



GRID COMPUTING

Keith Norman

TESSELLA SUPPORT SERVICES PLC

Issue V1.R3.M0

February 2004



Introduction

In the late 1980's, the particle physics research community had a problem. CERN, the European Organisation for Nuclear Research, was staffed by literally thousands of scientists from dozens of countries. Although they spent time on the CERN site, their research work was often carried out at their home universities and research centres. These scientists were accustomed to using IT to exchange information (email, for example, had been an invaluable tool for them for some years) but they were now experiencing real difficulties in exchanging complex experimental data. Then in 1990, Tim Berners-Lee, a scientist working at CERN, conceived the World Wide Web as the solution to the problem of sharing scientific information between the disparate systems in scientists' home institutions.

The rest is history...or almost. The applications of the Web have far outgrown the original problem of sharing scientific data between scientists. In the meantime, the scientists' needs at CERN have far outgrown the Web. CERN's latest major project, the Large Hadron Collider (LHC), is set to push the boundaries of the Web into uncharted territory. The LHC experiment will crash particles into each other at a speed close to that of light. The resultant burst of energy will give scientists an insight into the earliest stages of the Universe, of the order of one millionth of a millionth of a second after the Big Bang. The collisions between the particles will happen at a rate of almost a billion times every second, leading to data rates of a few petabytes (10^{15} bytes) per second. Super-fast electronics will remove uninteresting background from the data, but data suitable for analysis will still be of the order of 10 petabytes per year.

So how will the CERN scientists analyse this extraordinary amount of data? Certainly not exclusively on the CERN site – the computing needs are phenomenal. Soon after the development of the Web, CERN technology experts began to consider this problem and realised that the solution would require a new sort of network, a network that built on the traditional concepts of simply sharing data between devices, and added the idea of sharing computing resources.

The New Computing – Grids

The name given to the idea of building on the existing Web to provide not just shared data, but also shared computing resources, is “The Grid.” The name suggests an analogy with electrical power transmission via the national power grid, for example, if you buy a kettle you don't have to worry about where the electrical generator is, so long as you have a standard socket in the wall and a standard plug on the kettle. The user of the kettle just has to boil the water by

pressing an obvious (ie, standard-style) button. All of the decisions about where the power actually comes from, how it was made, and how it is fed to the wall socket are made behind the scenes. Similarly, in the Grid Computing world, the idea is that a user submits a request to run an application with some input data through a standard interface, and behind the scenes the Grid allocates a suitable resource, checks on the progress of the application, and eventually returns the results to the user. As in the kettle analogy, the user need not know where the application ran, or where storage for the running application was held.

In the grid world, computing resources mean more than just data or processing power – it might mean disk space, memory, tape backup, or even software licences.

Figure 1 shows some examples of Internet based systems – web services, email, WWW pages and the Grid. The lightning strikes represent the “magic” that works behind the scenes to link across the Internet. These are interfaces, either programmatic APIs called from code in the case of web services, or graphical user interfaces in the case of WWW pages and email.

