What is this Thing Called the Grid?

Or: Is the Emperor Naked?

Leif Nixon

22 oktober 2003

Please, interrupt whenever you have questions.

These slides will be available from `http://www.nsc.liu.se/lcsc/`

The Google Hype Measurement

Some quotes

Irving Wladawsky-Berger, Vice President, IBM:

Grid Computing is really the natural evolution of the Internet. This is really looking at the Internet with all it's promise of universal connectivity and reach, and making it work far better [. . .]

[. . .] the promise here is our customers will be able to have their cake and eat it too.

Wolfgang Gentzsch, Engineering Director, Sun:

19th Century: Steam Engine
20th Century: Combustion Engine
21st Century: Grid Engine

Holly Korab, NCSA:

Cyberspace on steroids.

News clippings

IBM Takes Sony PlayStation Gaming to the Grid

The new online gaming network, based on grid technology, will be unveiled next week at the Game Developer's Conference in San Jose, California. It will allow developers to engineer games directly onto a grid and will make it easier for them to support online gamers.

(Hey, let's build a cluster of Playstations and hook them up to the grid! Wait, that's already been done. . .)

Sun Unveils Grid Computing Solutions Program

[. . .] more than 6,000 grids have been deployed worldwide – over 2,500 in EMEA alone

(It seems Sun have their very own definition of what a grid is.)

Money makes the world go around

Some EU funded grid projects:

- 'ASP-BP': Development of a framework for 6 experiments regarding ASP technologybased applications applied in different industrial fields.
- 'COG': Demonstration of the applicability of Grid technologies to industry.
- 'CROSSGRID': Development of techniques for large-scale grid-enabled real-time simulations and visualisations.
- 'DAMIEN': Development of essential software supporting the Grid infrastructure.
- 'DATAGRID': Development of techniques supporting the processing and datastorage requirements of next generation scientific research.
- 'DATATAG': Development of techniques to support reliable and high-speed collaboration across widely distributed networks.
- 'EUROGRID': Development of core Grid software components.
- 'FLOWGRID': On-demand CFD (Computational Fluids Dynamics) simulation and visualisation using Grid computing.
- 'GEMSS': Development of new Grid-enabled tools for improved diagnosis, operative planning and surgical procedures.
- 'GRACE': Development of a search and categorisation engine for flexible allocation of computational and data storage resources in Grid environments.
- 'GRASP': Development of architecture and business models for delivering ASP services over the Grid-enabled networks.
- 'GRIA': Development of business models and processes that make it feasible and cost-effective to offer and use computational services securely in an open Grid marketplace.
- 'GRIDLAB': Development of software capable of fully exploiting dynamic resources.
- 'GRIDSTART': Accompanying measure with objective of maximising the impact of EU-funded Grid and related activities through 'clustering'.
- 'GRIP': Interoperability of Globus and UNICORE, two leading software packages central to the operation of the Grid.
- 'LeGE-WG': Network of Excellence facilitating the establishment of a European Learning Grid Infrastructure by supporting the systematic exchange of information and by creating opportunities for close collaboration between the different actors.
- 'OPENMOLGRID': Development of tools for molecular design based on UNICORE enabled distributed computing environment.

Call it “grid”, and you get funding.

An often felt frustration

Ian Foster, Argonne National Lab:

Grids have moved from the obscurely academic to the highly popular. We read about Compute Grids, Data Grids, Science Grids, Access Grids, Knowledge Grids, Bio Grids, Sensor Grids, Cluster Grids, Campus Grids, Tera Grids, and Commodity Grids. [. . .]

If by deploying a scheduler on my local area network I create a "Cluster Grid", then doesn't my Network File System deployment over that same network provide me with a "Storage Grid"? [. . .]

Is there any computer system that isn't a Grid?

