Introduction to ARC and Swegrid

Leif Nixon
National Supercomputer Centre
nixon@nsc.liu.se

January 10, 2005

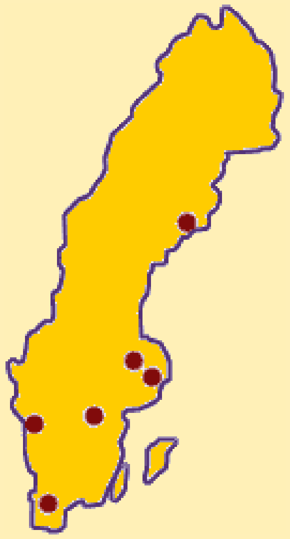
SWEGRID

Please, interrupt whenever you have questions.

What is Swegrid?

Hardware:

- Six clusters of 100 nodes each at six different sites
- Gigabit Ethernet interconnect
- Bunch of storage (approx. 60 TB)



Software:

- Different Linux versions
- Different environments, according to site policies
- ARC middleware to glue stuff together

SWEGRID

ARC

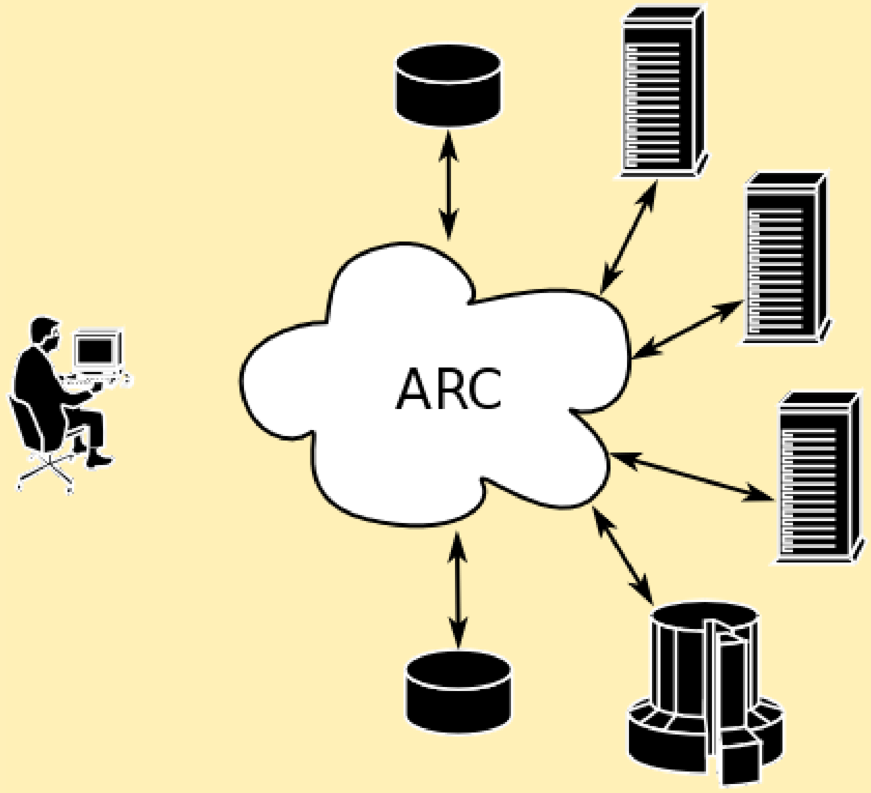
The *Advanced Resource Connector*, or ARC, is the middleware produced by the Nordugrid project.

It can be thought of as a second level batch system.

Among other things, it:

