experiments done in my Department (NMR in solid) [47]

Compute-intensive work (rather than, say, interactive or visualisation etc) so that access to raw CPU time is more important than ownership, control, network etc. [49]

I would have also answered one that only served researchers doing similar type of work if allowed.

The Inter-institution bandwidth is still low

Imperial College is large and diverse enough to support Intra-College Grids [50]

---

very interested to how latency/bandwidth affects grid computing [51]

Difficult to make a single choice - I would tick different boxes depending upon different research efforts

1. Specific grid research would need exclusively restriction to my group

2. General astrophysics work - no restrictions

3. Certain work in the medical area - national level [52]

Large facilities are out of CZ (in general) [53]

Code sharing among universities should be promoted [54]

Need for co-operation with other groups and machine types [55]

See #4. [56]

What I actually have in mind is a grid for the Max Planck Society, for all of Germany. I think the problem which should be kept in mind when grid technology is developed is that of funding. I can hardly imagine a future in which cycles are "just there", and all you need is a technology to get access. Rather, at least in the high-end sector, there will be an institution who PAYS for the cycles - actually substantial amounts of money. Thus, such a system has to be able to do some sort of accounting. I fear that a system which is too open will simply be too difficult to handle, and generate too much overhead when institutions start to not only share, but actually TRADE cycles - but maybe I am wrong, and all this can be done easily. In that latter case, probably the maxim "the bigger the better" will work best. [57]

Two levels are interesting: one at group level with tight and adjustable shared tasks grid, and another one primarily to store data in standard form worldwide, generate data in standard form worldwide, and perform standard tasks.

Other types are also possible but only in more special situations. [58]

Many collaborations are international and often the most fruitful bring together different experience and models. [61]

Main application of grid technology could be in the field of Global Change to share data between research teams with no geographical restrictions. [63]

We are doing calculations that are very "demanding". Of course we don't mind to share the resources with other resources but we have to find a way as to be a benefit for all of us. Sharing with others with the same needs will cause more trouble than good... [65]

Because of the way we manage work. Typically this is done at the department level. Also because of the network bandwidth restrictions to outside of the department. [66]

See 4. [68]

I do not know enough about the available technology to make a less restrictive choice (data security etc.) [69]

My department needs some computer power for their work. [70]

This is supposed to be our testbed, so it should not be limited. [72]

With the resources available today, a the grid must at least be on an institutional level to become large and powerful. In near future, one could imagine setting up quite large local linux clusters/grids at a reasonable cost. [76]

The idea is to share as many computational resources as possible. Since in my country we don't have the necessary computational facilities, we could only gain in sharing resources world wide. [77]

Our work involves coastal and ocean modelling; as such we need access to global databases. Initially though, a grid at a national level would suffice. [78]

Initially there will be some geographical restrictions. We start from local access only first (testing purpose), but we will gradually relax that restriction. [79]

Main objective is development of efficient algorithms which requires a 'constant' environment in order to allow proper comparison of the results. I have doubt this could be ensured over a institutional level. [80]

High-performance computing is sometimes troublesome to handle. The direct contact with the person close to the machine is sometimes strictly required. [81]

It's the type we currently use. [82]

From our use of shared facilities we find that there is an almost permanent lack of computer resources due to 'overcrowding' and sometimes misuse of computing facilities. [83]

We are working to the definition of international research projects based on a large use of grid technologies for different applications. [84]

No difference where the location of the system is, as long as there it is accessible through the internet [85]

Interactive Access Requirement

Yes	37
No	17
Do not know	26
No Response	4
Tot	84

Comments:

To allow remote access and manipulation of both simulated and real data.

To allow more effective outreach activity. [1]

I want to run jobs with interactive graphics on large computers, because that helps in intuitive understanding of the computational algorithm.

Also it can give a good idea of the physics of the system I am looking at. (This makes computational physics a bit like experimental physics). [2]

The interface around the codes are constantly changed to fit the given problem => new compilations, testing before real runs, may to a large extend be possible to do this locally. [3]

So far I haven't seen a metacomputer environment which would work so well in practice that interactive access would not be needed. [6]

This is valid, if thinking of operational, daily services. [13]

Some of the target computers should admit a double use as personal workstations [17]

We mainly work with specialized programs (as opposed to standard software like Gaussian) which require testing/optimizing on the machine (or at least on a computer of similar type). [18]

Remote code development work. [21]

From experience, you always need to be able to go in to a machine and see exactly why things aren't working, or move data etc. Hopefully this won't always be true in the future. [22]

How can you optimize your programme on a computer with respect to its architecture (cache structure, vector pipes, etc.) and its compiling system if you have no interactive access?

What about debugging in batch mode? [23]

Experience has shown that there will be situations where you need to intervene interactively [25]

---

for code development and data processing [27]

Testing, monitoring and especially debugging [28]

It would be sufficient to have several front-end to the global resource in order to be able to apply for a query of a give resource (or pool of resources). [31]

____

For our developments Yes. [32]

____

I know too little about what GRID will be able to provide. But the HEP analysis requires substantial access to graphical tools. Most likely graphical interface through local machines with access to remotely processed data will be fine. Some kind for monitoring of processing will be necessary, but interaction with a running process is most likely not necessary. [33]

____

For some services, interactivity is required. [38]

____

There are some applications used by our users (Matlab, CAE, Mapple, MSI...) [41]

____

Computational Steering

Interactive Knowledge Discovery [50]

____

setting up my remote environment, compile source code [51]

____

Some test compilations etc set up tasks. These would be followed by batch orientated computation. [52]

____

The information needed from remote sites, can be formatted modified, or processed more economically at the site where it is available and where it is generated. [58]

____

Generally no, some sort of batch system would suffice. Although some means of interactive testing is desirable. [61]

____

We all loving to travel but sometimes its not possible. Furthermore it's very much convenient to working at your office instead of having to arrange a whole trip... [65]

____

We might need to start windows NT software that is not (yet) command-line oriented. [69]

____

For experimentation purposes. [72]

____

For compiling, unless we already have a binary version for that

target machine. [74]

____

If I need to compile the application on the target machine, it is easier when the porting is done interactively [75]

____

code developing and debugging, graphical visualisation. [77]

____

Parameters for the different programs must be checked by the users [79]

____

Interactive access is more suited to build up user trust into the grid environment. Also for debugging purposes it is more convenient to have interactive access with the same user environment available. [80]

____

Interactive access is more suited to build up user trust into the grid environment. Also for debugging purposes it is more convenient to have interactive access with the same user environment available. [81]

____

Interactive use is particularly important during development and testing of near 'production' size computer runs. [83]

Experience of Grid technologies

	Yes	No
CONDOR	7	78
LSF	13	72
CODINE	6	79
NIMROD	0	85
Globus	14	71
UNICORE	3	82
LEGION	3	82

Other Grid Technologies:

No [14]
Cactus Code (Grid application software) [22]
NQE, IBM LoadLeveller [41]
none [54]
Java RMI [56]
maybe PVM? [69]
VAX OS [70]
OpenPBS [80]

Comments:

We run LSF on our 24 cpu compact cluster and is not 100% satisfied. has problems handling user time and there for impose a fair share policy. My only be due to our lack of knowledge to set it up properly [3]

We are using Globus and we are implementing a Knowledge Grid on Globus. [4]

Only used for load balancing. Simple enough. [21]

The GUI does not keep its promise to have an easy access to different computing resources.

There is a lot of code written for secure login, but missing is:

- no shared file system on the different platforms

- no information about all the jobs currently in the system (no information when your job will be executed)

- very hard to use for program development

What about the political constraints? Which institution in this concert is really willing to give significant shares of computer time to users from different institutions or countries? [23]

No experience [29]

We have adopted a scheme with LSF and AFS, where all the resources of ENEA computing centers are related and let available by a GUI which allows to select the resources on the bases of the evidence of their current status (and use). This is a very simple and immediate tool which greatly helps the use of these resources. [31]

Adopting GLOBUS is just underway and it was a bit premature to pretend any experience worth to mention. [32]

Just started. We know what the programmes should be able to provide, but we have no actual measurements. We have submitted jobs from the NBI linux farm on the Lund University linux farm and visa versa using GLOBUS. [33]

Condor seems quite nice, but limited. We are trying also with Globus, but still does not work. [38]

Frankly, we do not have much experience [40]

Very easy to use, completely satisfying my requirements. [47]

Condor is very effective

Globus very low level and restricted [50]

I believe Globus is the most practical/advanced middleware available; it works pretty well, anyway it's not ready for production environments right now. [51]

Certification and security under Globus proved time consuming and during setup the latencies were high and required a relatively high performance gateway. We are working on this and should improve matters in the next few weeks [52]

Not very user friendly, difficult to implement [55]

Have used LSF for a 128cpu Origin 2000, and some minor testing of its multi-grid features with the other HPC centres in Norway. LSF was very nice as a batch scheduler.

Are working on EUROGRID which is a project which has evolved from unicore. Involves interactive access, file transfers, multigrid resource broker, etc..

Are using JAVA RMI for our distributed GRISK (http://www.ii.uib.no/grisk/ project) [56]

---

no experience [57]

Would you consider running a parallel CFD program on a group of workstations as grid- technology? As a test it was effective, and I could imagine using a similar system to perform some of the tasks I mentioned earlier [69]

There is no easy use so far. Effectiveness is hampered by too much of administrative problems between sites. [72]

For our applications LSF has been easy to use. [76]

We have evaluated Codine as a batch system for our clusters of workstations and for the CLiC parallel computer. However we found it does not suit our needs and the final decision was in favour of OpenPBS which is in production for almost 2 years now. [80]

Codine has been evaluated at University as a batch system for our clusters of workstations and for the CLiC parallel computer. However we found it does not suit our needs and the final decision was in favour of OpenPBS which is in production for almost 2 years now. [81]

I find Globus to have an untidy design and to be user-unfriendly. [82]

My group have recently made an installation of GLOBUS on a cluster of Beowulf type and we are experimenting it. [84]

Ranking different factors

	ni	si	im	vi	tot

Faster CPUs	05	03	27	49	84
Greater total memory	13	09	31	31	84
More processors	06	06	30	42	84
More memory per processors (DM)	17	13	33	21	84
Production run environment	23	19	21	21	84
Test scalability on more procs	21	25	29	09	84
Longer run times	13	13	31	27	84
Larger storage devices	24	18	26	16	84
Fast I/O	30	17	24	13	84
Better throughput	15	12	32	25	84
Better turn around time	21	09	39	15	84
Ease of use	13	14	30	27	84
Security of code/data files	27	21	19	17	84
Best machine for the job	14	16	28	26	84
Resources not locally available	15	15	30	24	84

Where:

ni = “Not important”
si = “Of slight importance”
im = “Important”
vi = “Very important”
tot = “Total”

Other Resources:

I don't understand what the importance of the Production run environment is. [2]

Access to remote data and remote applications [4]

More security as much as possible!!! [41]

Ease and versatility of use [58]

access to machines with high memory bandwidth [80]

high-bandwidth low-latency communication networks for parallel computing combined with fastest processors available [81]

Code development tools [83]

Seeking access to resources not locally available

Evaluate performance	26
Use only once	06
To test portability	14
No response	39

Other Use for Resources:

Do production runs [2]

run future numerical experiments [3]

to access remote programs [4]

Use permanently [9]

Large production runs [11]

To know about and characteristics [14]

---

production runs [18]

Production [21]

---

production runs [23]

---

for production use [27]

---

production [28]

---

to have a faster machine for the computation [29]

To process data [33]

---

to increase computer time availability [34]

To test portability [41]

To work [43]

For production [44]

---

use continuously for my calculations [47]

PRODUCTION [57]

All of the above may come up. [58]

Use often! [64]

---

national super computing resources [68]

---

production runs [71]

To do production runs [73]

---

production runs [77]

---

production [79]

---

both "performance evaluation" and "test portability" [80]

---

execution of very large, not every-day CFD-computations [81]

Comments:

For us the importance of grid technology is for federating activity on large archives with a broad geographical distribution. This activity will often be compute intensive and so will require geographically distributed processing. [1]

That depends on the problem I am working on. It may be 10000 linux computers to run a 5 hour job that I need good statistics on or it may be a large Origin3000 to perform one very large simulation.