Cloudscapes of vapourware

An often repeated claim: *"There are OGSA[1] implementations available in Python, Perl and .NET".*

Actual state of implementations:

- **Python:** *"This code is presented as pre-alpha/development quality code."*
- **Perl:** *"OGSI::Lite is an experiment in creating a Grid Services Container using Perl."*
- **.NET:** *"The latest version of OGSI.NET is Tech Preview 1.1."*

Makes you feel safe and cozy, right?

[1] The Open Grid Service Architecture

The big question

Is the emperor naked?

Well, let's see. . .

Actually, you *can* use grid stuff today.

Especially suitable are tasks that:

- are computation intensive
- are through-put biased (large numbers of similar, independent jobs)
- run well on commodity hardware

We'll look at an example (alright, *toy* example) – raytracing.

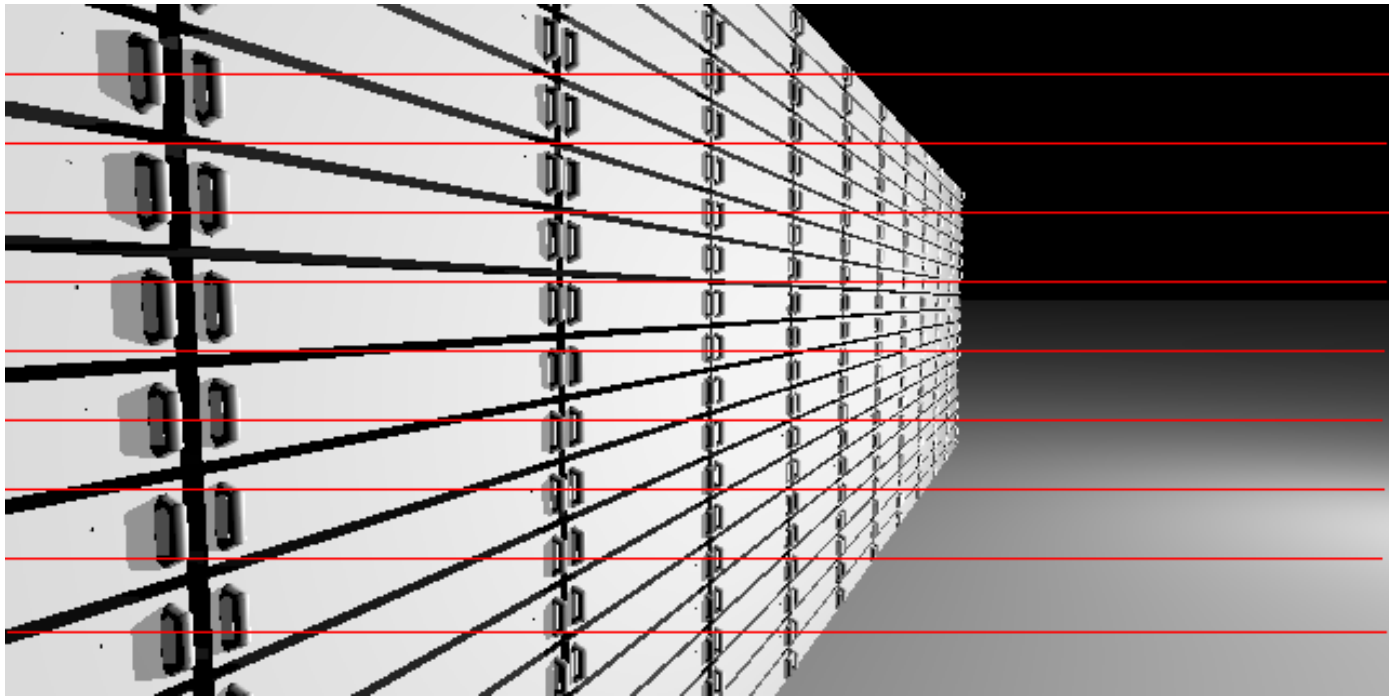
Example: Raytracing

Raytracing is good, since it is an embarrassingly parallel task; the computation for determining the colour of a pixel in the image is independent for each pixel.