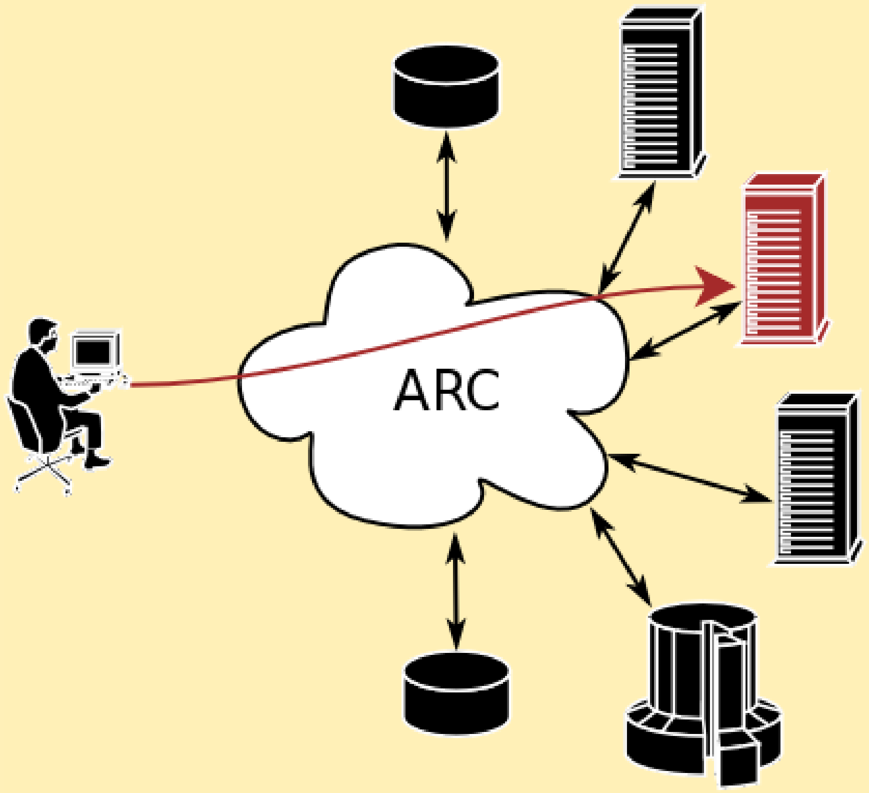
- keeps track of site configurations – OS, hardware, installed applications
- keeps track of the current load situation – available CPU:s, queue lengths, free disk space
- sends jobs to a suitable site, and submits them to the local batch system
- handles stage-in and stage-out of files