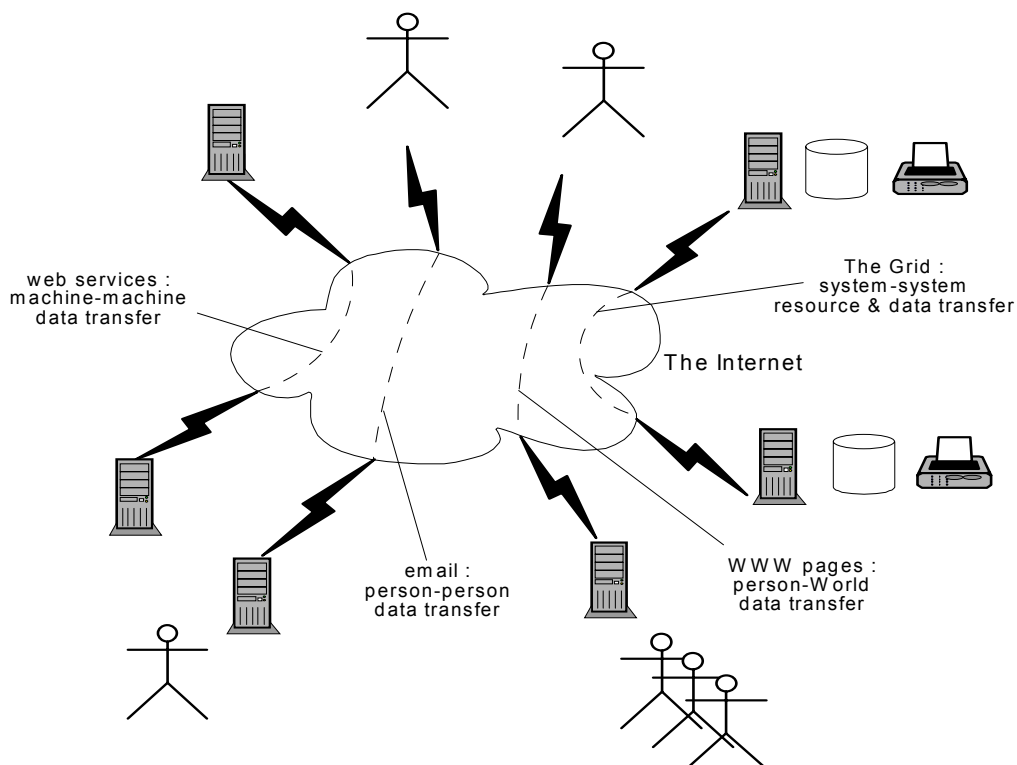


Figure 1 Examples of Internet Usage

The vision for the computational Grid, as in the power grid analogy, is that it should be:

- ❑ *dependable*
- ❑ *consistent*
- ❑ *pervasive*
- ❑ *inexpensive*

The qualities of dependability, consistency and inexpensiveness hold true to a large extent in the case of web services, WWW pages, and email, and although they are not truly pervasive, they are at least broadly accessible to many people.

With the experience gained over the last decade in the growth of the Internet, it is true now that organisations can already build distributed Grid environments which are certainly dependable, and if not inexpensive, at least show large cost benefits. What is less true is that access to such environments is “everywhere”, or that such distributed environments can easily communicate with each other; the consistency and pervasiveness goals are currently the subject of most development.

The Grid and Distributed Computing

It is worth noting that the terms “Grid” and “Distributed Computing” are often used interchangeably, but there are subtle differences between the two. Distributed Computing (ie, sharing resources between multiple machines) has been with us in various guises for decades – think of multiprocessor mainframes and sysplexes, Beowulf Clusters, shared network drives, and so on. In recent years, the idea of running distributed applications over the Internet has become increasingly important. Such applications (eg, <http://www.climateprediction.net>, or <http://setiathome.ssl.berkeley.edu/>) will typically download some code and data on to the client machine, process the data, and then upload back to a central server. These are certainly aspects of the Grid framework, but in itself, this does not make these systems true Grid applications. The qualities of *dependability* and *inexpensiveness* are satisfied by such applications, but it is not true to say they are *pervasive* (a port of the code typically needs to be made to each client operating system – although more generic systems are being introduced), nor are they

consistent (each application typically uses its own interfaces, security methodologies, resource allocation tracking, and so on).

This is not to imply that distributed applications are unimportant, but it is probably true to say that they are steps on the road to the fully-fledged, consistent, pervasive Grid.