This means we can decompose the rendering of an image into a number of independent computations.

Example: Raytracing

Decomposition of a model of a rackmounted cluster:



We can submit each slice as a separate grid job.

Brief overview of running grid jobs

First, establish your grid credentials. Then, for each slice of the image:

1. Create a job description, using XRSL, that specifies:
   - input and output files,
   - what command to run,
   - requirements on the machine that will run the job,
   - and so on. . .
2. Submit the job to the grid. It will automatically be sent to the most suitable machine.
3. Wait until execution is finished.
4. Retrieve the output files.

When all slices are retrieved, paste them into a composite image.

Meanwhile, let's step back for a moment. . .

What are people actually trying to achieve, and why?

Two facts

A) Today, we have large amounts of resources connected through high bandwidth networks;

- Computing resources
- Storage and databases
- Instruments
- All sorts of stuff, really

But they all have their own access mechanisms. (Just think of the differences between PDC and NSC.)

B) Computing tasks are getting huge. (LHC will generate approx. 5 PB/yr)

A new utility

The goal of the grid

Connect all these resources. Provide a uniform and seamless way of accessing them.

The long-term vision

The resources should "just be there" when you need them. Computing should be just another utility like tap water or electricity.

Sounds simple, right?

A brief timeline

- Interest in parallel and distributed computing in the 80s.
- Testbeds connecting American supercomputing centres in the late 90s.
- Foster, Kesselman: "The Grid: Blueprint for a New Computing Infrastructure", 1998.
- Lots of software development – Globus, Legion, Condor, NWS, SRB, NetSolve, AppLes, Unicore – but not much actual use.
- Around 2000 grid interest starts to boom. High energy physics driving force – especially CERN. Globus toolkit becomes leading middleware.
- 2002–2003: Commitment from big players in industry. Hype levels are rising. OGSA/OGSI standards emerge.

People have been thinking of these things for a long time. Many of the problems involved are non-trivial.

To formalize things. . .

A grid is a *decentralized* system spanning *multiple administrative domains*, where both the set of users and the set of resources vary dynamically.

It can handle extremely large numbers of systems and volumes of data.

It provides *uniform* access to *heterogenous* systems, both from the point of the user and from the point of the user's application.

It provides a flexible mechanism for *locating resources* based on the user's criteria.

It provides *non-trivial quality of service*.

Don't believe what they say

There is today no such thing as a grid in actual operation. [2]

However, there are many testbeds, some of which seem to work reasonably well, within certain limits.

We will focus on one of these, **Nordugrid**.

[2] Sun's claim of 6,000 operational grids is, frankly, b*llsh*t. They have a very... *creative* definition of the word "grid".

Nordugrid

The Nordugrid project started out in 2001 with a small project team.

The team began building a grid middleware based on the Globus version 2 toolkit[3], with the focus on getting something working fast. "Start with simple things that work and proceed from there."

The Nordugrid middleware has been used to build a testbed, currently encompassing approx. 1000 CPUs.

Swegrid will use the Nordugrid middleware.

[3] I.e. *not* OGSA based.

Whom can we trust?

One of the basic problems in building a grid is how to authenticate users and machines throughout the grid.

The Globus, and hence Nordugrid, approach is to use certificates (X.509 style) based on public key cryptography. The security subsystem is known as GSI, Grid Security Infrastructure.

Public key cryptography

In public key cryptography, you have two keys;

- a private key, which you keep secret, and never share with anybody
- a public key, which is published for the whole world to see

Anyone can send a secret message to you by encrypting it with your *public* key. It can then only be decrypted by using your *private* key.

Contrariwise, you can sign documents with your *private* key. Anyone can then verify that signature using your *public* key.

Who is who?

But there is a problem here; how do we know that a certain public key really belongs to a certain person?

The solution is to introduce a trusted third party – the Certificate Authority, or CA for short.

The CA signs public keys of users, thereby certifying that they belong to certain individuals.

