

# Using the grid

Dirk van der Knijff

David Bannon

David Abramson

Lindsay Hood

Markus Binsteiner

Marco La Rosa

## Workshop outline

- Introduction
- Scene setting - what is “The Grid”
- Authorization - getting a certificate
- Nimrod - a tool for parametric studies
- Using the command line
- Handling data (EGEE)
- Wind up

# Introduction

This is a hands-on workshop. Think of it as being like a driver-instruction course, not a motor-mechanics course.

Grid software should be considered like the OS on your computer - you might install it yourself, but you don't understand most of it and have no need to - if it doesn't work you call an expert.

Grid policies and usage are however relevant. You can (sometimes...) choose the OS you use and similarly you (or your collaborators) can choose which grid "flavour" to use.

# What is "The Grid"?

The grid isn't actually a thing at all. It's a combination of a number of different pieces of software and a set of policies which enable people to share ICT resources.

Rather than defining it we will first look at the drivers which caused people to develop the concepts and tools which we now call "grid technology" and the policy requirements associated with their use.

Note: as usual, since the drivers were in place first, many different solutions were developed which has lead to the current situation of many different "flavours" of grid software...

## Grid drivers (1990's)

- Exponential increase in cpu performance
- Huge expansion in network bandwidth
- HPC move to “cluster computing”
- Massive data sources
- Data aggregation
- Focus on “collaboration”
- Insistence on “efficiency”

## CPU Performance

cpu performance has been following “Moore's Law” for over 40 years now with no end in sight. While users now can perform vastly complex operations in genuinely interactive fashion, this has led to a huge reserve of “unused” cpu capacity spread across millions of desktop systems.

## PC's - then and now

- 1990
  - 486DX, 33MHz 1.5 MFlops
- 2000
  - PIII, 933MHz 43.1 MFlops
- today
  - Core2duo, 2.16GHz 250.8 MFlops

## Network bandwidth

The dot.com boom in the late nineties may have caused some financiers much grief (and many others), but it had the effect of every telecom organization, and even some governments, installing new fibre-optic cables seemingly everywhere. While their expected users didn't arise, we now have a globe that seems to have high-speed, high-bandwidth connections everywhere. Most of us have at least 100Mbps to our desktop, and most Uni's now have 1Gbps to the world.

## Networks - then and now

- 1990
  - 1-10 Mbps to desktop
- 2000
  - 10-100 Mbps to desktop
- today
  - 100-1000 Mbps to desktop

## Cluster computing

In 1990, High Performance Computing (HPC) was still mostly done on computers with a small number of fast cpus such as the Cray vector computers. But the rate of improvement in commodity processors was making this uneconomic. Many people were experimenting with “massively-parallel” systems built from simple processors, but the introduction of clusters of commodity systems made a new paradigm possible. Now it seemed anybody could build a supercomputer.

## HPC - then and now

- 1990
  - Cray Y/MP 1cpu 283 MFlops
  - 8cpus 2233 MFlops
- 2000
  - Cray T3E 1cpu 47 MFlops
  - 512cpus 23743 MFlops
- today
  - SGI Altix 1cpu 400 MFlops
  - 1024cpus 362513 MFlops

## Massive Data Sources

The 1990's also saw a vast increase in the amount of data being generated.

As science got "bigger", govt's funded fewer bigger experiments.

Earth observing satellites started producing data.

Bio-informatics came of age.

More and more data was "born digital", and if it wasn't plans were afoot to "digitize" it.

In its October 20, 2000 issue The Economist reports on a study performed at UC Berkeley that attempted to estimate how much data is created annually throughout the world.

Here is a summary of the findings-

- 80 billion photos at 5 MB apiece yields 400 petabytes.
- 4,250 films at 4 GB apiece is 17 terabytes.
- 2 billion x-rays at 8 MB apiece is about 16 petabytes.
- 610 billion emails annually in the United States.
- 576,000 terabytes of telephone calls.
- Published information (meaning movies, tv shows, books, magazine articles), 285 terabytes a year.
- Individuals create 740 petabytes of information a year.
- Amount of disk space sold attached to personal computers in 1999: 1 exabyte.