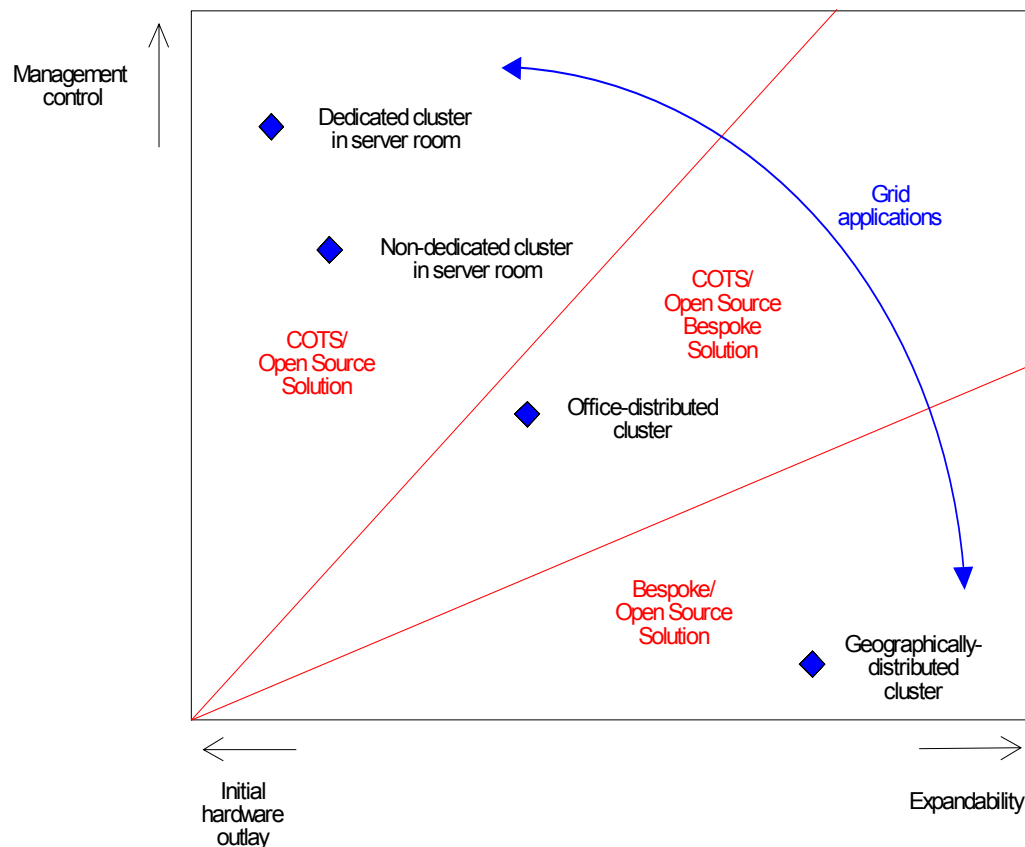


Figure 2 Methods of distributed computing

Figure 2 shows four methods of distributed computing :

1. **Dedicated clusters in a central location** (e.g., 6 servers in a server room in a Beowulf Cluster) There is a high degree of control over such systems, but initial outlays can be large.
2. **Non-dedicated clusters in a central location** (e.g., 4 accounts servers in a server room which are not utilised after 5pm) Distributed applications can be run overnight on such machines.
3. **Office-distributed clusters** (e.g., desktop workstations which are used during the day for reading email and running word processing software, but which are not used overnight) Distributed applications can be run

overnight, or potentially during the day, either as a screensaver or low priority.

- 4. Geographically-distributed clusters** (e.g., desktop workstations outside the corporate firewall, say at another company office, or home users' PCs) Again, applications could be run as a screensaver, or low priority all the time. There is clearly very little control that a systems manager can have over widely distributed machines, but the potential for growth is enormous.

Distributed computing architectures can be implemented using either Commercial Off The Shelf (COTS) packages, open source, or bespoke solutions. COTS solutions are typically more suited to "internal" distributed applications (i.e., inside the corporate firewall), as there may be licensing issues. Table 1 below compares some of the implementations of distributed computing solutions.

Implementation	Example usage	Licence	Benefits	Disadvantages
Globus Toolkit	·Academic research distributed across many globally distributed institutions	Open source	·No licence costs ·Relatively mature on UNIX ·Support from "big players" - IBM, Cray, etc	·Less mature on Windows ·Potentially less easy to implement than commercial solutions
Commercial grid hardware and software based on existing product line (eg, Sun N1 Grid, Oracle 10g)	·Distributing applications within one organisation	Commercial licences	·Specialist technical support · Complete solution out of the box	·Licencing costs ·Can be tied into single vendor
Commercial generic grid software (eg, United Devices Grid MP, Vita Nuova Inferno)	·Distributing applications within one organisation, or potentially across geographically distributed sites	Commercial licences (usually per seat)	·Very generic ·Good technical support ·Support for Windows and UNIX	·Typically just a base product - not a complete solution ·Licence costs
"Build your own"	· Any distributed computing	None, unless third party contractor keeps IPR	·Bespoke solution - does exactly what you need ·Potentially easy to "tune" as time goes on ·No licence costs	· Need specialist staff to implement and support

Table 1 Comparing distributed computing implementations

The Promise of the Grid

People use the term “Grid” to mean different things. The World Wide Web is well understood. The idea of the Web conjures up images of web browsers, search engines, chat rooms, booking airline tickets online, and so on. The equivalent concept in the Grid world are still under design, and there are as yet no smoking guns to represent the equivalent of Internet Explorer, Google, and the company intranet in a Grid environment. In particular, nobody yet knows what the Grid “killer app” is likely to be, in the same way as the CERN scientists did not initially appreciate how the Web would change both the way that industry works, and how we spend our leisure time. Perhaps the most important application of the Grid will be in generating innovation.