The public key and the CA's signature together form a *certificate*.

(You still need to get hold of the CA's public key in a secure manner, though.)

Not just persons

Not only do we need to identify persons – we also need to identify the remote machines we speak to. Hence *server certificates* are issued to machines in the same way as to individuals.

Server certificates work the same way as personal certificates, except that they point to a hostname instead of a person.

Be paranoid!

The private key must be kept private.

Really private.

And I mean *private*.

Delegation

Grid resources need to do stuff on behalf of the user – like reading and writing files using the privileges of the user.

We certainly don't want to transfer our private key to the remote resource.

The solution is that the user issues a time-limited certificate giving the holder the same rights as the user. This *proxy certificate* is signed by the user's private key.

A bit of paranoia is needed in dealing with the proxy certificate. We don't want nasty people to get hold of one of our proxy certificates. Still, proxy certificates are not quite as sensitive as your private key.

Authorization

Users and resources are grouped into VOs, virtual organizations. Each VO has a VO manager that decides who can be a member of the VO.

Resource owners can decide authorization based on VO membership, and don't need to keep track of individual users.

(Instead of saying "This resource can be used by users $U_1, U_2, U_3, \ldots U_n$", you can say "This resource can be used by the members of the virtual organization X".)

Data handling

Another problem is handling the large data sets that you often have in the grid world. (In fact, it is often the size of the data sets that force users onto the grid in the first place.)

Nordugrid offers two main tools for this;

- gridftp
- data replication

Gridftp

Gridftp (earlier known as gsiftp) is based on the ordinary ftp protocol.

Gridftp uses GSI for authentication and authorization.

Data transfer can be made using several TCP streams in parallel for increased bandwidth.

Data transfers can be initiated by a third party; a client can transfer data between two servers, without needing the data stream to go through the client.

Nordugrid uses gridftp where ever data needs to be moved around.

Data replication

Data sets can be replicated to:

- achieve higher availability – servers do crash sometimes
- keep the data close at hand – avoid repeatedly downloading the data over long distances

A Replica Catalog keeps track of the various physical copies of each replicated file.

Resource location

So, OK, we've got tools to handle the across-administrative-domains stuff, and to handle large data sets, but how do we actually find the resources we need?

- Information system
- Resource specifications

The Nordugrid information system

- Based on the Globus MDS component. Uses the standard LDAP protocol.
- Each grid unit (cluster, storage element, . . .) runs its own Grid Resource Information Server (GRIS).
- Each GRIS registers to a national Grid Information Index Server (GIIS).
- The national GIISes register to a set of top-level redundant GIISes.

The result is a hierarchical, dynamic system – only the top-level GIISes need to be known to clients. Caching is used at each level to increase performance.

The Nordugrid information system

Example: Which clusters in Sweden have i686 class processors?

```
$ grid-info-search -h grid.quark.lu.se \
-b mds-vo-name=sweden,o=grid -x \
nordugrid-cluster-architecture=i686 \
nordugrid-cluster-name

# farm.hep.lu.se, Sweden, grid
dn: nordugrid-cluster-name=farm.hep.lu.se,Mds-Vo-name=...
nordugrid-cluster-name: farm.hep.lu.se

# seth.hpc2n.umu.se, Sweden, grid
dn: nordugrid-cluster-name=seth.hpc2n.umu.se,Mds-Vo-name=...
nordugrid-cluster-name: seth.hpc2n.umu.se
                    :
                    :
```

Don't worry

There is also a nice web interface to the information system, the Grid Monitor, at `http://www.nordugrid.org/`.

Resource specification

The resource specification contains two types of information:

- It describes the job to be run
- It specifies what resources are needed, in terms of CPU-time, installed software, operating system, and so on.

The latter information is used to find a suitable machine to run on, the former is used to actually run the job.

Resource specifications are written using the Extended Resource Specification Language (XRSL).