I will do performance evaluations, but only if I really have to. I much rather do the physics (and waste a bit of computer-time). [2]

Large parallel computers. We are working with 3D time dependent MHD codes => to make more realistic model large array sizes are required. This implies access to large amounts of RAM and disk storage and many cpus to cut down production time. [3]

Supercomputers/high performance computers [9]

Parallel platforms (SP, T3E, SGI) [11]

---

large aggregate memory, fast CPUs [21]

I don't understand this question. [22]

Looking for faster systems which are suited to run my codes as fast and efficiently as possible. [23]

Fast CPU's, large amount of memory [26]

---

a variety of computers which are cost-effective for particular aspects of our work eg high-memory shared memory for data analysis, very high performance for data generation, plus distributed data storage [27]

Local resources are not sufficient for production experiments [28]

---

resources with more than 1000 processors [29]

Large MPP platforms, specialized HW (i.e. GRAPE, QCD machines etc.), large Vector facility. [31]

Big clusters of a type that is not available locally. [32]

CPU, storage and network. [33]

Fast parallel machine for large scale calculations not possible on conventional machines, in an easy to run environment [35]

HP series 720 or higher machines [37]

---

parallel machines [38]

Some grid-based computational tools [40]

Other operating systems, compilers, math libs and faster cpus. [41]

Distributed memory systems [42]

Processing power [43]

See below [44]

The speed is most important in my calculation. For example: simulation of diffusion in solids for 50000 atoms is a very time consuming calculation, and to find the best model I have to compare experimental results with many sets of calculations. Such a procedure takes hundreds of CPU's hour on CRAY SV1-1A (the one available for me). [47]

I don't understand the question - what 'resources'? And surely all grid resources are not 'locally available', rather, they are distributed. [49]

More computing power [54]

---

visualisation equipment [55]

Big machines with lots of power. Both coherent parallel machines like eg a Cray T3E, and things like Beowulf clusters. [57]

Usually resources we need and do not have. This varies in time. Most usually some software or library is missing. [58]

CPU time and storage [63]

Speed, memory, more CPU's [69]

---

faster and less occupied mainframe [70]

---

large memory, large bandwidth, large cpu speed [71]

---

for example data repositories or other computing platforms [72]

A parallel or virtual parallel machine with large number of processors and large memory [73]

---

large memory, fast CPUs, Np>>1 [77]

Not sure what this question is asking. [78]

---

parallel machine [79]

- different operating systems and architectures

- different interconnecting networks (SAN)

- different computing concepts (e.g. shared memory vs. message passing) [80]

---

fast MIMD parallel machine with compatible C/MPI programming environment [81]

Multi-processor resources [83]

Tackling bigger problems

Yes	60
Do not know	14
No	08
No Response	02
Total	84

Comments:

Comparative analysis of large simulations carried out at different sites

Correlation queries on large astronomical databases held by different institutions (e.g. ESA, ESO, NASA) [1]

I recently ran a large number of jobs that took about 7 days on a linux computer or 12 hours on 64 processors on the t3e. I ran into a time-limit on the t3e and would have loved to simply run the job on many linux-computers for about 14 days. With a bit of localized data analysis the size of the resulting data-files would not have been not too bad. [2]

Distributed data mining.

Grid programming. [4]

Increase the system size in some simulations to tackle problems which can not be handled now. [6]

Receptor and enzyme protein motion within membrane and with small ligand molecules [8]

To study surface processes on larger, and more realistic surfaces. [9]

Calculation of larger and thus more realistic models of chemical systems, with the use of more expensive, and thus accurate computational methods. [10]

First-principles simulations of biochemical molecules (> 1000 atoms) [11]

We would be encouraged to couple ocean, atmospheric and bio-geochemical models at high resolution possibly each running on a different platform. [12]

Heterogeneous reactions, reactions in solution, large polyatomic collisions,etc. [14]

With reservations. For the time being the parallel computing resources available to us have been adequate, but increased computational power is always welcome. Full scale ship flow simulation is an example of a computationally intensive case. [16]

Molecular Dynamics simulations of large biological systems. Application of Quantum Mechanics in some steps of the research replacing molecular mechanics [17]

Problems in soft-condensed matter often involve a wide spread of time and length scales. Problems include prohibitive long relaxation times of glassy systems (eg, studying ageing phenomena) to a wide spread of length scales in multi-component systems (eg, from the interfacial structure on the length of Angstroems to the domain size on the micrometer scale). [18]

Higher resolution models [21]

More accurate and detailed simulations. [22]

MD simulations with very large number of particles

Full QCD simulations on larger lattices

The question is: Would I spent a lot of time to execute these codes on connected systems or would I prefer to run these on one larger system? At the moment I would prefer the latter one, because it is too complicate to have a combination of systems combined for a longer time period (not to speak from data distribution). [23]

Simulation of large biomolecules, in vivo simulation of biochemical reactions, computer assisted drug design [25]

Solving larger problems then possible on the computer resources available in house. A large part of the computations we perform is solving systems of linear equations. At present we solve systems with up to 100.000.000 unknown, but we have problems which require solving systems with more than 500.000.000 unknowns. [26]

---

this depends on the resources available - since our work is computationally limited, a grid should allow us to do more and/or work more efficiently through greater sharing of resources [27]

Coupling of multi component models for Earth System Science experiments [28]

LES fro e.g. transonic flows over a wing [29]

Porting of a code on a “mixed” architecture (MPP+Vector+specialized machines) in order to increase the level of computational efficiency reached on a given application. [31]

I believe that HEP provides sufficiently complex problems for the time being [33]

To go into higher dimensions, finer meshes and time scales [37]

Combinatorial Optimization Problems. Dynamic Programming. DNA problems [38]

---

try solve some combinatorial problems (came from graph th.) [41]

---

quantum chemical calculations on systems too large to handle by conventional methods. [43]

Electronic structure calculations of bigger systems [44]

The results of Monte Carlo simulation of solids are better for larger "sample" taken for calculations. The size of such sample is restricted by CPU speed. The memory requirements are not the problem with the machines I can use. [47]

Larger and more realistic chemical systems, such as including solvent effects. [49]

Parameter Searching. Collaborative Interactive Science [50]

Large scale simulations of pulsar magnetospheres and radiative transfer for patient treatment planning. [52]

Data mining used in transport area and rolling Mill optimisation (project Prodactool) [53]

Complex flow simulations; LES; two-phase flows; increased resolution [54]

Some of the physics applications we are interested in are scaleable [55]

Affordable computing, GRISK http://www.ii.uib.no/grisk [56]

Eg. we could develop, more detailed and so more realistic reaction models. [58]

Large coupled models simulation [63]

Bigger systems (more atoms), more sophisticated algorithms, clusters, vdW complexes.... [65]

Interactions between fluids with a free surface and moveable structures. [69]

simulate more complex problems with more number of parameters [70]

on could at least give it a try. Flow Simulations, Combustion, Turbulence research [71]

some multiphysics applications [75]

In applications, we generally must limit our models with respect to size and accuracy, and "real size" systems and accurate calibrations are always aimed for. [76]

Simulations with larger (astrophysical) systems and for longer (physical) time. [77]

provided data infrastructure on line, would want to run real-time forecasting of coastal water quality. [78]

Large scale computer simulations of complex biological systems post genomic era applications [79]

combined computation of gas-particle flows with heat, momentum and mass transfer between fluid and particles and with interaction between particles themselves; turbulence interaction with particles and vice versa [81]

We developed an Operational Air Quality Model for Campania Region, based on two uncoupled models: a Circulation Model (for weather forecasting) and an Air Quality model (for air pollution forecasting). We could approach a system with coupling of phenomena. [84]

Perform simulations more demanding in CPU-power resources [85]

Architectures

	Yes	No
Fast single CPU	35	50
Multiple procs, shared mem	51	34
Multiple procs, distributed mem	57	28
Vector processing	16	69
SIMD	28	57
Large memory	28	57

Specific Machines:

All the above are already available [1]

SGI, T3E, IBM-SP, PC-cluster [11]

SGI [25]

no preference [29]

Origin, Sun, CRAY, Beowulfs [38]

scalar architecture - SMP e.g. SGI MIPS based, Compaq Alpha based [41]

Cray, SGI, IBM [43]

IBM SP, SGI ORGIN, LINUX CLUSTER [44]

like CRAY SV1-1A [47]

UNIX HP ???) [53]

Cray T3E, Origin2000, Alpha/Linux clusters [62]

SGI with 12 processors sounds good [65]

Challenge [70]

NEC; Hitachi; VPP and others [71]

Other Architectures:

shared/distributed multiprocessors [20]

It should be irrelevant [28]

dedicated or specialized (i.e GRAPE) [31]

____

parallel machines [35]

____

Part 3 – User Profile

Field of Work

Astronomy	07
Biochemistry	01
Chemistry	16
Climate	02
Computing	14
Engineering	11
Genetics	01
Mathematics	02
Other	11
Physics	17
Tele	03
Total	85

Other Fields:

parallel computing [4]; Semiconductor physics [11]; Air Pollution Modelling [20]; User Support and Physics [23]; fluid mechanics [29]; academic institution, university [37]; solving equilibrium systems [40]; Geology [42]; Molecular Modelling [46]; bioinformatics [67]; Atmospheric Chemistry /Meteorology [68]; oceanography [78]; Mathematics, Physics, Engineering Science and Computer Science [80]; Combination of Physics, Chemistry and Materials Science [85];

Problem types:

Cosmological simulations

Astronomical Data Analysis [1]

Simulation of multiphase and viscoelastic fluids. [2]

Solar physics using the MHD equations to describe the magnetised fluid behaviour of both the interior and the atmosphere of the sun. [3]

---

data mining of large scientific data sets [4]

---

ab initio molecular dynamics simulations in condensed phases [5]

Materials Physics on an atomistic level [6]

Molecular design of multiphoton absorption materials, nonlinear optical materials and molecular electronic devices [7]

Protein modelling, Creation of new drug molecules [8]

Simulations of thin film growth [9]

Modeling of transition metal complexes, with special emphasis on structure, homogeneous catalysis and bioinorganic chemistry. Use of standard ab initio methods, as well as development and application of quantum mechanics/molecular mechanics methods. [10]

First-principles calculations of defect related properties in semiconductors. Calculations based on density functional theory. Clusters/supercells with 100-600 atoms used and the program is developed within my group. [11]

---

high resolution global ocean modelling [12]

Weather forecasting using numerical atmospheric models [13]

Dynamics of elementary reactions involving 3,4 or 5 atoms and using Quantum Mechanics tools or Quasiclassical Trajectory Methods. [14]

Numerical simulations on nonequilibrium phenomena [15]

Computational simulation of ship flow problems with a free surface. [16]

---

drug design, modeling of ligand-receptor interaction, mathematical models describing biologic properties of small molecules [17]

---

structure and dynamics of glasses (both idealized models as well as modelling of silica and polymer glasses) phase transitions in macromolecular materials (phase separation of mixtures, mesophase formation of block copolymers, tilting transitions in lipid monolayers, adsorption of polymers at surfaces) [18]

Experimental High Energy Physics [19]

Studying high pollution levels in Europe by large-scale air pollution models described by systems of PDEs [20]

N-body gas dynamic simulations [21]

Numerical relativity (black hole, neutron star, gravitational wave simulations) [22]

In general we try to support our users with respect to code paralleization and general use of our CRAY T3E and T90 systems.

By myself I am very interested in full QCD simulations to study the influence of sea quark effects and to find fast algorithms to calculate the fermion determinant.

In our group we are also investigating algorithms for MD simulations. [23]

Direct numerical simulation of unsteady, periodic flow over turbine blades. The objectives of these simulations are to provide reference data for the development/improvement of turbulence models and to better understand the underlying physics of the interaction of free-stream turbulence and boundary layers c.q. separation bubbles. [24]

Program development, application in quantum chemistry [25]

Analyzing large amount of data from production animals. The purposes of these analyses are:

1) Generate knowledge about the genetic background and relationship among traits of importance for the different livestock industries.