That said, the main thrust of Grid development has been for compute-intensive applications from the science and engineering sectors. For the time being, these might be described as pseudo-Grid applications, which probably bear little resemblance to the way that the Grid will work in ten years’ time. Compare this with the early days of the Web. In the absence of generic browsers, simple command line interfaces were quite widely used in the scientific community, an almost unimaginable way to access data on the Web ten years on.

Bearing in mind that the purpose of the Grid is to support *distributed* computing, some types of application are unlikely to fit naturally into a Grid environment, at least for the time being. For example, a word processor is likely to perform more efficiently running on a local machine than running on a remote processor, even if the remote processor is very fast. Delays in network traffic and an increase in the number of failure points will probably outweigh the small advantage of not having processing power and memory installed locally.

To get a feel for the type of applications that are driving the early days of the Grid, we should first explore the aims of deploying an application into a Grid environment. Firstly, it is **an effective way of using under-utilised resources**. Consider the desktop machines in your organization. It is a fair bet that almost all of them are not being used for at least sixteen out of every twenty-four hours, and not at all at weekends. Even during working hours, running a word processor and email application does not stretch a 2GHz PC with reasonable memory. Similarly, a new machine is likely to come with a 40 or 80Gb hard disk with a very substantial fraction of this free, even taking into account the operating system and applications.

Secondly, **many applications can benefit from running on multiple processors in parallel**. An application that can be split into steps that do not rely on each other is a good candidate for running in parallel. Sometimes it can be obvious how to split the application into individual jobs. Often, the split is not so easy to identify, or there will be some natural barrier in the way the application was originally written (if one hundred separate jobs are all reading and writing to the same database concurrently, the bottleneck is likely to be the database itself). There is a large class of problems where the split between machines may not be obvious, but can be arrived at with some thought, for example, some large matrix inversions can be suitably parallelised by using mathematical methods. It may be that there is no way to completely split a problem into a series of parallel problems, but even in these cases it may be possible to split to some extent, but still allow individual nodes to exchange information. Such information might be boundary or starting conditions for the next phase in a workflow, or systems with response requirements where some parts of the problem must run at higher priority than others. Problems like this can be parallelised, but are usually only practical to run in managed networks where administrators have some control over network bandwidth between nodes.

Thirdly, applications deployed in a Grid environment will naturally **allow costs and processing to be shared amongst different organisations**. This is of benefit to large collaborations, such as the CERN experiments, where the nature of the Grid makes sharing of data and processing natural and simple.

The Users

Although the early days of the Grid are likely to be driven by processing scientific data, the promise of low cost, dependable, and standardised access to resources is already proving attractive to a range of organisations. For example, Novartis are harnessing the power of under-utilised PCs across the company to provide an extra 5 trillion operations per second. The Grid will improve product time to market, increase flexibility, and allow better use of company resources. Access to the Grid will allow:

- ❑ government agencies to set up very large scale computing infrastructures
- ❑ hospitals to share high volume patient data (such as X-rays and mammograms) securely

- ❑ the film, media and entertainment industries to render images orders of magnitude more quickly than is currently possible using local farms of fast processors – potentially leading to high quality 3D virtual worlds
- ❑ environmental agencies to predict and analyse the progress of oil spills in real time
- ❑ chemical engineering firms to model large scale catalyst surfaces
- ❑ the nuclear industry to model the seepage of nuclear waste into bedrock
- ❑ designers of high speed mixers to model the mixing process for high load conditions
- ❑ medical scientists to model protein folding
- ❑ the defence industry to model explosion dynamics
- ❑ the computing industry to improve methods of pattern matching for antivirus and spam-filtering applications
- ❑ organisations deploying RFID tags on a wide scale to populate and manage ultra large databases (e.g., some organisations are spending hundred of thousands of dollars on RFID tagging their entire stock inventories).