Resource specification – Hello World

The grid version of Hello World:

```
&(executable="/bin/echo")
 (arguments="Hello Grid!")
 (stdout="hello.txt")
 (gmlog="debuginfo")
 (jobname="My Hello Grid")
 (cputime=1)
```

Resource specification – I/O

Input and output files can be specified in a flexible manner:

```
(inputFiles=(file1.dat http://example.com/test.dat)
            (file2.dat gsiftp://niak.nsc.liu.se/myfile.dat)
            (file3.dat rc://dc1.uio.no/2000/his/atlas.dat)
            (file4.dat /scratch/bigfile.dat)
            (file5.dat ""))

(outputFiles=(output1.dat gsiftp://example.com/result17.dat)
             (output2.dat ""))
```

Resource specification – Runtime environments

You can specify that your job needs a certain software package.

```
(runTimeEnvironment=POVRAY-3.5)
```

In this case, version 3.5 of the POVray raytracer is requested. This serves two purposes;

- The job will be submitted to a machine where POVray 3.5 is available.
- The job's environment will be properly initialized to run POVray (the `povray` binary will be in `$PATH`, and so on).

Resource specification

You can do much more in XRSL. Full details are in the documentation on `http://www.nordugrid.org/papers.html`

By the way. . .

The XRSL for one of the raytracing jobs looks like this, in case you were curious.

```
&(executable=runpov.sh)
 (arguments=101 150)
 (stdout="pov.out")
 (stderr="pov.err")
 (gmlog="pov.log")
 (jobname="monopov-101-150")
 (cputime=5)
 (runtimeenvironment=povray-3.5)
 (inputfiles=(monolith.pov "")
             (front.png ""))
 (outputfiles=(monolith-101-150.png ""))
```

The user interface

The Nordugrid user interface consists of command line tools to submit jobs and access them in various ways.

There is also a web portal. (No personal experience, though)

The user interface - proxy certificates

To create a proxy certificate:

```
$ grid-proxy-init
Your identity: /O=Grid/O=NorduGrid/OU=nsc.liu.se/CN=Leif Nixon
Enter GRID pass phrase for this identity:
Creating proxy ..................................... Done
Your proxy is valid until: Wed Oct 22 00:26:51 2003
$
```

To destroy it when you are done:

```
$ grid-proxy-destroy
$
```

(Strictly, these commands are not part of the Nordugrid User Interface – they are straight from Globus.)

The user interface - submitting jobs

The `ngsub` command locates a suitable resource (by querying the information system) and submits the job there.

```
$ ngsub -f hello_grid.xrsl
Job submitted with jobid gsiftp://toornaarsuk.distlab.diku.dk:2811/
jobs/17235972981974760487
$
```

Note that there is no central broker; all of the work matching resources to the XRSL is performed locally.

The user interface - checking on jobs

To check how the job is proceeding the ngstat command is used:

```
$ ngstat gsiftp://toornaarsuk.distlab.diku.dk:2811/jobs/17235972981
974760487
Job gsiftp://toornaarsuk.distlab.diku.dk:2811/jobs/172359729819747
0487
  Jobname: My Hello Grid
  Status: FINISHING
$
```

Thankfully, you can use the job name defined in the XRSL to specify the job:

```
$ ngstat "My Hello Grid"
Job gsiftp://toornaarsuk.distlab.diku.dk:2811/jobs/172359729819747
0487
  Jobname: My Hello Grid
  Status: FINISHED 2003-10-21 12:34:30
$
```

The user interface - retrieving results

When the job is finished, you can download its output files using `ngget`:

```
$ ngget "My Hello Grid"
Downloading files to /home/nixon/docs/grid/test/hello_grid/17235972
981974760487
ngget: Download successful. Deleting job from gatekeeper.
$ ls 172359729819747604 87/
hello.txt grid.debug
$
```

Output files are typically kept on the resource for 24 hours and then erased. Either download them before that, or specify in your XRSL file that they should be uploaded somewhere.

Conclusion

So, is the emperor naked?

Let's check. . .