2) Predict breeding values for us in the selection process to insure genetic progress in the breeding goal for the species in question. [26]

QCD [27]

---

climate research on a decal to 100 year timescale, atmospheric modelling, coupling atmospheric models to ocean, chemical, and vegetation models [28]

Turbulence. Numerical Simulations of turbulent flows [29]

Materials physics [30]

We are currently developing novel methods to cope with computationally intense problems in protein modeling, genomic and proteomic analysis, material science (at the atomic scale), astrophysics. We mainly deal with "particle" models but, more recently, we have also tackled problems of sequence analysis and finite elements (for the simulation of EM propagation in different media). [31]

New computer systems architeture [32]

Handling Petabyte data from the ATLAS experiment at LHC, looking for the smallest building blocks in nature and their fundamental interactions. [33]

Electronic structure of molecules and magnetic properties [34]

Theoretical studies of crystals with technological properties (design and properties, polymorphism). [35]

Atomistic modeling of materials [36]

Mathematical modelling of combustion engines, modelling of work cycles including the transport and chemistry [37]

——————————————————————————*Parallel Algorithms for Combinatorial Optimization. Performance Modelling. Languages for Parallel Computing. Dynamic Programming. Branch and Bound. Divide and Conquer. Nested Parallelism [38]*

Gravitational wave data analysis [39]

We are interested in embedding of computational chemistry into general flow and transport modelling systems. A part of our activity is devoted to solving nonlinear systems concerning various types of chemical equlibria. [40]

Distributed processing, simulation of queuing systems, graph theory and their usage. [41]

Underground water flow modelling [42]

Quantum chemical calculations of large molecular systems, including adsorption of large organic molecules on semiconducting surfaces. [43]

Electronic structure calculations of surfaces, interfaces, and materials on atomic level and also dynamics. The overall goal is to develop a physical understanding on an atomic scale of various materials, surface and interface phenomena and to predict physical properties of systems that have not yet been made. [44]

---

1) New discretization schemes for PDEs

2) Iterative methods for linear algebraic systems

3) Air pollution Models and Systems

4) Models for Superconducting materials

5) Distributed Systems [45]

Protein unfolding and stability. My specific problem is studying the stability of the eye lens protein family gamma-crystallins, using several unfolding protocols. The objective being an understanding of the early events involved in cataract formation. [46]

Simulation of internal dynamics in solid state. [47]

---

materials modelling [48]

Surface structure and reactivity of complex oxides. [49]

Grid Middleware Development and deployment [50]

We are interested in a number of problems - at the moment our work is in the following areas

1. Pulsar Magnetosphere Simulations - both monte carlo and analytical

2. Development of Parallel Image Deconvolution Algorithms

3. Development of radiative transfer simulations - mainly monte carlo [52]

Data mining [53]

Compressible and incompressible flows in turbomachinery; two-phase flows; unsteady flows; reacting flows in nonequilibrium [54]

Middleware. Scientific applications [55]

---

no [56]

All sorts of problems in theoretical polymer physics: Fundamental questions like basic physical mechanisms in dynamics of polymer solutions and melts, conformations of charged systems (polyelectrolytes, charged complexes,...), electrorheology, rubber elasticity, but also more applied research concerning the mapping of different levels of modeling onto each other. [57]

Atomic and subatomic reaction modelling. [58]

CFD [60]

Photonic Devices [61]

Computational modelling and simulation of materials [62]

The job of my group is to afford to researchers the best computing environment. Real target of this study is CERFACS' researchers. Their main fields of interest are:

- *Global Change*
- *Computational Fluid Dynamics*
- *Electromagnetism*
- *Parallel algorithms*

Following answers are for CERFACS' researchers [63]

Catalysis, Thermochemistry [64]

Linear and non linear properties of atoms, molecules, clusters. Intermolecular interactions. Molecular engineering [65]

We are evaluating a platform that we built for developing application-enabled mobile agent systems. [66]

Protein structure comparison and protein structure prediction

Genome anlaysis - annotation, comparison of genomes

Microarray (gene expression) data analysis [67]

Emissions ® Transport ® Change in chemical composition ® Depositions.

Numerical simulations of meteorological and chemical processes of importance on scales from global to local using both comprehensive weather prediction and chemistry transport models, and chemistry box models to test out 'new' reactions and species. [68]

Control of irrigation and drainage systems. This includes:

- Study of the behaviour of flow regulation devices under different conditions, including conditions outside their design specifications.

- Study of open channel hydraulics.

- Study of linear and non-linear control.

- Optimisation of water use. [69]

---

new concepts of the transmitter and receiver in mobile radio systems [70]

CFD: solving PDE's with finite volume schemes, Lattice Boltzmann, Multigrid, Large sparse Matrices [71]

---

grid programming models and supportive tools [72]

Electro-Magneto-Hydrodynamic instabilities and pattern formation [73]

Quantum chemistry: electronic structure calculations [74]

---

combustion, aerodynamics [75]

Chemical reactions, in particular, catalyzed polymerization and refinement of different small organic molecules.

High-resolution (synchrotron-radiation based) photoelectron spectroscopy. [76]

N-body systems: close encounters and instabilities.

Planetary systems: stability and formation.

Non-linear networks: numerical solutions.

Numerical integration of ODE's (parallel algorithms). [77]

---

shelf / ocean dynamics and biogeochemistry;

resolve shelf / ocean dynamics;

determine nutrient & carbon budgets & fluxes;

quantify primary production;

assess exchanges between ocean & shelf seas;

explore climate change scenarios; [78]

Protein modelling

Protein-ligand interactions

Binding to membrane surfaces [79]

- adaptive Finite Element algorithms and parallel solvers for positive definite Problems

- domain-adaptive wavelets and parallel multiscale methods for boundary integral operators

- fine structure of resonances in the Bernoulli-Anderson-Model

- parallelization of irregular numeric algorithms

- SAN couples clusters (SAN Grid experimental system)

- electronic states in amorphous materials

- transport properties of disordered correlated many-particle systems

- simulation and modelling of relaxation processes in complex systems

- band structure calculations

- long term evolution of huge dynamic systems

- simulation of plastic/elastic deformation of materials

- efficient parallel algorithms for numeric simulation of multiphase flow

- methods for simulation of gas kinetic processes

- integral equation methods for coupled electromagnetic and charge-transport problems

- parameter identification in material sciences [80]

---

flow simulation (CFD) of mixtures of fluids/gases with particles/droplets with fluid-particle and particle-particle interaction [81]

Development of technologies based the condensed graph computing model, and the development of an Irish computational grid. [82]

Computation of inert and combusting turbulent flows using Direct Numerical and Large Eddy Simulation techniques [83]

Mathematical Software Libraries, Advanced Environments for HPC, Air Quality Modelling, Visual Computing, Digital Film Restoration [84]

Research in our group aims at the development and implementation of molecular modelling strategies for predicting the properties of materials from chemical constitution. The scale of the simulation strategies followed ranges from atomistic to mesoscale. [85]

Operating Systems Used

Unix	54
NT	12
Windows 2000	18
Total	84

Unix flavours:

Linux, AIX [1]; linux [2]; SUN OS, Linux, Irix, DEC OS [3]; Solaris, Linux [4]; Linux, DEC, Sun, Cray, SGI, IBM [6]; T3E, SGI, Alpha [7]; IRIX, linux, SunOS [8]; AIX, IRIX [9]; Linux, HP, SGI, IBM [10]; Std UNIX or LINUX [11]; in the group PC machines (Linux) or workstations (AIX,HP-UX,IRIX)ns as [14]; Dec-Unix, IRIX [15]; Silicon Graphics [16]; IRIX, Linux [17]; SuSe Linux [18]; Linux [19]; many forms [21]; AIX,IRIX,LINUX,UNICOS,OSF,SOLARIS,HPUX [22]; UNICOS/mk, UNICOS, AIX, Solaris [23]; HP-UX, and UNIX on the IBM-SP, VPP, Hitachi SR8000 [24]; Irix 6.5 [25]; SUN Cluster [29]; Linux, Irix, Tru64 [31];

Linux [32]; linux (Redhat 7.1) [33]; Irix [34]; hpux [37]; all sorts of [38]; Linux [40]; IRIX, Unicos, Unicos/mk, Linux, Solaris, Tru64 [41]; IRIX, LINUX [43]; Solaris, IBM AIX, Linux, ... [44]; linux, irix6.5, OSF [46]; linux, AIX, DEC, SGI [49]; Tru Unix 64, linux [51]; Linux, DEc UNIX, Ultrix [52]; HP 10.20 [53]; SuSE Linux [54]; BSD [55]; solaris, linux, irix [56]; DEC, Linux, Irix (phasing out) [57]; AlphaDEC, SGI [60]; Linux, True Unix etc. [62]; Irix, Solaris [63]; Linux, HPUX, etc. [64]; IRIX 5.3 [65]; Linux & BSD [66]; Solaris [67]; IRIX / Linux [68]; Linux [70]; mostly SOLARIS, HPUX [71]; Linux [72]; Linux [73]; IRIX, SunOS, AIX [74]; sgi, fujitsu [75]; irix, aix [76]; Linux [77]; SG workstations & CSAR T3E [78]; Linux [79]; Linux, True64 Unix, HP Unix [80]; HP-UX, Linux (SUSE, RedHat) [81]; Linux and FreeBSD [82]; Compaq, SGI, Cray [83]; IBM, SUN [84]; IRIX,AIX,Linux [85];

Other OS:

Windows 95 [17]; Linux [24]; Windows 97, Windows 95 [26]; Linux [34]; Mac OS [35]; Linux [53]; wherever a java virtual machine is available [56]; Imux, Win98 [65]; vendor specific UNIX [71]; Linux [84]; win98 [85];

Type of User

Black box users: 26

Composition:

0%	53
1-25%	16
26-50%	04
51-75%	06
76-99%	04
100%	02

Users: 42

Composition:

0%	27
1-25%	28
26-50%	19
51-75%	06
76-99%	02
100%	03

Developer-users: 59

Composition:

0%	08
1-25%	21
26-50%	09
51-75%	15
76-99%	18
100%	14

Part 4 – Application Profile

Language

Do not know	01
C	56
C++	32
Java	18
F77	54
F90	55
F95	13
HPF	09
OpenMP	13
MPI	53
PVM	11
SHMEM	15

Other languages:

ZPL [2]; Perl [22]; Python [30]; PERL [43]; Python [44]; matlab [69];

Bottlenecks

CPU Constraints:

Not an issue	05
Minor	11
Major	43
Critical	26

CPU to main memory performance:

Not an issue	16
Minor	20
Major	37
Critical	12

Disk I/O Performance:

Not an issue	28
Minor	39
Major	17
Critical	01

Memory capacity:

Not an issue	14
Minor	22
Major	37
Critical	11

Data storage capacity:

Not an issue	25
Minor	34
Major	20
Critical	06

Network performance:

Not an issue	25
Minor	35
Major	17
Critical	08

Other bottlenecks:

Network performance can be important for interactive graphics, but we don't do it very often. [2]; You naturally scale to problems so that they fit to the computer you have access to => you can always use something larger/faster.... [3]; Poor turnaround is the major problem at present. The power is there but shared between too many jobs. [12]; graphic performance, major [17]; getting enough processors [78];

Machine Input Formats

0 bytes	01
< 1 Mb	23
< 10 Mb	15
< 100 Mb	17
< 1 Gb	18
> 1 Gb	11

Formats:

ASCII	60
Binary	44
Proprietary	13
Machine independent	04

Other input:

depend on the situation - new experiment or restart of an old experiment! [3]; GRIB, BUFR: general meteorological packing formats [13]; will there be grid support for matlab? [69]; gzipped data files [79];

Machine Output Formats

0 bytes	01
< 1 Mb	23
< 10 Mb	15
< 100 Mb	17
< 1 Gb	18
> 1 Gb	11

Formats:

ASCII	56
Binary	53
Proprietary	14
Machine independent	05

Other input:

*GRIB: meteorological packing code [13]; short integer (int*2) [68]; shortly to eb netCDF [78]; gzipped data files [79];*

Defining Job Requirements

Memory:

Yes	63
No	04
Do Not Know	08
No Answer	10

Memory requirements:

512(128) Mbs [1]; 300 (can be much more though) Mbs [2]; up to 20 Gbs - dependent on domain size! Mbs [3]; 100(50) Mbs [5]; 128(64) Mbs [6]; 100 Mbs [7]; 256 Mbs [9]; 200 Mbs [10]; 100/64 Mbs [11]; 256(64) Mbs [12]; 70 Mbs [13]; 1000 Mbs [14]; 512 Mbs [15]; 600 Mbs [16]; 125 Mbs [17]; 10(1) up to 256(64) Mbs [18]; 250(1) Mbs [19]; as much as is available Mbs [21]; 200(64) Mbs [23]; 400(60) Mbs [24]; 1000 Mbs [25]; 2000-10000 Mbs [26]; 128(16) Mbs [27]; 64(16) Mbs [28]; 900 Mbs [29]; 400(16) Mbs [30]; 200 Mbs [34]; <1 GB Mbs [35]; 3500 Mbs [36]; 1000 Mbs [37]; 256(16) Mbs [38]; 512 Mbs [39]; 400 Mbs [40]; 100 Mbs [41]; 200 Mbs [42]; 500 Mbs [43]; 512Mb/node Mbs [44]; 1 Mbs [45]; 30 Mbs [46]; 500 Mbs [48]; 512 Mbs [49]; 200 Mbs [50]; 512 Mbs [51]; 1000 Mbs [52]; 1000 Mbs [53]; 5 000 Mbs [54]; 10 Mbs [56]; 20(64) Mbs [57]; >512 Mbs [59]; 1Gb Mbs [60]; 64 (128) Mbs [62]; 300(32) Mbs [63]; 256 Mbs [64]; 64 Mbs [66]; 500 Mbs [67]; 400 Mbs [68]; 100 Mbs [70]; 9000 Mbs [71]; 140(4) Mbs [73]; 100-2000(1-32) Mbs [74]; 500 (16) Mbs [75]; 50 (8) Mbs [76]; 16 Mbs [77]; 512(256) Mbs [78]; 100-200 Mbs [79]; 350 (64) Mbs [81]; 10(128) Mbs [83];