## Data aggregation

Another consequence of both “big science” and the data explosion was the aggregation of data. Huge experiments stored vast amounts of data locally, while in other cases, people pooled data.

Researchers were starting to discover (following on from business) the benefits of putting all their data in one place e.g. the human genome project where data from multiple groups was aggregated (and then replicated) resulting in faster results for all.

## Collaboration

While collaboration has always been part of Uni life, particularly in the research field, the “big science” paradigm made it essential. Some High-Energy physics collaborations have in excess of 500 active members, while “soft” collaborations such as that between the users of major instruments such as the Hubble telescope or satellite sensors involve many thousands with access to the same data.

## Efficiency

Lastly, the “bean-counters” became ascendant. They required efficiency, return-on-investment, value-for-money, etc., as part of grants, and as money became tight, everybody looked for ways to make the money go further. This was also taken up by research funding agencies who “mandated” efficiency in various ways. Sometimes this was as simple as cutting a grant by an “efficiency improvement” amount, at other times just requiring substantiation and extra paperwork.

## All together

Thus, by 2000 we had vast amounts of data stored in many locations, hordes of underutilized computers (mostly on peoples desktops), massive investment in new networks (dot.com), collaboration becoming the normal mode and everybody trying to become more efficient.

## So... Enter the Grid

In 1998, Ian Foster and Carl Kesselman published a book entitled “The Grid: Blueprint for a New Computing infrastructure”. This book promoted the concept of sharing, initially computational, resources.

They defined:

*“A computational grid is a hardware and software infrastructure that provides dependable, consistent, pervasive, and inexpensive access to high-end computational capabilities.”*

## Grid concepts

By 2000, this had been refined by them (and Steve Tuecke) as:

“The sharing that we are concerned with is not primarily file exchange but rather direct access to computers, software, data, and other resources, as is required by a range of collaborative problem solving and resource-brokering strategies emerging in industry, science, and engineering. This sharing is, necessarily, highly controlled, with resource providers and consumers defining clearly and carefully just what is shared, who is allowed to share, and the conditions under which sharing occurs. A set of individuals and/or institutions defined by such sharing rules form what we call a *virtual organization*.”

## Great Idea, but...

A huge number of issues needed to be resolved -  
admin things like security, policy, payment,  
membership of virtual organizations,

technical things such as interoperability, resource  
discovery, resource access, availability, quality of  
service - the list just goes on.

It did however prompt people to start developing the  
software and discovering first-hand what was hard  
and what was easy...

## How to make it work

So how do we get from a good idea to an actual working grid where people can get things done? Firstly, Foster, Kesselman and Tuecke had been working on a software toolkit, the Globus Toolkit, for some time. They were looking for partners and willing guinea-pigs to test it. In 2001, they released version 2 (GT2) of the toolkit as “production ready”. In practice it wasn’t as ready as they thought, but the experience gained by the many sites that adopted it has led to most of the issues being properly understood at least if not all solved as yet. In 2005 they released GT4 which underlies most new grid implementations.

## The three A’s

- Authentication
  - Who are you
- Authorization
  - What are you allowed to do
- Accounting
  - Who did what when
  - (I’m not going to talk about this)

# Authentication

Users, hosts and services need to be able to authenticate themselves in the grid environment. Experience in using grids for remote computations has demonstrated the need for unattended user authentication in addition to interactive authentication. Unattended authentication of users is needed when a user is making frequent requests to remote servers and does not want to repeatedly type in a passphrase, and when a long running job may need to authenticate itself after the user has left.

## Authentication 2.

Basically, authentication between two entities on remote grid nodes means that each party establishes a level of trust in the identity of the other party. In practical use an authentication protocol sets up a secure communication channel between the authenticated parties, so that subsequent messages can be sent without repeated authentication steps, although it is possible to authenticate every message. The identity of an entity is typically some token or name that uniquely identifies the entity.