Some of these applications, particularly simulations, are already possible on individual processors or local farms of interconnected machines. However, the Grid will allow such simulations to run orders of magnitude faster on orders of magnitude larger datasets.

Note that the resources to which the Grid will allow access are not limited to processor power alone. Network storage, networked licences, networked hardware, and even network bandwidth itself will all be Grid-enabled. Businesses will benefit from not having to purchase expensive storage options, but from being able to outsource their storage requirements to the Grid. Access to a business storage network will be transparent to the users, but disks hosting the business' data may be in another country. Organisations will be able to add storage capacity dynamically, without having to schedule any downtime.

What Does The Grid Need To Provide?

To be effective, there are several services that the Grid needs to provide:

Consistent Interface: A persistent problem in the IT industry is the lack of interoperability between disparate systems. Grid technologies need to tackle this head on by providing simple, consistent access to computing resources, regardless of whether a processor is running a Windows operating system, Linux, or a proprietary mainframe OS. Interfaces need to be provided both for submission to the Grid and for extracting data from the Grid. For user access, the development of the Web encountered similar problems and solved them effectively with the introduction of standardised web browsers, such as Internet Explorer and Netscape. From the programming perspective, the problem has more recently been addressed with the introduction of web services under .NET and J2EE.

Information Service: In the early days of the Web, information services were limited. Search engines existed, but the technology that powered them was underdeveloped. Thinking back to the early 1990s, typing a search phrase into a search engine would probably lead you to the site you were looking for, but it would be buried in a mass of other irrelevant hits. Information services are even more important in a Grid environment. The services need to provide information about what resources (processing power, special equipment, software licences, additional services) are available, and this information needs to be automatically maintained. The information must be complete and descriptive, but at another level, abstract. The goal of the Grid is to hide the specifics of where a particular processor resource is, how it works, and so on, from the end user (just as the Web abstracts information from the physical location of that information).

Resource Allocation: Users will need to submit requests for computing resources in an abstract way (i.e. without specifying the precise details of where their application should run) to a brokering system which will allocate and schedule resources effectively. This is a major challenge for the designers of tomorrow's Grid applications. There is no real parallel in the development of the Web, although job scheduling systems which handle multiprocessor architectures have been around in the mainframe and UNIX world for many years.

Data and User Security: Data on the Grid may need to be transferred and accessed in a secure fashion. Users' identities may need to be kept secure, and access to resources will certainly need to be controlled. Security has been a driving force in Web developments since its inception, and although there is still

no 100% solution to the security problem, a great deal of progress has been made with digital certification and signatures, remote authentication, and so on. There can be no doubt that many of the security technologies that have pushed the Web forward will be reused in Grid applications.

Software Components

To implement a Grid requires a complex software suite. Part of the software needs to keep track of resource availability and monitoring. This component needs to monitor all nodes available on the Grid, measuring their overall capacity, and what capacity remains available on a particular node. Both large and individual nodes need to have network components to allow them to communicate. Each node needs a component to register its existence with the rest of the Grid and identify it for security purposes, maybe using a digital signature. Furthermore, each node will require a component for accepting resource allocation requests. For a commercial grid, there may need to be a purchasing component to enable users to buy access to resources. Finally, there will need to be an overall security component associated with users of the Grid. This will limit utilisation of the Grid to validated individuals, and may be coupled to the purchasing component.

Figure 3 shows how the software components of Grid applications fit together. The diagram is reminiscent of the way components are shared in a traditional client-server application, but bear in mind that the Grid components are distributed – the Grid is the server, and the individual servers are in fact clients of the Grid themselves.