Run time:

Yes	72
No	03
Do Not Know	05
No Answer	05

Run time requirements:

3 days on a days with 512 procs [1]; 12 hours on a hours with 64 procs [2]; 5 days on a days with 40 procs [3]; 10 hours on a hours with 50 procs [5]; 12 hours on a hours with 64 procs [6]; 600 minu tes on a minutes with 32 procs [7]; 20 hours on a hours with 32 procs [8]; 6 days on a days with 1 procs [9]; 2 days on a days with 1 procs [10]; 12 hours on a hours with 64 procs [11]; 12 hours on a hours with 64 procs [12]; 60 minutes on a minutes with 128 procs [13]; 4/5 days on a days with 1 procs [14]; 10 hours on a hours with 1 procs [15]; 5 days on a days with 1 procs [16]; 3 days on a days with 1 procs [17]; 4 hours on a hours with 32-128 procs [18]; 25 hours on a hours with 16 procs [20]; 10 days on a days with 64 procs [21]; 4 hours on a hours with 64 procs [23]; 33 days on a days with 60 procs [24]; 500 minutes on a minutes with 5 procs [25]; 1- 50 days on a days with 1 procs [26]; 8 hours on a hours with 128 procs [27]; 12 hours on a hours with 16 procs [28]; 100 hours on a hours with 32 procs [29]; 2 days on a days with 16 procs [30]; 1 days on a days with 1 procs [33]; 7 days on a days with 1 procs [34]; 3-10 days on a days with 1 procs [35]; 5 hours on a hours with - procs [36]; 2 days on a days with 2 procs [37]; 5 minutes on a minutes with 1 procs [38]; 100 days on a days with 100 procs [39]; 20 hours on a hours with 1 procs [40]; 2 days on a days with 1 procs [41]; 3 days on a days with 2 procs [42]; 2 days on a days with 8 procs [43]; 8 hours on a hours with 24 procs [44]; 4 hours on a hours with 1 procs [45]; 10.5 hours on a hours with 1 procs [46]; 3 days on a days with - procs [48]; 12 hours on a hours with 16 procs [49]; 2 hours on a hours with 32 procs [50]; 50 minutes on a minutes with 1 procs [51]; 2000 hours on a hours with 1 procs [52]; 100 hours on a hours with 1 procs [53]; 14 days on a days with 50 procs [54]; 1000+ days on a days with 1 procs [56]; 200 days on a days with 64 procs [57]; 50 minutes on a minutes with 1 procs [59]; 16 hours on a hours with 4 procs [61]; 4 hours on a hours with 64 procs [62]; 12 hours on a hours with 8 procs [63]; 2-3 days on a days with 1 procs [65]; 1 hours on a hours with 1 procs [66]; 60 minutes on a minutes with 1 procs [67]; 15 hours on a hours with 4 procs [68]; 4 days on a days with 1 procs [70]; 1 days on a days with 8-16 procs [71]; 2 hours on a hours with 4 procs [73]; 0.1-5 days on a days with 1-32 procs [74]; 12 hours on a hours with 8 procs [75]; 1 days on a days with 8 procs [76]; 40 minutes on a minutes with 1 procs [77]; 12 hours on a hours with 256 procs [78]; 10-30 days on a days with 2-8 procs [79]; 4 hours on a hours with 128 procs [81]; 20 minutes on a minutes with 16 procs [82]; 500 hours on a hours with 64 procs [83]; 10 days on a days with 1 procs [85];

Disk storage:

Yes	41
No	25
Do Not Know	12
No Answer	07

Temporary disk requirements:

500,000 Mbs [1]; 300 Mbs [2]; 100 Mbs [7]; 200 Mbs [10]; 20000 Mbs [12]; 1000 Mbs [13]; 1000 Mbs [14]; 500 Mbs [17]; 100 Mbs [18]; 100000 Mbs [19]; 1000 Mbs [21]; 9660 Mbs [23]; 5000 Mbs [25]; > 1.000 Mbs [26]; 1000 Mbs [27]; 2000 Mbs [28]; 2000 Mbs [29]; 200 Mbs [30]; 500 Mbs [33]; 4000 Mbs [34]; <4GB Mbs [35]; 75 Mbs [37]; 128 Mbs [38]; 1000000 Mbs [39]; 300 Mbs [42]; 100000 Mbs [43]; 10000 Mbs [44]; 0 Mbs [46]; 0 Mbs [49]; 1000 Mbs [53]; 5 000 Mbs [54]; 0 Mbs [57]; 5000 Mbs [59]; 10000 Mbs [60]; 100 Mbs [63]; 500 Mbs [64]; 20 Mbs [66]; 300 Mbs [68]; 1000 Mbs [71]; 2 Mbs [73]; 2000 Mbs [74]; 100 Mbs [76]; 15000 Mbs [78];

Recompilation frequency:

Never	15

Every time we have to look at a new problem	41
Before every run	05
Only when we combine a different set of modules	24

Number of runs before recompilation:

1	16
< 10 times	39
< 50 times	10
< 100 times	07
> 100 times	13

Applications used:

Own code (partly developed at EPCC) [1]; I call outside routines like convert (which uses gs) to produce movies. [2]; own codes [3]; CPMD car-parrinello MD [5]; own code + DMol + Castep [6]; Gaussian, Dalton, own code [7]; Gaussian, GROMACS, Sybyl, Unity, own code [8]; CASTEP, DMOL3 GAUSSIAN (not used on my work stations) [9]; Gaussian, ADF, molpro, Jaguar, Gamess, own code [10]; Own code [11]; own code [13]; own code [14]; own code [15]; own code [16]; AMBER, GRID, GAUSSIAN, GAMESS, MOPAC, mipsim (own code), golpe (own code), volsurf (own code), almond (own code) [17]; own code [18]; Blas, Lapack [20]; own code [21]; Own code (Cactus Code) [22]; own code, Amber for benchamarks [23]; tecplot, ensight, gnuplot [24]; Gaussian, ACES2, NWChem, GAMESS-US [25]; SAS [26]; Unified Model from UKMO, Off-line chemical models - own code, Data analysis codes -own code, Ocean codes from SOC, LODYC, France, OASIS from Cerfacs, France [28]; own code [29]; InsightII (for protein modeling) (BLACK BOX) GROMACS (for protein modeling) (source code available), FHI (for DFT calculations) (source code available), LAUTREC (for DFT calculations) (source code available), gravitational n-body (home-made), Tight Binding Molecular Dynamics (home-made), many-body potential molecular dynamics (home-made), Reverse Monte Carlo for the study of amorphous systems (home-made) [31]; Own code [33]; Gaussian, Crystal, CASTEP, Jaguar [34]; Gaussian (recompiled by us), Own code [35]; Mostly Car-Parrinello molecular dynamics [36]; own code [37]; own code [39]; own codes (mostly) [40]; own code [41]; GAUSSIAN98, CRYSTAL98 [43]; AMBER, other group members use Gromacs. Also own code for analysis of the output [46]; VASP [49]; Own Code [52]; own code [53]; FLUENT [54]; own code [57]; own codes (in most cases) [58]; own code [61]; own codes [62]; Numeca (CFD), arpege (Climate), opa (Climate), nsmb (CFD), avbp (CFD), [63]; Gaussian, Gamess, CRYSTAL, etc. [64]; We are using G9x versions [65]; own code and BLAST, FASTA, SSEARCH, ... [67]; own code [68]; own code, Sobek (written by Delft Hydraulics, interactive windows program). [69]; own code [70]; none [71]; own code [73]; own code (dalton program package, dirac program package) [74]; Gaussian, Amsterdam Density Functionals (ADF), MolCas, Molecule, own code [76]; own code [77]; own code [78]; Gaussian, Gromacs, Own code: pKa calculation, peptide-membrane association [79]; Own code and DOT [82]; own code [83]; MM5 Circulation Model [84]; Gaussian, MSI Software [85];

Web Interface

Yes	40
No	22
Do Not Know	23
No Answer	00

Reasons:

But it would take considerable work to develop [1]

We have not defined one and I am not an expert in this, but I don't see why this should be impossible. [2]

I should be possible to do this, just do not know how to do it.. [3]

No implementation is available [4]

---

compiling through the web sounds ridiculously complicated compared to just opening a few xterms. [6]

---

w e have not needed this user interface. It's easy to set up. [11]

Not at the moment, but interface has been considered [13]

---

it will be highly desirable but at present no one of the application I use have a web interface available [17]

It depends on how smart the web interface is. [18]

We do run our code through a web interface [22]

---

crucial items are:

- *access to a large amount of huge datasets on the corresponding platform. If this cannot be guaranteed it is easier to login directly on the target platform.*
- *code development (if a batch job is submitted on a remote system, what about debugging?)*
- *all (!!) compiler options on the target platform have to be available within the web interface [23]*

If I understand it correctly, to efficiently run an application through a web interface, the application needs to consists of several tasks which do not need to communicate with each other too often. In our application a lot of internal communication is needed between the sub-processes. [24]

Input options required are difficult to standardize, hence control input should come from a plain-text file [25]

---

too much communication [29]

Long batch jobs (days) are unsuitable for a Web interface [30]

---

not yet, we are working on it. [33]

We are happy with the current environment [35]

I never thought about it [36]

---

it is not necessary [41]

Because this interface does not exists, have to be written. [63]

---

this is usually not allowed on most HPC centres%A9 In fact, it is also not necessary [71]

Not yet available but we will implemented this [79]

Problems that require interaction via graphical interfaces mostly have high data volume to be transported/analyzed (e.g 3D-flow field visualization). There are no solutions for this via web interfaces so far. [80]

---

possibly this might be possible in general, but I can not see any advantage in that in comparison with the currently used working approach; there is also interaction with commercial grid generators/postprocessors for CFD data, possibly making it difficult to handle the turn-around-cycle for the program via a WEB interface [81]

I can perceive of no advantage. The codes are ‘research codes’ [83]

Resources Required

Yes	49
No	22
Do Not Know	08
No Answer	06

Resource type required:

For one set of programs I require the ZPL compiler. (see http://www.cs.washington.edu/research/zpl/) the rest is usually available as standard. There is a graphics library written in c with xlib that needs to be built for some of our programs. [2]

OpenMP, HPF and in the future also MPI [3]

MPI [6]

Some programs require Portland F77 compilers [8]

BLAS,PBLAS,BLACS and SCALAPACK. [11]

MPI or shmem, HPF in some cases [13]

---

numerical libraries (NAG, BLAS and LAPACK), etc. [14]

Nag, X11 graphical libraries [15]

Message Passing Interface (MPI) [16]

ANSI C, numerical math libraries (matrix operations), MPI [18]

MPI, Open MP, HPF, free numerical libraries [20]

Some applications use OpenMP or HPF [21]

Depends on the modules used, but typically just MPI [22]

BLAS/LAPACK are desirable [25]

FSPAK, MPI [26]

At the moment many codes used including the Unified Model from UK Met Office rely on the availability of nupdate a very old Cray product for code maintenance. We are not privy to the UKMO's plans to replace this. [28]

Python, Many Open Source packages [30]

Most of the parallel jobs are in MPI for distributed memory machines. A few applications have been written in HPF (lower scalability). Astrophysics n-body code has been written by using a heterogeneous computing resource made by a MIMD and a SIMD platform (APE100) running together. In this case we had to develop a message passing library to communicate data from the MIMD to the SIMD platform. [31]

In some test cases we needed access to OpenMP. [32]

CERN lib [33]

Special compilers and libraries [35]

Standard packages for algebraic operation and fast Fourier transforms [36]

MPI, PVM, OpenMP, HPF, GSL, GNUplot, yacc/bison, flex [38]

---

own code [42]

MPI [43]

Fortran preprocessing, blas routines [44]

LAPACK [49]

MPI [51]

Fits I/O Packages, OpenMP, FFTW [52]

MPI [54]

HPF has been used in some experiments [55]

MPI (message passing interface) [57]

Usually yes: libraries [58]

Nag Fortran SMP library, OpenMP [61]

---

mathematical libraries [62]

Generaly : MPI, scientific libraries (blas, lapack, scalapack), iso fortran compiler. [63]

This depends on the specific grid technology. Our own code could be adapted from its current matlab-fortran hybrid form to something else. [69]

MPI, openMP [71]

MPI [72]

MPI [73]

MPI (otherwise only sequential execution), vendor optimized numerical libraries (to run faster, not critical) [74]

---

compiler with extension pointer CRAY [75]

We definitely need MPI. [76]

MPI [77]

MPI [78]

---

c/c++ compiler (gnu or commercial). Certain libraries such as those implementing, for example, fast fourier transformation. [79]

Some applications require numerical libraries (LAPACK, etc.) with proprietary extensions. [80]

Java [82]