## Authentication 3.

The most commonly used identity token in Grid applications is the X.509 public key certificate. This contains a public key, a subject name in the form of a multi-component Distinguished Name [DN] and a validity period and is signed by a trusted third party called a Certification Authority (CA).

The X.509 certificates are used with the TLS (SSL) security protocol to make a secure authenticated connection between two parties.

A grid CA is defined as a CA that is independent of any single organization and whose purpose is to sign certificates for individuals who may be allowed access to the grid resources and hosts.

## Proxies

In actual fact we don't use the certificates directly, what we do is generate a passwordless token called a proxy certificate (usually simply called a proxy) which has a limited lifetime and is attached to the task we have submitted to the Grid. This proxy can then be used at any time during the task execution when access to resources is required e.g. to access data, or if a job has to be restarted on another system.

Proxies can be generated directly, but are usually generated by a service run by a virtual organization which adds its own signature to the proxy.

## Authorization

Having “passed” authentication - i.e. we know who you are, we now need to check what you are authorized to do. This was initially done by adding the certificate to a list but this wasn't scalable. Now we usually rely on the signature added to the proxy by the virtual organization. This allows a person who is a member of a number of different VOs to have different access depending on who they asked for the proxy being used. e.g. I can be a member of the development VO and also the user VO on a grid with different access to resources.

## Access

Having determined who you are and what you are allowed to do, the next bit is even harder. Fortunately, as a user you don't have to worry as much about it. In some way, determined by local site policies and practice, you will be assigned a username (possibly created dynamically) with the appropriate access to the local resources. If you need to access non-local resources, your proxy will go along for the ride and again the resource should be provided in some way.

## Virtual Organizations

This sharing is, necessarily, highly controlled, with resource providers and consumers defining clearly and carefully just what is shared, who is allowed to share, and the conditions under which sharing occurs. A set of individuals and/or institutions defined by such sharing rules form what we call a *virtual organization*. VOs existed before the Grid, however they have a special place in relation to the three As. Since the grid is all about sharing resources and these are usually controlled by a VO, it makes sense to offload the authorization task to the VOs - they know their members and what they are allowed to do.

## Discovery

In order to be able to run a job, we need to know that the system is capable of running it. It's pointless sending a windows job to a linux system. This is done by systems advertising themselves and their environment to central servers. The Globus names for these are GRIS, for the local data collector, and GIIS for the central server which advertises the information to others.

## Resource Broker

Finally, we need a program to find the resources you require. This is called the resource broker. It determines the location of suitable resources and arranges access to them for you. Some tools, such as Nimrod which you will see later, carry out this function themselves, otherwise you need to access some external service. VO's usually provide one or more such services.

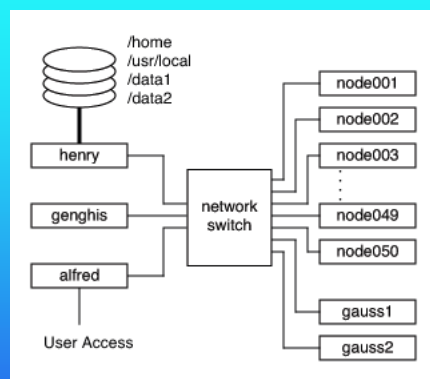
## The “Sandbox”

In normal operation of the grid a job may require access to many resources. These resources include the application required to execute the job, the data (if any) on which the job will run, auxiliary files that may be required for settings or other purposes, and hardware resources such as temporary disk space and CPU. The term "sandbox" has been used in the context of the grid to describe a temporary work space created on a computing resource that encompasses disk space, auxiliary files, and possibly the job application itself.