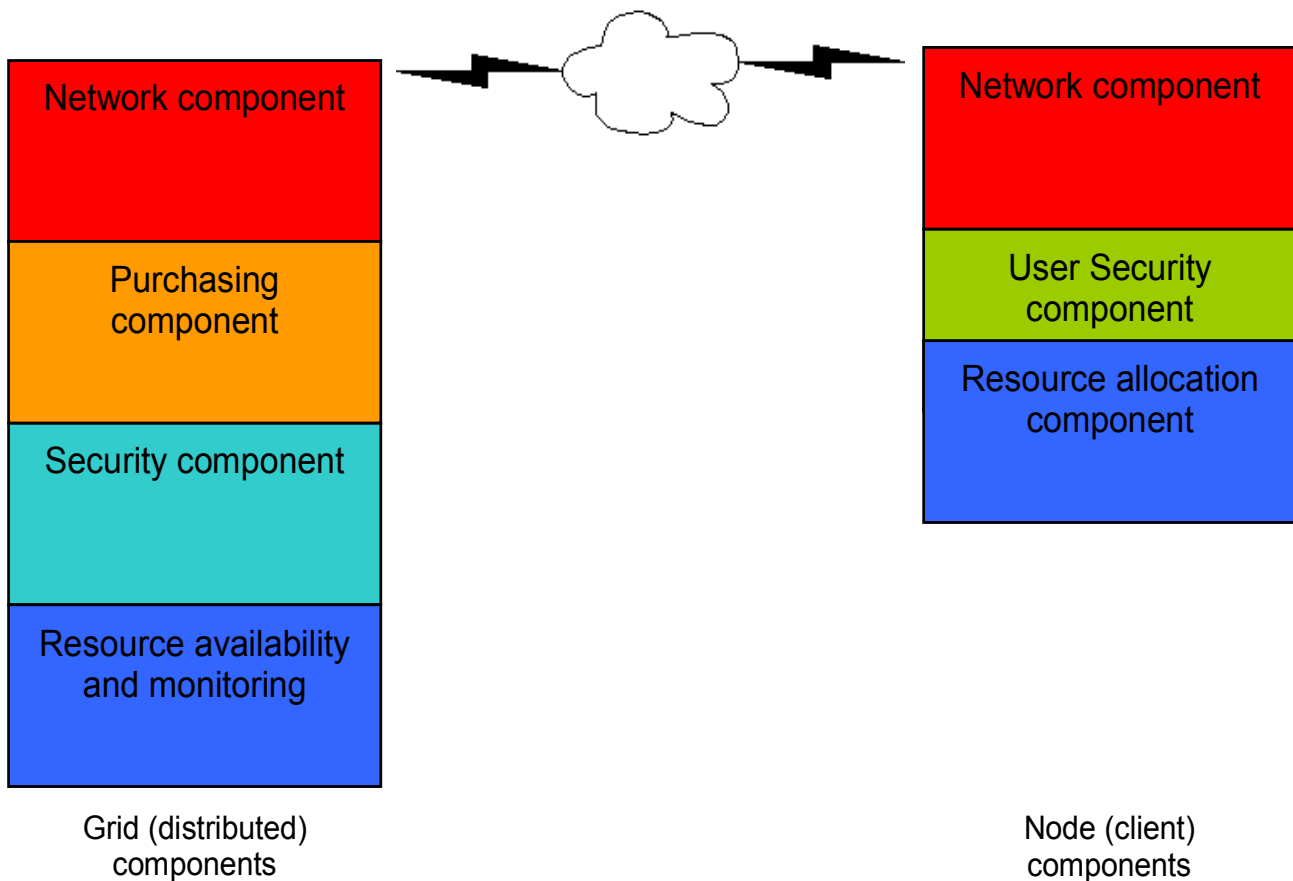


Figure 3 Software Components

Application Toolkits

The development of the Web was intimately tied up with the development of the web browser. Although there are still a variety of flavours of web browser, Microsoft Internet Explorer and Netscape Navigator are now fairly ubiquitous on PCs, with appropriate “cut-down” versions on handhelds and mobile phones. These provide a standard view of the Web through a consistent interface. Interestingly, consistent underlying applications layers for communicating across the Web have only comparatively recently emerged with the introduction of web services in Microsoft .NET or J2EE.