ScaLAPACK, BLACS, MPI, ANSI C, ANSI Fortran, PG Fortran 90 compiler, RSL (free) library based on MPI by Argonne, etc. [84]

Lapack, NAG libraries [85]

Improving Portability

No effort	23
Some effort	45
A lot of effort	17

Changes required for grid compliance:

I think that the ZPL codes should be very portable. I have run the codes on may platforms and there have been few problems. [2]

It is already running on both SGI, SUN, DEC => very little effort is required -> main change is to go from HPF and OMP to MPI... [3]

The code is easily portable to any Unix with Fortran and MPI. [6]

I do not know. [9]

Removing calls to machine-dependent routines and extensive testing. [10]

The code is portable on all machines of interest. [11]

At the moment the code is running on many platforms from Linux workstations to T3E and Fujitsu. Of course big runs cannot be run on workstations [13]

Minor changes [14]

Parallel computing aspects of the code [16]

---

in some cases I would need to fully rewrite the interface and replace for a command line or WEB based interface

computation I do not expect to be a problem, since the code should compile without problems in any standar compiler [17]

---

not many, replacing some calls to matrix operations [18]

Depends on which code [21]

None (but I don't understand what is wrong with gmake ... it is available and consistent on all platforms) [22]

The codes are tuned for a specific hardware. If you change the platform the code will run but probably not very efficiently. How can I implement this in an "easy portable way"? [23]

Our code already runs on various platforms such as the IBM SP2, IBM SP-SMP, Hitachi SR8000, VPP 5000 etc. It doesn't need special external libraries, only MPI needs to be available. [24]

Not much since we try to make the code portable, and have tested it on several computer systems. [26]

Our plans will be very dependent on the plans at the UKMO [28]

Our codes are already portable across many UNIXes [30]

Not much, for the most portable they should be recompiled (most are written in MPI-CH) [31]

Not a lot. Most of the operations are taken care of by GLOBUS. [33]

More parallelism [35]

I do not have a clear cut answer to this question [36]

---

none [38]

Minimal: already fairly portable [39]

---

probably use some communication libraries [40]

I do not know [44]

Very little. [46]

Minor changes if any [53]

At this time, do not know [54]

mostly build and link our fortran c++ library on new architectures, if the pure java version is too slow. [56]

Practically none. We try to produce simple and portable code from the very beginning. [57]

quite limited [67]

No simple answer possible: At the moment there are three candidates: one is a matlab fortran hybrid, the other is pure matlab (but could be replaced if an affordable off the shelf fluid structure interaction model was available). The third is a commercial GUI controlled Windows package, but possibly the developers (Delft hydraulics: www.wldelft.nl) would be interested in Grid technology, they are working on a parallel version at the moment. [69]

random generator [70]

They have already been ported to many platforms, and they are as portable as we want to make them. [74]

Very little. Mainly things like changing the scripts used to run the compiled programs (they are mainly stand-alone commercial packages that need parallelisation programs like MPI of OpenMP). [76]

i/o file referencing [77]

don't know until get grid access [78]

Not very much, code is usually easily portable [79]

Code has been run on a wider variaty of machines. There were some changes for Cray-computers due to incompatibilities in Cray-MPI. That are basically all changes we have to make to the code in order to run on different environments. [81]

The code is already reasonably portable and much effort has already been spent making it so. [83]

Do not expect to need to change much in order to make the code portable. [85]

Converting to Java

Yes	13
No	64
No reply	08

Reasons:

The effort involved is larger than we would currently wish to make for our simulation packages. Some of our archive manipulation software in other projects is currently written in Java [1]

The performance of java is still too poor. [2]

We require high speed and my impression is that Java is not necessarily very efficient for large scale numerical simulations. [3]

Porting Fortran to java sounds like a major effort. [6]

The code we are working on is a results of 15 years of research. It is just too expensive to re-write it. [7]

Inefficient language... [8]

Very large applications, very little knowledge of Java [10]

Most of the code is written in F90 to get optimal speed. Only user interface written in C++. [11]

Efficiency [12]

---

1. Too much work to re-write

2. According to our knowledge Java is not very efficient in number crunching' [13]

Our numerical results must be worked out to obtain a physical description of the system. [14]

Performance hit, platform independence as such is not seen as an issue. [16]

I need high performance and I do not expect to obtain so from an interpreted language

I will not rewrite my code unless forced to do so

Much on the contrary I expect other people to give me reasons (and good ones!) for rewriting code [17]

I don't know enough Java and to rewrite applications in Java takes -I suppose- to much time. [18]

It very difficult to change the code already written [19]

Bloody stupid idea. [21]

We use mainly ANSI C with is also platform independent. The scientists write their application modules in Fortran, and that is unlikely to change in the near future. [22]

---

much too slow!! [23]

We don't have portability problems. [24]

---

1) The existing code comprises several 100,000 program lines

2) Java appears not optimal for heavy computational problems [25]

Complex code using subroutine libraries developed by others. [26]

---

performance is critical [27]

We would only consider it for parts of the codes we use over which we have control. We would need to consider the training implications which can be great for much of our research is done through short turn contracts or PhD students and so there is a large turn over of staff. [28]

---

too much effort; we have to concentrate on the physical problem [29]

Python..Extended answer: Python allows dynamical linking of binary modules with high performance. Performance is an important issue, Web interfaces aren't. [30]

Too long. If required I could think to implement new codes using updated tools (such as java or similar). [31]

We just made the change to C++ as the main programming language. [33]

Not likely to be cost effective [35]

It seems that too much time will be wasted. My aim is to get scientific results as fast and effectively as possible, not to invest in major programming efforts [36]

Too much work, we do not have enough time and people for doing that. [37]

---

1. Portability

2. Portability :-) [38]

Already using Java in places, eg data management, where Flops are not critical. [39]

We need performance and numerical portability. We think that the numerical portability is not sufficiently supported by Java. [40]

Java is slow. [41]

I hope that having this platform would be nice. [42]

Only for interfacing. Not fast enough for the actual calculations [43]

My impression is that the performance is too poor for numerically intensive computing [44]

Too slow, dangerous floating point arithmetic [45]

Java is currently not efficient enough at handing the large quantities of data we generate. Much of the original code is in fortran, so conversion to java will not be easy. [46]

I am to busy with my work to learn new programming language. [47]

Size and complexity of code precludes rewrite [49]

Component nature, Rich inter-working, Portability, Ease of use [50]

---

1) heterogeneous grid computing

2) excellent language [51]

Too slow at present and not compatible with other astronomical systems [52]

It would be a new domain for us. Limited resources. [53]

Computational speed is critical [54]

---

already have a nearly pure java version. [56]

MUCH TOO SLOW! There are many people in our community who even believe C is too slow, and Fortran is the only acceptable language. We are not that strict, and allow C and Fortran - but nothing else. C++ is only permitted for applications which are not time-critical, like evaluations.

Java IS conceivable as a tool to LAUNCH an application, but I guess standard scripts (shell, perl, tcl,...) will do a better job there. [57]

Too many man-month work. [58]

---

speed [59]

No time to do it [60]

---

need for numerical Fortran libraries [61]

Not for all applications, but maybe some. Some are already in Java. [67]

Performance too low. Amount of work involved. Little knowledge of java [68]

Problem 1: speed.

Problem 2: readability for those not familiar with JAVA (read students, maintenance programmers, Ph.D. students etc.): from what I understand, JAVA has a rather idiosyncratic assignment statement (effectively, all object assignments are pointer assignments/assignments by reference). [69]

---

it will take too much time [70]

What about Performance ?! [71]

---

compilers are getting better, we are developing fast communication libraries for Java [72]

Many man-years of coding in our applications. I do not know of any advantage of changing to Java now. [74]

Since speed is essential, I think we have to stick to Fortran or C. But I don't know enough about Java to be absolutely certain. But we mainly want to spend our time with applications and not code development. [76]

---

if it would mean to gain access to HPC resources. [77]

---

have java interface to F90 code [78]

Although, computing is still faster in fortran/c/c++, but we could imaging that certain pieces could be written in Java (especially the user interface). [79]

- expected performance is low

- complexity of programs typically too high for a rewrite [80]

---

performance issues. I did not tried it by myself, but all what I have heard about JAVA is that it is comparable very slow programming language. May be this will change with time. [81]

Most of it is already written in Java. [82]

I do not consider Java to be well suited to scientific computing. F77/F90 are well suited to our needs. C++ would be an alternative. [83]

I do not expect many members of our group to be willing to learn Java. [85]

Part 5 – Infrastructure

Access to local systems

vendor type amount of memory number of processors Usage

IBM top end workstations up to 4Gbyte up to 4 2 proc-years

GRAPE-3A xxxx 5 5 proc-years

IBM SP2 20 Gbytes 16 1 proc-year

CRAY T3E 128Mb per processor 840 50 proc-years [1]

Dell Pc 512 MB 2 Don't know [2]

DEC Cluster 24Gb 24Gb 50% of time [3]

Damicon Alpha Linux 5000 Mb 10 70000 [6]

SGI O2 128MB 1 500

SGI O2 128 1 500

SGI Octane2 256MB 1 800

PC linux-cluster 1GB 12 1500

SUN S450 512MB 2 3000 [8]

SGI (7) IRIS(1) 256MB (5) 1GB(1) ' 2(2)1(5)24h/day

SGI OCTANE(5) 512(1)

IBM 3AT, 3BT 256MB(2) 1(2) 24h/day - [9]

SGI O200 3Gb 6 15000 hours

HP ??? 500Mb 4 10000 hours

Linux Pent/III 128Mb ~10 2000 hours [10]

PC 128 MB 32 3000 h [11]

SUN/SGI workstations 768MB 1 2000

SGI ORIGIN 24 240 [12]

SGI Origin 2000 8 [13]

PC's 1Gb(each one) 2 (each one) 8000h (each one) [14]

SG UNIX 256Mb 1 all

Dec UNIX 512Mb 1 all [15]

3x SGI INDIGO%B2 64-256 MB 1 N/A

2x SGI O%B2 128-256 MB 1 N/A [16]

SGI O2 250Mb 4 ???

SGI 1200 1Gb 32 ???

intel PC 250Mb 5 ??? [17]

SuSE Linux cluster 256 24 (exclusive for our group) cluster of various Linux PC 30 [18]

SGI Origin 4 a lot

SUN work-stations [20]

---

*ITC Linux PC 1Gb 2 24*365 [21]*

---

sgi origin 2000 16GB 64 fully used [22]

CRAY T3E-600 128 MB / PE 512 -

CRAY T3E-1200 512 MB / PE 512 35000

CRAY T90 2 GB 10 - [23]

IBM SP2 500Mb per node(=processor) 128(prod q) 525k [24]

SGI O200 (2x) 4GB 8 100% [25]

IBM rs6000 .5 GB 1 >8000

IBM rs6000 1 GB 1 >8000

IBM rs6000 1 GB 1 >8000

IBM rs6000 1 GB 1 >8000

IBM rs6000 1 GB 4 >8000 [26]

Cray T3E 8GB 128 dedicated [27]

SGI Origin ? 6 large

Suns variable ? ? Every day

PC clusters variable ? ? testing [28]

Fujitsy VPP300 16 Gb 8 [29]

---

*Compaq Alpha 512 MB 168 365*24*

*AMD Athlon 512 MB 10 365*24 [30]*

IBM SP2 32 GB 32 TBMD

CRAY SV1 16 GB 16 CP Mol.Dyn.