# Computational Grids

The first “grids” were computational grids - the reservoirs of untapped cpu cycles on desktops and under-utilized servers (and student labs) had already exercised the minds of HPC cluster managers and a number of ad-hoc solutions to utilize them were already in place. Authentication and Authorization can be handled on a person-to-person basis for small numbers of users and Access can be dealt with by systems managers or skilled users using various work-arounds. Remote job submission was thus a fairly well understood issue with a recognized need for a standardised solution.

# aside - Cluster Computing



This is a typical cluster implementation. On the left we have the “head” nodes while on the right we have the “compute” nodes. Users usually connect to the head node and then submit jobs which are run on the compute nodes.

## Cluster computing continued

- Single management domain
  - Username/password same on all nodes
- Shared file systems
  - nfs mounted everywhere
- Good connectivity
  - Low-latency and high bandwidth
- Single OS
  - May be different releases...

## Computational Grids 2.

With their ready experience with running jobs for users on different boxes, cluster managers were already aware of many of the issues, and were also able to fudge many issues. On a testbed, it's easy to give every test-user an account on all systems, or move files by hand if required. It's also possible to monitor each stage of a jobs progress and correct glitches when there's only a small number of them. By this means we gradually got the stuff working. GT4 supports job submission, file transfer, credential management, registry and database access services all in a reliable secure fashion.

## Data Grids

Data is much harder! While it's easy to imagine your job running on some remote computer that you've never heard of, getting a piece of data that you don't even know if it exists is not. Most digital data is not "google-able" so unless someone has written clear textual meta-data that won't work. What we do have is data management capabilities which allow us to move, copy, replicate and store data. A VO can thus form a "data-grid" where it's members can store and access data using a common methodology, but these rarely interoperate. The Virtual-Observatory project is one such grid which does have a public interface.

## Meta-data

This is the problem. Most data is not well described and therefore becomes unusable very easily (usually as soon as the project ends...). I'm sure everyone here has lost files because they've forgotten where they are, and the problem is much worse for digital data - even a clearly labeled file can be unusable if we don't know the format.

There are two main problems:

- 1. Getting people to add the meta-data to their data, and,
- 2. Knowing what we actually will need when we want to access the data at some future time.

## “Other” Grids

Since the definition of the grid involves resource sharing, we can also have grid that share other types of resources. The AccessGrid project is one example, although some don't consider it a “real grid”, it is certainly an example of sharing ICT resources to achieve a single goal. Other examples include using the grid middleware to co-ordinate instruments such as in the Earth System Grid in the US, and, closer to home, the intention to allow remote usage of the Synchrotron at Monash.

## Grid flavours

Because of the interest in grids and various problems encountered, development became fragmented. No-one was in control. The Globus group's first release was buggy and limited and people who wanted to start “doing grids” now, developed their own fixes to suit their own needs. Most Grids are based on either Globus ToolKit (GTK) 2.4 or 4.0 (3.0 was a disaster), but there are further choices of “helper” applications leading to a situation where we have different “flavours” of grid software used by different groups however we are now moving to interoperability.

## Grids

- APAC National Grid
  - Multiflavour - GTK 2.4 ,GTK 4, VDT
- EGEE (European Grid for E-scienceE)
  - Glite (developed from GTK 2.4)
- TeraGrid
- GridPP
- NorduGrid
  - Modified GTK 2.4 (ARC)

## Not Grids

Ian Foster, in 2002, proposed the following checklist for a system to qualify as a “grid”. It must:

- coordinate resources that are not subject to centralized control
- use standard, open, general-purpose protocols and interfaces, and,
- deliver nontrivial qualities of service.

Using this definition, many “Grid” products would not qualify. There has been a tendency to label extended cluster management systems with centralized control as grids, and similarly, the Web is not (yet) a grid because it fails to deliver co-ordinated use of resources (although such usage has been built on top of it).

# today

1. David Bannon - get a certificate
2. Lindsay Hood - facilities available
3. Morning tea
4. David Abramson - Nimrod-G
5. Markus Binstener - the command line
6. Marco La Rosa - EGEE demo
7. Questions...