In the case of the Grid, the reverse situation is occurring, where the applications interface layer is emerging before consistent user interfaces. There are already many widely used tools for building web services, such as IBM’s Websphere, Oracle Portal, and Microsoft’s .NET family. On top of these, open standards are being developed and implemented that will move data-sharing web services to resource-sharing grid services. A key player in pushing the standards that are required by the Grid and developing the tools to implement these standards is the

Globus Project (<http://www.globus.org>). The tools produced by Globus are referred to as the Globus Toolkit. The toolkit is open source, and provides libraries for resource monitoring, discovery and management, security and file management. Compaq, Cray, SGI, Sun, Veridian, Fujitsu, Hitachi, NEC, Entropia, IBM, and Microsoft have all announced commitments to the Globus.

The Globus Toolkit is evolving, but work is continuing to develop better standards for error handling and notification, database handling, workflows, and so on, and to improve overall reliability and quality of service, driven by the Globus Project in partnership with IBM amongst others . This is being specified in the Open Grid Services Architecture (<http://www.globus.org/ogsa>).

The toolkit has several components designed to integrate the distributed hardware of the Grid through a collection of libraries and services:

Globus Resource Allocation Manager (GRAM): A basic library service to handle job submission.

Grid Information Service (GIS): A directory service to locate Grid resources. Also known as the Metacomputing Directory Service.

Grid Security Infrastructure (GSI): A library providing security services.

GridFTP: A data transfer protocol for high bandwidth wide area networks. Based on FTP, GridFTP includes GSI security, multiple data channels, partial file transfers, authenticated data channels, and reusable data channels.

Globus Access to Secondary Sources (GASS): A remote data access component.

Globus also provides a layer above these services to give a simple user interface for submitting jobs, starting a remote shell, and so on.

Using Globus

The Globus Toolkit can be downloaded from <http://www.globus.org/>, either as prebuilt binaries, or as original source. Full instructions are given for installation. Once installed, users must obtain a Grid Certificate. This is similar to other private key Internet certificates that are already widely used. With a Grid Certificate, users only have to enter a password once per Grid session, without having to re-authenticate on every machine that they use on the Grid. Typically, an administrator assigns user logons and Grid Certificates to users, sets up the environment and installs the certificates on a Globus home machine for that user. The home machine will be the one from where the user submits jobs to the Grid,

although users are allowed to have more than one home machine.

Once the Grid Certificate is installed, it must be activated. This is referred to as obtaining a Grid Proxy and the activation is by default valid for twelve hours. With the proxy activated, the user can start to submit jobs to the Grid. There are various ways of doing this in Globus:

- ❑ `globus-job-run` runs a job in the foreground, with output coming back to the user's terminal, like `rsh` for the Grid.
- ❑ `globus-job-submit` runs a job in batch, so the user can log out and leave the job running. Output is collected with `globus-job-get-output`.
- ❑ `globusrun` runs a job using Globus' scripting Resource Specification Language (RSL). `globus-job-run` and `globus-job-submit` are in fact front ends to `globusrun`.
- ❑ `mpirun` submits jobs to the Grid for parallel processing using the Message Parsing Interface, a commonly used standard for programs designed to run on parallel processors.
- ❑ `globus-sh-exec` spawns a shell on a remote Grid node, and handles the environment setup for that shell.

For example, to submit a simple "Hello World" application:

```
globus-job-run gridserv.tessella.com /bin/echo "Hello World"
```

will print

```
Hello World
```

on the submitter's terminal.

There are a range of commands for monitoring the progress of jobs, clearing job output, and cancelling jobs, which mirror those of traditional batch systems.

After a job is submitted, Globus contacts the remote host and authenticates. The remote host then submits the job to a job manager service. This service decides how to run the job, either by forking a new process immediately, or by submitting to a local batch processor.

The Globus Toolkit also provides a set of interfaces to the GIS (Grid Information Service). These are accessed via a set of `grid-info-*` commands. GIS publishes information via the Lightweight Directory Access Protocol (LDAP), enabling it to be used as a directory service for the components of a particular grid node, or at a higher level as a lookup system for names of users.

Remote data access is implemented in a set of Globus Access to Secondary Storage (GASS) interfaces. Turning on remote file access on a server is accomplished via a `globus-gass-server` command, which opens an https socket on that machine. A remote machine can then download files from the GASS server via a `globus-url-copy <fromURL> <toURL>`.

In addition to the command line interface, the toolkit provides a rich set of API calls for accessing the Grid natively from C or C++, or through Java using some extensions to the base Globus package.