Beowulf EV67 16 GB 16x2 TBMD, CP

MIMD+SIMD sparc 4GB(MIMD)+2GB(SIMD) 8x2(MIMD)+512(SIMD) astrophys. [31]

---

self-made cluster 256MB/proc 24 10% [32]

Mixed 300-1000MHz 128-512 Mbyte 1 per machine 40 in total 100% [33]

SGI Origin200 256 Mb 2

SGI O2 192 Mb 1

SGI Indigo2 192 Mb 1 [34]

SGI 200 512 MB 2 Intensive

PC-Linux pentium 1GB 2 Intensive

CEPBA + CESCA resources [35]

HP 720 1 120 [37]

Digital Alpha servers

Beowulfs [38]

Compaq Alpha 512 MB/processor 18 100% [39]

---

n/a Intel 512MB 1 2000 [40]

---

look at: www.man.poznan.pl [41]

---

n/a PC 128 MB 2 2400 hours [42]

SGI Octane 512 MB 1 100%

AMD LINUX 256 MB 6 75% [43]

SGI ORIGIN 15.5 + 20 GB 64+40 30% of available time [44]

Compusys Digital 128MB 1 x 4 machines 10-15,000

EV56

Silicon R12000 1.5GB 6 ~20,000

Graphics

Digital EV56 192MB 1 x 2 machines ~100 [46]

IBM power3 16 G 16 [48]

IBM SP 256MB per node 48 4000 hrs

COMPAQ DS20 512MB 2 4000 hrs

Desktop PC 512MB 1 8000 hrs [49]

Fuitsu AP3000 20G 84 General

Compaq Quaadrics 50G 32 CFD

Alpha

PC Cluster 16G 16 Comp Sci [50]

Compaq AlphaCluster 8 GB 16 40 [51]

SGI Origin 3800 40 GBytes 40 100%

Compaq Alpha 1.5Gbytes 4 100% [52]

---

8 x HP 780 ??? 4 ??? [53]

---

various PC 500 - 1000 MB 60 continuously [54]

IBM SP SP 48

PC CLUSTER 32 [55]

SGI origin2000 24GB 128 40%

SGI onyx2 2GB 4 50%

SGI O2 128MB 1 90%

Sun Blade 1000 4GB 4 100%

various sun and linux workstations [56]

DEC alpha 64 MB ... 256 MB ~70 (ws cluster) cont. [57]

SGI Cray Orig. 2000 128 8000

+ others [58]

SUN ultrssparc II 2 Gbytes 4 [59]

DEC ALPHA 1Gb, 2 ,full

DEC ALPHA 1Gb 2 full

SGI Origin200 1Gb 2 full [60]

SUN ES6500 30GB 30

SUN

various desktop [61]

Cray T3E 512 >200000

Compaq Alpha Cluster 96 ? [62]

SGI O2000 16 GB 32 120 000

SGI PowerChallenge 1,2 GB 8 6 000 [63]

AMD Thunderbird 1000MHz 256Mb 1 100%

HP - 512Mb 2 40%

SP2 - - 44 -

SGI 64 - [64]

PC 12bMB 1-2 never stops [65]

SGI Origin x GB 128 3000? [67]

SGIs - 100-2000Mb 1-4

Usage: Developement, testing, result evaluation, Terminal access to super computers. [68]

PC-compatible 256MB 1 Pentium II 300 workstation

CRAY-T3E none

CRAY_J90SE none [69]

PC 128 MB 1 960 [70]

---

home made PPro200 128 128 100000 [72]

Self-made Linux cluster 1GB 4 [73]

---

sgi O2000 16G 32 [75]

SGI Origin 2000 192 Mb pr. proc 128 20000 [76]

---

*Pentium 133-450 16-128 1 6*365 h/year [77]*

SG O2 1GBeach 20 8h/day [78]

PC/linux 256 1 Desktop

Compaq Alpha 256 6 (cluster) Computing

SGI IRIX unix 256 1 Graphics [79]

We have several self-made clusters of workstations available which are mainly/exclusively used by the SFB393 department:

CLiC: 512MB/node, 528 CPUs

Roulette: 512MB/node, 72 CPUs

Athlon Cluster: 512MB/node, 12 CPUs [80]

CLiC: 512MB/node, 528 CPUs

Athlon Cluster: 512MB/node, 12 CPUs

Alpha Cluster with Myrinet: 512MB/node, 8 CPUs

all clusters self-made, different vendors on delivery of parts (Qunat-X, MegWare, etc.) [81]

Compaq XP1000 (6off) 128Mb 1 100%

SGI Octane 256Mb 2 100%

Quadrics/

Compaq Alpha 2Gb/node 32 30000 hrs [83]

IBM SP 10GB 20

cluster

Beowulf 2GB 16

cluster

Beowulf 8GB 32 [84]

Dec/Alpha 667MHz ? 2 100%

SGI Indigo2 R10K 512MB 1 100%

SGI Indy R5000 256MB 1 100%

Various Linux PCs PIII 256MB 8x2 100% [85]

Access to national systems

vendor type amount of memory number of processors Usage Provider

---

Not using at present [1]

SUN the lomond cluster, but I only used it a little bit because the batch job policy did not allow to run jobs for long enough times. [2]

SGI R12000 64Gb 128 72,000 UKAFF [3]

---

cray t3e 100 64 30000 NSC

ibm sp2 100 64 20000 pdc [5]

Cray T3E 65000 Mb 512 40000 CSC Finland

SGI Origin 100 Gb 128 30000 CSC Finland [6]

T3E 45.6 GB 256 48.000 NSC

SGI 96 GB 96 24.000 NSC [7]

SGI Origin2000 huge... 128 large number CSC [8]

SGI ORIGIN 128 ? 1000 NSC

? T3E 128 ? 3000 NSC

IBM SP 64 ? 500 HPC2N

Fujitsu VX 128 ? 250 PDC [9]

IBM SP2 12Gb 44 20000 CESCA

HP V2500 8Gb 16 10000 CESCA

SGI O2000 8Gb 64 5000 CEPBA [10]

IBM SP 128 MB 64 8000/64 HPC2N

IBM SP 256 128 4000/128 PDC

SGI T3E 128 256 10000/256 NSC

SGI 3800 1028 96 1000/96 NSC

Beowulf 512 32 4000/32 NSC [11]

CRAY t3e-1200e 816 200,000 NERC

SGI ORIGIN2000 128 20,000 NERC [12]

T3E 128 MB/node 512 CSC.fmi.fi, operational work, development

SGI Origin 2000 ~100 backup for operat. [13]

IBM SP2 0.5 Gb CESCA

SGI O2000 1.Gb CEPBA

HP V2500/N4000 0.5 Gb CESCA

about 30.000 h/year [14]

SG UNIX Don't know don't know 4000 CEPBA [15]

SGI ORIGIN 2000 160 GB 128 20000 CSC [16]

IBM SP2 ??? 64 ??? CESCA [17]

HP V-2500 32GB 30 60% ZDV Mainz

*CRAY T3E 512 3*7500*12 (PE hours) HLRZ Juelich*

CRAY T3E 512 500000 HLR Stuttgart [18]

SGI SGI Origin 8-32 a lot UNI-C

SUN 4-16 DTU

SP2 8-32 not now

IBM SMP 2 nodes 8 procs per node not now

Fujitsu vector 1 not now [20]

---

*Cray T3e 64*128 128 500,000 EPCC [21]*

---

hitachi SR-8000 don't remember (lots) LRZ

cray T3E ?? 256 ZIB

cray T3E ?? 766 RZG [22]

IBM SP-3 130 GB 256 Benchmarks Karlsruhe

Hitachi SR8000 F1 928 GB 112x8 Benchmarks Munich (LRZ) [23]

Hitachi SR8000 +/- 4800Mb per node(=8proc.) 512(prod q) 256k [24]

SGI O3000 96GB 96 84000 NSC [25]

At the moment we do not have access to extern computer resources. Previously we had access to an IBM SP installed at UNI-C. [26]

Compaq Alpha 1GB 8 dedicated U. Glasgow

APE APEmille 2GB 128 dedicated U. Swansea [27]

Cray t3e-1200 256MB per proc >800 ~700K pa CSAR

SGI Origin 2000 128 small CSAR

O3000 variable 128 small CSAR

Fujitsu VPP300 2GB per proc 8 large CSAR

Cray t3e-1200 ? ? small UKMO [28]

NEC SX 5 64 Gb 32 Uni Stuttgart [29]

None [33]

SGI Origin2000 64 20000 h CEPBA

HP V2250 16 15000 h CESCA

HP V2250 8 8000 h CESCA

IBM SP2 44 6000 h CESCA [34]

---

none [35]

SGI Origin 2GB 2 50 HPC ICCC [37]

CRAY T3E

Origin 2000

IBM SP2 [38]

---

none [39]

---

n/a SGI 2GB 2 2000 ICCC [40]

SGI

Cray

HP

SUN [41]

SGI Origin 2GB 2 1000 hours [42]

Cray T3E 128 MB / node 200 2000h/month NSC

SGI Origin 3800 512 MB/node 100 1000h/month NSC [43]

IBM SP2, SP2-NH 115GB 300 45000h PDC

SGI Origin3800 96GB 96 30000h NSC [44]

CRAY T3E 256MB / processor 788 0 CSAR [46]

Nec SX5 IDRIS

IBM power4 (September2001) 256 IDRIS [48]

HP 780 ??? ??? [53]

Cray T3D unknown 1024 1 000 FZ Juelich [54]

NATIONAL FACILITIES [55]

---

unknown [56]

Cray T3E ? ~800 1 Mio MaxPlanckSoc [57]

---

many [58]

-SGI- Origin2000 3Gb, 4, full [60]

Fujitsu VPP5000 ? 31 5000 Meteo-France

Compaq AlphaServer ? 256 40000 CEA

SGI O2000 ? 256 100000 CINES [63]

SGI Origin-3800 160 Gb 160 Computing Res. council

SGI ORIGIN-2000 24 Gb 128 Computing University [68]

CRAY Origin2000 none SARA

IBM RS/6000 SP none SARA

SGI Origin 3800 none SARA

SARA=Stichting Academisch Rekencentrum Amsterdam: www.sara.nl [69]

SGI Challange 1 GB 12 1500 Poznan MAN [70]

NEC vector ~48 GBbytes 32 10%A9000 RUS

VPP vector ~GB/proc 16 10%A9000 RRZE

*Hitachi scalar/pseudo vector 1 TByte 8*100 10%A9000 LRZ [71]*

---

home made PPro200 128 24 100 Univ.Amsterdam

home made PPro200 128 24 100 TU Delft

home made PPro200 128 24 100 Leiden Univ [72]

Hewlett-Packard Shared Memory 48 [73]

Fujitsu VPP5000

Cray T3E [75]

SGI Origin 3800 1 Gb pr. proc. 160 40000 University of Trondheim

HP Superdome 2 Gb pr. proc. 44 10000 University of Oslo [76]

Silicon Octane ? 8 not yet in use

Graphics [77]

CRAY T3E/1200E 870 150khours CSAR [78]

Compaq Unix Gigabytes 99 (cluster) computing www.csc.fi

IBM Unix Gigabytes (?) computing www.csc.fi

SGI Unix Gigabytes 16 computing www.csc.fi

DEC/Compaq Unix Gigabytes 8 computing www.csc.fi

Check website www.csc.fi for exact details [79]

Sorry, I don't have figures on that available at the moment. [80]

Cray-T3E 128MB/node 64 3000 CPU-hours/year University of Technology TU Dresden [81]

Cray T3E 256Mb/node 788 30,000 CSAR [83]

Access to external systems

vendor type amount of memory number of processors Usage Provider

---

Collaborative work at EPCC through the Virgo Consortium [1]

Cray t3e check CASR 256 100000 CSAR

SGI Origin check CASR also 256 now? 500 more CSAR in future [2]

SGI Origin ? > 1000 5000 NCSA, USA [6]

SGI T3E 256 MB 816 20000/880 CSAR UK [11]

Fujitsu(s) development ECMWF(England) [13]

---

none [18]

SUN 1-16 rare TRACS

CRAY T3E up to 128 rare TRACS [20]

---

*Cray T3e 800*128 800 1,401,600 Max Plank, Ge [21]*

---

sgi origin 2000 at NCSA

ibm sp2 at SDSC

NT supercluster at NCSA

linux cluster at AHPCC

compaq teramachine at PSC

linux clusters at NCSA [22]

---

no access [23]

- none - [25]

Variable, used by some researchers under EU contracts [28]

CERN [33]

---

none [35]

PARSYTEC
IBM SP3 [38]

---

none [39]

---

see above [42]

NONE [55]

---

unknown [56]

Varies [58]

Fujitsu VPP5000 ? 117 5000 ECMWF [63]

HP - 512Mb 2 40% Univ. Barcelona
SP2 - - 44 - CESCA/CEPBA
SGI - - 64 - CESCA/CEPBA [64]

PC 100x512MB 100x2 ? NIMR, UK [67]

---

none [78]

Sorry, I don't have figures on that available at the moment. [80]

---

non for the moment; but currently there is also no need for that [81]

Cray T3E 128x256MB+128x128MB 256 10000 CINECA [85]