Conclusions – the Future of Distributed Computing

In this article, we have explored some of the computing applications that will benefit from running in a distributed environment. Distributed computing has existed in various forms for many years, and we are already being rewarded with results from *climateprediction.net* predicting the range of possible climates over the coming decades, *seti@home* looking for extraterrestrial radio signals, and a variety of other projects. However, the idea of the Grid is to provide a consistent framework for distributed computing resources in the same way as web browsers and web services provide a standard interface to distributed data. Instead of having a circus of distributed applications, organisations will be able to deploy to the Grid in a consistent, dependable way. This will pave the way for truly pervasive computing.

Grid technology, in particular the Globus Toolkit, is already in use but so far Grid applications are limited to discrete clusters of users. There is as yet no single Grid, compared to the “single” World Wide Web. Potentially the biggest problem to overcome now is not the software technology (Globus and COTS solutions on top of web services are maturing), nor hardware technology (many desktop PCs use only a few percent of their resources for most of the day), nor network technology (Gigabit Ethernet is becoming standard on even desktop PCs, and broadband access to the Internet is increasingly ubiquitous). Instead, the challenge is to find the Grid applications that will transform our working and leisure lives.

Tessella Support Services plc
Creating Software for Science and Engineering

Tessella's services range from feasibility studies, through system design, development, implementation and ongoing support. Our expertise includes:

Data Analysis Software
Data Capture
Simulation Software
Advanced Graphics
Systems Support
Database Applications

Other Technical Supplements available include:

- | | |
|---|--|
| <input type="checkbox"/> Active Server Pages | <input type="checkbox"/> Linux |
| <input type="checkbox"/> Archiving of Electronic Info | <input type="checkbox"/> Microsoft .Net |
| <input type="checkbox"/> Automated GUI Testing | <input type="checkbox"/> Object Oriented Programming |
| <input type="checkbox"/> Bayesian Statistics | <input type="checkbox"/> Pocket PC |
| <input type="checkbox"/> Beowulf Clusters | <input type="checkbox"/> Portable GUI Development |
| <input type="checkbox"/> C++ | <input type="checkbox"/> Printer Technology Guide |
| <input type="checkbox"/> COM | <input type="checkbox"/> Real Time Systems |
| <input type="checkbox"/> Computational Fluid Dynamics | <input type="checkbox"/> Regression Testing |
| <input type="checkbox"/> Computer Image Processing | <input type="checkbox"/> Security and the Internet |
| <input type="checkbox"/> Decision Support Systems | <input type="checkbox"/> Simulation |
| <input type="checkbox"/> e-GIF | <input type="checkbox"/> Soft Computing |
| <input type="checkbox"/> Electronic Data Capture | <input type="checkbox"/> Software Design Methodologies |
| <input type="checkbox"/> Electronic Lab Notebooks | <input type="checkbox"/> Software Development Cycle |
| <input type="checkbox"/> Excel | <input type="checkbox"/> Software Documentation |
| <input type="checkbox"/> Extending the Life of Software | <input type="checkbox"/> Software Portability |
| <input type="checkbox"/> FDA 21 CFR Part 11 | <input type="checkbox"/> Software Re-engineering |
| <input type="checkbox"/> Formulation | <input type="checkbox"/> Software Specification |
| <input type="checkbox"/> FORTRAN 90 | <input type="checkbox"/> SQL |
| <input type="checkbox"/> Grid Computing | <input type="checkbox"/> UNIX Inter-Process Comms |
| <input type="checkbox"/> High Throughput Screening | <input type="checkbox"/> UNIX Systems Performance |
| <input type="checkbox"/> Instrumentation | <input type="checkbox"/> Web Services |
| <input type="checkbox"/> Integrated Lab Systems | <input type="checkbox"/> Windows 2000 Services |
| <input type="checkbox"/> J2EE | <input type="checkbox"/> XML |
| <input type="checkbox"/> Java | <input type="checkbox"/> X Windows |
| <input type="checkbox"/> LIMS | |



INVESTOR IN PEOPLE

Tessella Support Services plc

3 Vineyard Chambers, Abingdon, Oxon, OX14 3PX, England

Tel: (+44) (0) 1235 555511 Fax: (+44) (0) 1235 553301

E-mail: info@tessella.com Web Address: <http://www.tessella.com>