Bandwidth

```
 Internal: 200 Mbits/s  External: 1000 Mbits/s [1];
 Internal: - Mbits/s    External: - Mbits/s [2];
 Internal: 100 Mbits/s  External: 100 Mbits/s [3];
 Internal: - Mbits/s    External: - Mbits/s [4];
 Internal: - Mbits/s    External: - Mbits/s [5];
 Internal: 100 Mbits/s  External: 100 Mbits/s [6];
 Internal: 500 Mbits/s  External: 500 Mbits/s [7];
 Internal: 100 Mbits/s  External: 100 Mbits/s [8];
 Internal: - Mbits/s    External: - Mbits/s [9];
 Internal: - Mbits/s    External: - Mbits/s [10];
 Internal: 10 Mbits/s   External: 100 Mbits/s [11];
 Internal: 1000 Mbits/s External: 34 Mbits/s [12];
 Internal: - Mbits/s    External: ~200 Mbits/s [13];
 Internal: 100 Mbits/s  External: 100 Mbits/s [14];
```

```
Internal: - Mbits/s    External: - Mbits/s [15];
Internal: 100 Mbits/s  External: 10-100 Mbits/s [16];
Internal: 100 Mbits/s  External: - Mbits/s [17];
Internal: 100 Mbits/s  External: 1000 Mbits/s [18];
Internal: 10 Mbits/s   External: 1 Mbits/s [19];
Internal: - Mbits/s    External: - Mbits/s [20];
Internal: 100 Mbits/s  External: 10 Mbits/s [21];
Internal: - Mbits/s    External: - Mbits/s [22];
Internal: 100 Mbits/s  External: - Mbits/s [23];
Internal: 100 Mbits/s  External: - Mbits/s [24];
Internal: 100 Mbits/s  External: 100 Mbits/s [25];
Internal: 100 Mbits/s  External: 2 Mbits/s [26];
Internal: 100 Mbits/s  External: 100 Mbits/s [27];
Internal: - Mbits/s    External: - Mbits/s [28];
Internal: 10 Mbits/s   External: 100 Mbits/s [29];
Internal: 100 Mbits/s  External: 100 Mbits/s [30];
Internal: 32 Mbits/s   External: 8 Mbits/s [31];
Internal: 155 Mbits/s  External: 32 Mbits/s [32];
Internal: 100 Mbits/s  External: 1000 Mbits/s [33];
Internal: 100 Mbits/s  External: 100 Mbits/s [34];
Internal: 100MB Mbits/s      External: - Mbits/s [35];
Internal: - Mbits/s    External: - Mbits/s [36];
Internal: - Mbits/s    External: - Mbits/s [37];
Internal: 2 Mbits/s    External: 100 Mbits/s [38];
Internal: 100 Mbits/s  External: 612 Mbits/s [39];
Internal: - Mbits/s    External: - Mbits/s [40];
Internal: 100 Mbits/s  External: 75 Mbits/s [41];
Internal: - Mbits/s    External: - Mbits/s [42];
Internal: 10 Mbits/s   External: 100 Mbits/s [43];
Internal: 100 Mbits/s  External: - Mbits/s [44];
Internal: 100 Mbits/s  External: 50 Mbits/s [45];
Internal: - Mbits/s    External: 100 Mbits/s [46];
Internal: - Mbits/s    External: - Mbits/s [47];
Internal: 100 Mbits/s  External: - Mbits/s [48];
Internal: - Mbits/s    External: - Mbits/s [49];
Internal: 100 Mbits/s  External: 10 Mbits/s [50];
Internal: 2 Mbits/s    External: - Mbits/s [51];
Internal: 1000 Mbits/s External: 150? Mbits/s [52];
Internal: 100 Mbits/s  External: 100 Mbits/s [53];
Internal: 100 Mbits/s  External: 10 Mbits/s [54];
Internal: 100 Mbits/s  External: 100 Mbits/s [55];
Internal: 100 Mbits/s  External: 155 Mbits/s [56];
Internal: - Mbits/s    External: - Mbits/s [57];
Internal: 100 Mbits/s  External: - Mbits/s [58];
Internal: 100 Mbits/s  External: 1-10 Mbits/s [59];
Internal: - Mbits/s    External: - Mbits/s [60];
Internal: 100 Mbits/s  External: 1000 Mbits/s [61];
Internal: 150 Mbits/s  External: 150 Mbits/s [62];
Internal: 100 Mbits/s  External: 2 Mbits/s [63];
Internal: 10 Mbits/s   External: - Mbits/s [64];
Internal: - Mbits/s    External: - Mbits/s [65];
Internal: 6 Mbits/s    External: 1 Mbits/s [66];
Internal: 100 Mbits/s  External: 100 Mbits/s [67];
Internal: 10 and 100 Mbits/s External: 100 Mbits/s [68];
Internal: 10 Mbits/s   External: 10 Mbits/s [69];
Internal: 0,128 Mbits/s      External: - Mbits/s [70];
Internal: 100 Mbits/s  External: - Mbits/s [71];
Internal: 100 Mbits/s  External: 30 Mbits/s [72];
Internal: 10 Mbits/s   External: 10 Mbits/s [73];
Internal: 100 Mbits/s  External: - Mbits/s [74];
Internal: 10 Mbits/s   External: 2 Mbits/s [75];
```

```
Internal: - Mbits/s    External: - Mbits/s [76];
Internal: 10 Mbits/s   External: - Mbits/s [77];
Internal: 100 Mbits/s  External: 2Mbits/s Mbits/s [78];
Internal: - Mbits/s    External: - Mbits/s [79];
Internal: 10-100 Mbits/s     External: 10-100 Mbits/s [80];
Internal: 100 Mbits/s  External: 100 Mbits/s [81];
Internal: 10 Mbits/s   External: 10 Mbits/s [82];
Internal: 100 Mbits/s  External: 100 Mbits/s [83];
Internal: 100 Mbits/s  External: 10 Mbits/s [84];
Internal: 10 Mbits/s   External: 2 Mbits/s [85];
```

Part 6 – Security and Services

Quality of Service

Training (in a class room environment):

Not important	23
Useful	35
Important	15
Very important	12

Training (distance learning):

Not important	13
Useful	39
Important	22
Very important	11

Support (answer technical queries or resolve difficulties):

Not important	03
Useful	16
Important	36
Very important	30

Remote visualisation tools and support to analyse data:

Not important	17
Useful	32
Important	24
Very important	12

Long term secure data storage:

Not important	24
Useful	28
Important	18
Very important	15

Ease of use:

Not important	07
Useful	12
Important	32
Very important	34

Guaranteed turn around time:

Not important	10
Useful	24
Important	37
Very important	14

Security of code, input and output data:

Not important	15
Useful	24
Important	22
Very important	24

Availability and reliability:

Not important	04
Useful	08
Important	34
Very important	39

Other QOS issues:

The support should be a high level support and not a secretary. The complete system (which supercomputers are connected, which shares are available, how long are the queues, ...) should be transparent for the end user [23]

The administration, accounting etc must not be too onerous . [28]

---

probably not [42]

---

private databases [70]

Availability of good quality documentation [83]

Presence of Firewall

Yes	52
No	27
Do Not Know	06
No Answer	00

Access Permissions Control

Would you like to have a control over the level of permissions you grant to other Grid components, such as scheduling brokers, bandwidth reservation brokers etc.

Yes	43
No(*)	08
Do Not Know	33
No Answer	01

(*) I want security issues like proxy credentials to be transparent to me.

Other permission issues:

I would of course want to be sure that noone could use this service to gain access to any restricted areas of my computer (like private files, in store IRC's e.t.c.). [2]

I assume this will be the answer, though I will not be the one that take that decision. Also because we still will like to make most of the smaller development and experiment work locally with all the advantages that this have to data access and data analysis. [3]

Security.... [8]

Ease of use [10]

I see sufficient control and adequate training as the two most important elements security. [16]

I am suspectfull to any application from outside my group that access my computers. There is a lot of valuable data on them and It is my responsibility to look after that and protect it from potential damages [17]

Don't care how it is done [21]

Security and grid computing is very hard to combine. If you increase security you decrease accessibility and have another source of unexpected bugs. My opinion is to have as much security as necessary but do not annoy the user with too many security checks. [23]

---

not sure of the answer since we're likely to have a complex situation involving users with different access privileges [27]

Resources at the University are shared and administered by very different agencies (Department, University, HEFC, Research Councils) and so there needs to be some control to satisfy these broad and possibly conflicting requirements. [28]

---

you never know what happens when you are not in control [29]

Because autonomy is important for our tests. [32]

We would like to be able to reserve local resources for exclusive use. That should be dynamically changeable. [33]

Simplicity [38]

Own resources are expensive; we are not a service provider. [39]

We do not have a deep knowledge of this stuff. [40]

There is no secure binaries, daemons, brokers, whatever... [41]

Require safety against hackers, but also need flexible access possibilities for users at different locations. [43]

There can be bottlenecks in the department's accesses to the outside world. We are fed through University college's link to Janet and there can be periods of high use currently. It might be useful to restrict grid access during such times. [46]

Access Control required [51]

Again not able to give a straightforward answer - some of applications need transparent security others I would like to control access. In particular bandwidth reservation for some applications [52]

No experience [53]

Security issues [54]

Our system people would, most probably, not be willing to allow access from a remote system without them carefully monitoring and knowing what is going on, in any respect. We had quite some problems with unauthorised breaking into our systems. Furthermore, my feeling of our machines is in essence that of private property, so I must be given the right to turn off external access, if, for instance, the load becomes too heavy, etc. But maybe I misunderstood the question? [57]

I am uncertain to what extent my group can use Grid systems [68]

Uncertainty about the nature of the system [69]

There is a need for controlling what we give to other users. In other words: we would like to be able to use our resources for ourselves in the moment we need it. [72]

---

for better management of my resources. [77]

We don't have yet a grid system. It is something we are looking into and we have to see how things are to be done. [79]

We sometimes need to run experiments without any external disturbances. [82]

Mainly security issues and possible interference with the work of local users [83]

Part 7 – Future Needs

New Problem Areas

We would globalise our current simulation effort and the distribution of simulation products to the astronomical community.

We would work within the Global Virtual Observatory [1]

Scaled up versions of our simulations are of course easiest to implement. Right now I can visualize small simulations in real time. It would be a qualitative change if I could do so with large simulations also. But that would require a serious improvement in computation (maybe a factor 100-1000) and doing this remotely will also add the problem of bandwidth for the GUI. [2]

In 3D resolution is the main problem. Double the grid size and you have about 16 times longer run times and 8 times large snapshots. More realistic experiments requires higher resolution and more sophisticated physics. => a combination of these two will be used to get the experiments more realistic. [3]

---

most scaled up versions of current problems [5]

Use quantum mechanical models to study irradiation effects, which is partially a scale-up of current problems, partially would open up entirely new possibilities. [6]

Already with the current computer power, I have been able to work on some realistic systems. If the constrains were eliminated, it would be possible for me to study more real systems, and even get into the field of bio-photonics. [7]

Modelling the cooperation of proteins and other biological systems using both molecular mechanics and quantum chemistry.

This problem is partially a scaled up version, but only partially... [8]

I think they would be scaled up versions of my present research efforts. [9]

Computational chemistry struggles to study systems as close as possible to real systems, with methods as accurate as possible. An increase in computational resources will allow to compute larger systems with more accurate methods. [10]

Scale up to model large biochemical molecules and nanostructures. [11]

We are currently planning a 1/12 degree global ocean model which with current resources we will only be able to run for 10-20 years. It would be desirable to be able to routinely run ensembles of such models or to run one such model for several hundreds of years. This is a simple scaling of resources.

New problems would involve combining high resolution ocean models with other parts of the climate system - atmosphere, sea-ice, biology. All current climate modelling uses only low resolution ocean models and misses much of the basic physics of the ocean and therefore the coupling. [12]

Partly they are scaled upwards. New things are

1. better assimilation of all kind of meteorologiacl data

2. ensemble (a type of MonteCarlo) forecasting [13]

Would be possible to, consider more complex systems. [14]

Computational simulation of steering forces with ships as an extensions to the current flow simulations with ships in steady course. Simulation of ships with propulsors. [16]

I will keep doing the same things on larger systems. On some situations I will use better computational methods that now I can not afford [17]

Today, the simulation of atomistically realistic models is only feasible on very small time and length scales. Interesting questions on larger scales (like 100nm,microseconds) are accessible in coarse grained models and larger length scales are hardly accessible in models which keep some degree of molecular structure. If computational constraints were eliminated, we could study the correlation between atomistic structure and macroscopic properties (eg, how does the degree of branching of a polymer influences the mechanical properties like toughness). This seamless modeling over many decades in time and length scales would have important impacts on how materials are developed, but poses enormous challenges in terms of computing resources. [18]

Refine the grid discretization of the models

More 3-D computations

Solving inverse and optimization problems in air pollution modelling [20]

Scaled up and new. [21]

---

more accurate (larger), more interactive, more parameter studies. [22]

---

in our application fields (QCD, MD) it is more or less scaled up. [23]

The new problems I would simulate if there were no computational constraints would be scaled up versions of current problems. [24]

Solving larger problems then to day where CPU speed and memory puts limits on the size of problems we can handle. This would be partly by up scaling of existing software, partly by development of new algorithms and software. [26]

---

scaled up versions [27]

New problems would involve coupling model components together easily and transparently across multiple platforms as well as developing better coupling of models to data. [28]

---

problems at higher Reynolds numbers and multidisciplinary problems; new and scaled up problems [29]

We have lots of scientific problems that can be addressed by more powerful computers. We mainly need many more CPUs, and they should be as fast as economically feasible. [30]

No see earlier answer. [33]

More realistic problems, closer to the chemical industry [35]

They have larger dimensions, models have finer resolution, more complicated grids and smaller times steps [37]

They will be simply new scaled up versions [38]

Wider areas of astronomy, such as pulsar searches. The methods would be scaled-up and adapted versions of current methods. The scientific field is very different. [39]

We would like to run things in parallel and on a grid. We would like to compute mixtures of chemicals with more components, rock-solution contact problems. [40]

---

scaled up versions of current problems [41]

More detailed models, smaller discretization parameters, fracture flow and, moving boundaries and meshes [42]

Sclale up investigated system sizes to more realistically model the physical problems at hand.

New possibilities concerning on-the-fly dynamical calculations which by todays standards simply are not possible. [43]

---

(1) Decrease of turn-around time makes it possible to make studies of trends of various properties and to make searches for mterials with desired properties.

(2) The abilty to study systems with lower symmetry, including defects, which are important for instance, in catalysis.

(3) The study of complex organic molecules for the realization of molecular nanomachines [44]

Qauntum Computations [45]

Expansion of the current work to study more members of the protein family and other relevant proteins, for longer and running many sets of unfolding runs as opposed to the 4-6 currently undertaken.

This is a combination of new and scaled up work. [46]

Rather scaled up of current problems, with modifications enabled by faster CPU. [47]

---

new problems [48]

More realistic computer 'experiments', closely matching true experimental conditions. Mostly, scaled-up versions of current problems, but applied to systems that are unattainable at the present. [49]

Turbulent Flow CFD

stochastic Optimisation

PET image reconstruction [50]

Our current problems are computationally bound. [52]

Rapid prototyping:

Matlab/ Simulink to parallel/ distributed C code

Support for FPGA (field programmable logic) design: access to

- VHDL compilers.

- simulators

- Place and root tools

The lack of access to these tools is one of the major limitations for the microelectronic design in the eastern europe. [53]

Larger scale, more LES [54]

Scaled up appliactions, more work on middleware components [55]

---

scale up the problem [56]

This question is purely academic since the computational needs in computational statistical physics vary over many orders of magnitude. For decades, we have defined the complexity of the problems we have studied to be those which live just at the borderline of being doable, and we will most likely continue to do so. We would of course love to do protein folding, but that would, in our opinion, require MANY orders of magnitude. Applications in the more foreseeable future would include complex interplay of flow, charges and polymer conformations, or the structure of DNA-amphiphilic "lipoplexes".

This approach (i.e. defining projects in terms of CPU availability, which is the only feasible philosophy at the high end), also prevents me from answering question 3 - the needs, are, loosely spoken, simply infinite. [57]

We receive 30 man-month of users yearly, so all variety of problems may come up. [58]

3-Dimensional dynamic simulations (scale up versions of current problems) [60]

Longer time and length scales would be enabled. [62]

Large CFD problems resolution

Cooperation between Climate modeling research teams

Access to large numbers of processors for scalability studies [63]

Some of them are going to be scaled up versions of current problems but most of them are going to be "untouched" fields of computational chemistry. [65]

---

scale up, do more exhaustive analyses, include more data

possibly new problems [67]

Increase in resolution, number of components and processes

Scale up current problems [68]

Canal system control of large systems with water quality (scaled up version of current problem).

Behaviour of flow control structures: would jump from semi-analytical models to proper hydrodynamic models.

Resonance in channels: currently limited to smooth long waves. [69]

Scaled up problems in Turbulence. Also: Applying new algorithms if appropriate [71]

The size of the problem could certainly grow big, i.e. scaled up versions of current problems. A further possibility could be the coupling of continuum modeling with microscopic simulations. [73]

Scaled up versions of current problems (larger molecules can be investigated, thus more real-life problems can be attacked). [74]

---

scaled up versions of current problems? [75]

Quantum chemical calculations of real-size models and subsequently studying real-time reactions by means of molecular dynamics. This corresponds to a scale up of current problems. Within the same class, is very accurate quantum chemical calculations of photoelectron spectra. [76]

---

scaled up versions of already studied problems and more complex systems (more components and processes) [77]

---

higher resolution of coastal processes;

increased complexity of biogeochemical processes;

long climate runs;

predicting coastal water quality;

all these are long term goals of current work, but existing parameterisations (e.g. of pollutant transfer, biological interaction) are coarse. would like to resolve time/space spaces over which processes active, or at least derive realistic parameterisations. [78]

Probably scaled up versions of current problems, since questions there we not yet answered. [79]

There are too different opinions about this to give a concise answer here. [80]

Main problems are currently not in the availability of computing resources but in the way different essential tools are interacting with each other. A key component in CFD is still the ease of grid generation and the visualization of very large data sets of CFD data. For the moment we are able to compute larger CFD problems than we are able to process in visualization. But this might be an issue of availability of money for this research to our institution. [81]

Computations of more complex flow configurations with a much larger number of grid nodes. In depth investigations of DNS and LES results to devise improved sub-grid models. [83]

New problems and new computational approaches. [84]

Increse of the available computational resources would enable us to study larger same/similar systems, trac system-behavior in longer time-scales but also study more complicated systems that cannot be effectively simulated with the currently available resiources. [85]

Application Areas

The same or new versions of third party codes as today:

	In 5 Years	In 10 Years
1-10%	15	13
11-20%	07	12
21-30%	06	08
31-40%	06	04
41-50%	06	02
51-60%	07	04
61-70%	03	01
71-80%	01	01
81-90%	01	02
91-99%	00	01
100%	01	02

New third party codes:

	In 5 Years	In 10 Years
1-10%	17	11
11-20%	10	10
21-30%	14	10

31-40%	04	08
41-50%	05	03
51-60%	04	03
61-70%	02	02
71-80%	00	03
81-90%	01	02
91-99%	01	01
100%	02	06

The same in house codes as today:

	In 5 Years	In 10 Years
1-10%	11	18
11-20%	11	04
21-30%	04	05
31-40%	03	01
41-50%	03	02
51-60%	03	00
61-70%	01	02
71-80%	00	00
81-90%	00	00
91-99%	00	01
100%	01	02

Slightly rewritten or enhanced versions of current in house codes:

	In 5 Years	In 10 Years
1-10%	12	15
11-20%	13	13
21-30%	08	10
31-40%	11	02
41-50%	08	09
51-60%	05	03
61-70%	06	01
71-80%	02	02
81-90%	02	02
91-99%	00	00
100%	02	04

New codes written in house from scratch:

	In 5 Years	In 10 Years
1-10%	13	06
11-20%	12	14
21-30%	14	11
31-40%	05	08
41-50%	05	05
51-60%	08	02
61-70%	02	06
71-80%	02	07
81-90%	01	02
91-99%	00	01
100%	03	07

CPU Integer Performance:

x10 in 5 Years	x100 in 10 Years

Cannot tell	31	37
Much too small	06	01
Too small	20	16
Sufficient	25	27
Overly sufficient	03	04

CPU Floating Point Performance:

	x10 in 5 Years	x100 in 10 Years
Cannot tell	24	32
Much too small	16	08
Too small	24	24
Sufficient	20	20
Overly sufficient	01	01

Primary Memory Capacity:

	x10 in 5 Years	x100 in 10 Years
Cannot tell	26	32
Much too small	04	03
Too small	20	13
Sufficient	33	34
Overly sufficient	02	03

Short Term Storage:

	x10 in 5 Years	x100 in 10 Years
Cannot tell	29	36
Stuch too Stall	02	01
Too small	12	07
Sufficient	38	36
Overly sufficient	04	05

Long Term Storage:

	x10 in 5 Years	x100 in 10 Years
Cannot tell	28	35
Stuch too Stall	03	01
Too small	15	07
Sufficient	33	33
Overly sufficient	06	09

Application Performance

The bandwidth and latency performance seems to grow on a slower-than-Moore's law curve thus not keeping pace with the processor and memory performance development. In a future perspective, what is your opinion as to how this will affect your application's performance:

Memory bandwidth and latency:

Cannot tell	27
Negligible impact	01
Small impact	11
Large impact	29
Very large impact	17

Disk bandwidth and latency:

Cannot tell	28
Negligible impact	07
Small impact	28
Large impact	21
Very large impact	01

Network bandwidth and latency:

Cannot tell	30
Negligible impact	04
Small impact	19
Large impact	20
Very large impact	12

Single or Parallel Processing

Please indicate your view on the importance for you on future needs for the following:

Single processor performance:

	In 5 years	In 10 years
Not important	11	13
Slightly important	14	16
Important	32	31
Very important	28	25

Access to moderately parallel systems, less than 128 processors:

	In 5 years	In 10 years
Not important	07	08
Slightly important	11	10
Important	34	35
Very important	33	32

Access to moderately parallel systems, more than 128 processors:

	In 5 years	In 10 years
Not important	19	19
Slightly important	21	14
Important	21	21
Very important	24	31

Programming Languages

The importance to future work of the kinds of parallel programming models users will be interested in the future, 5-10 years.

OpenMP:

Not important	28	32
Slightly important	14	09
Important	34	34
Very important	09	10

Message passing:

Not important	23	26
Slightly important	04	04
Important	33	31
Very important	25	24

SHMEM:

Not important	52	57
Slightly important	12	13
Important	14	10
Very important	07	05

Co-array FORTRAN/UPC:

Not important	63	66
Slightly important	15	13
Important	07	06
Very important	00	00

HPF:

Not important	64	64
Slightly important	11	10
Important	09	08
Very important	01	03

Final Comments:

On the previous question I believe that other concepts that allow users to write parallel programs directly which will then use any of the above communication packages will be much more important. This concept is the basis of ZPL (http://www.cs.washington.edu/research/zpl/) and this or something like it will have more future than any of the above communication packages because of its ease of use. [2]

Difficulty/simplicity of parallel program development.

Availability of parallel problem solving environments. [4]

From my point of view I see no clear transition way to migrate from the single-processor philosophy to the distributed/parallel processing

I would have asked for how easy would be for me to access resources that would help on this migration [17]

---

1. *Question 10 in Chapter "Start" I answered with no. Please do not misunderstand this. I have plenty of work at the moment and I do not want to be disturbed by minor questions. If you have important and urgent queries, please do not hesitate to contact me.*
2. *2. In general I see no advantage of grid computing for end users with a need of program development and plenty of production runs at the moment. Neither the technical nor the political questions (access to resources abroad) have been solved yet. I hope that this will change in the future.*

3. This questionnaire took me 100 minutes instead of 20 minutes promised! [23]

Research funding is often on a short cycle 2-3 years and success rates are < 30% so it is very difficult to predict future requirements.

In such a large research group there are a large mixture of codes used so it is extremely difficult to give one general response. [28]

You haven't inquired about the importance of locally maintained, in-house Beowulf clusters. We believe that this technology is becoming a major trend in supercomputing. The price/performance is unbeatable, and local control over resources is perhaps the most crucial factor in maintaining a productive computing environment. Many colleagues around the world are moving into this direction, bypassing computer centers along the way.

It is not obvious that there will be a widespread need for Grid resources, unless of course they are made available for free (meaning that someone else is paying for it). [30]

I am not capable of answering to point 6 in a sound basis. [35]

Really impossible to project to 10 years from now. The answers will be strongly modified by how Grid computing develops. [39]

Risk of dramatically increasing costs due to commercialization of software on The Grid. [43]

Please try to add to the current HPCN trends the area of providing services, running the FPGA design packages

-simulators,

-VHDL compilers

-Place and route tools

for large multi-million gate FPGAs

This will be one of the major CPU time consumers in the future. It would be great enabler for people from the east if they could run these tools (which are in general offered by Europractice .... IST KA 4) remotely over the GRID... [53]

Interest mainly in middleware so some of the questions were inappropriate [55]

I don't consider myself as particularly literate, smart, or visionary on these issues. Some of my answers may be quite stupid, or off the point. You are most welcome to contact me again. I have tried to answer the questionnaire for the whole Theory Group which comprises some 30-40 researchers, whose computational needs vary very strongly.

Usually we have only one or two truly high-end people at the same time. [57]

I just want to say that I am currently doing research on a distributed computing model. Thus my answers my not reflect what is needed (required) by the typical associates of the EPPC. [66]

Please take into account that as of currently the group has relatively limited experience in using high performance computing solutions. This means that it is hard to give very accurate answers to many of the questions posed. We do see a need for hpc in the future since the amount of data in bioinformatics is growing very quickly and since we want to take on problems on a larger scale then earlier. [67]

Some of my answers are most certainly affected by lack of knowledge in the specific topic. [76]

Some of the questions in the "Future Needs" section do have different answers for different sub-groups. So these should be considered more a 'weighted average'. [80]

Part 8 – End

Whisky Preference

Ardbeg	1
Cragganmore	3
Glenkinchie	6
Glenmorangie	4
Highland Park	11
Lagavulin	11
Laphroaig	9
Other	23
Talisker	7
The Glenlivet	6
The Macallan	4