ARC



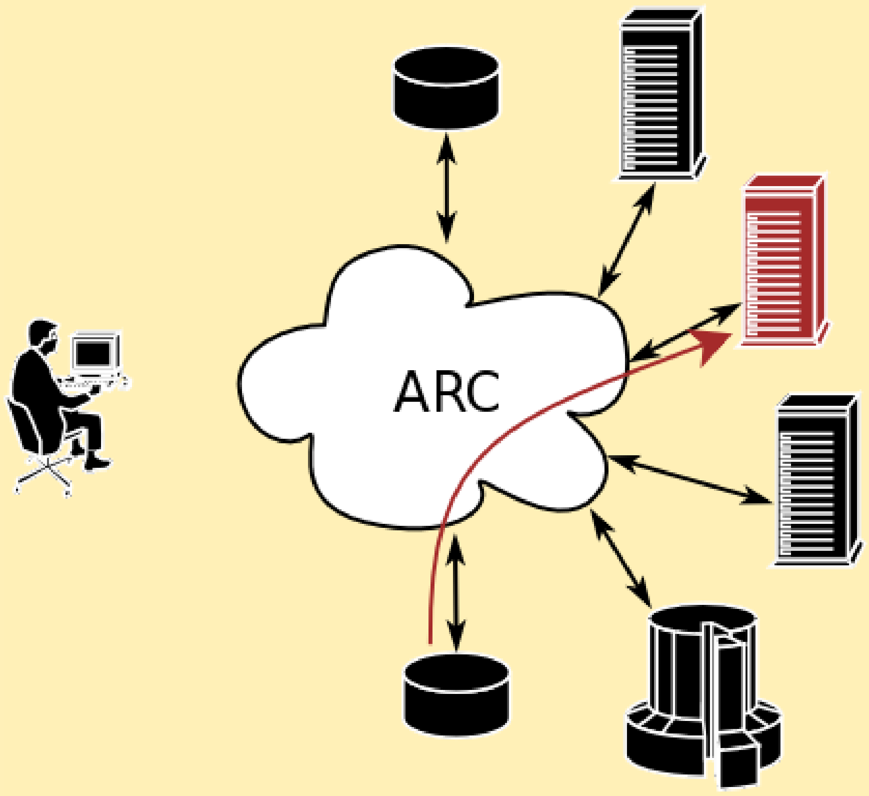
SWEGRID

ARC – job submission

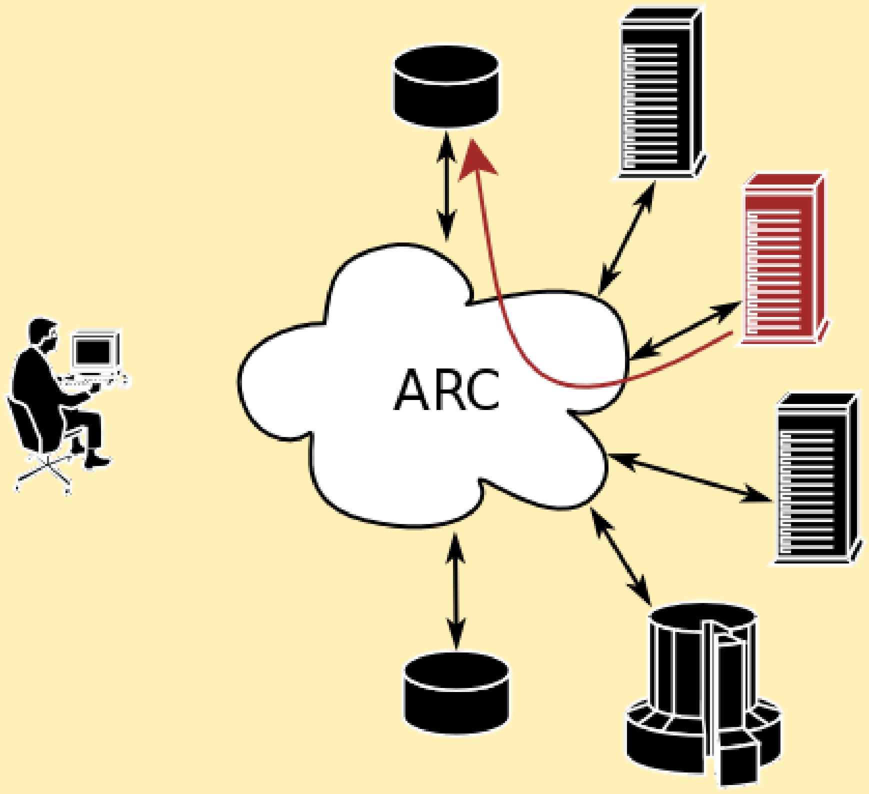


SWEGRID

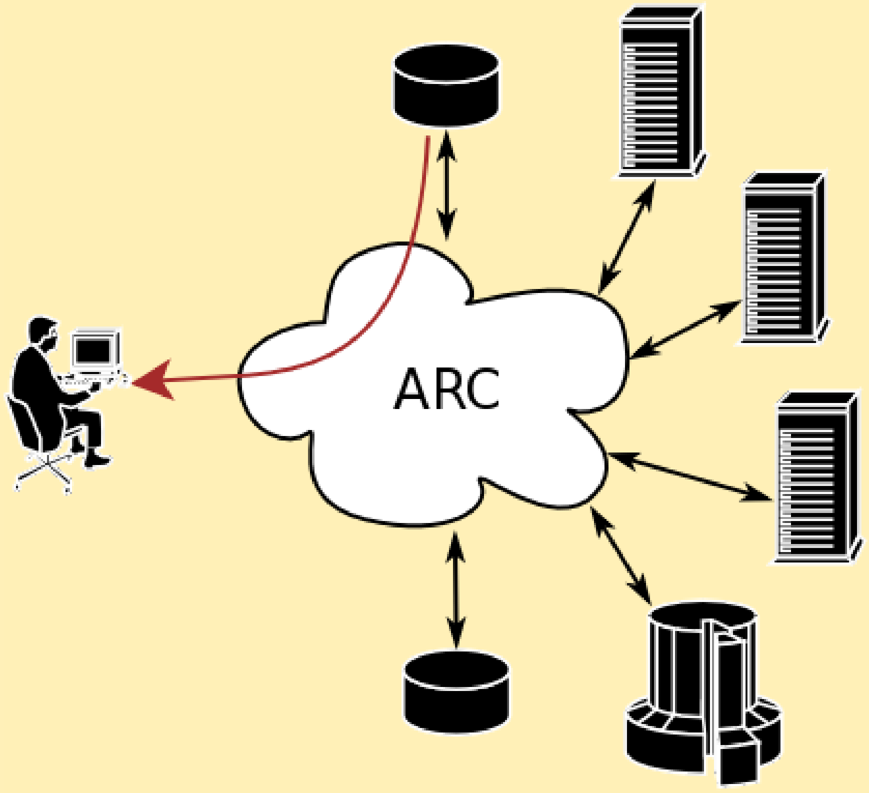
ARC – input files



ARC – output files



ARC – download



What does a job look like?

Job descriptions are written in XRSL (Extended Resource Specification Language).

A simple example:

```
&(executable="/bin/echo")
 (arguments="Hello grid!")
 (stdout="out.txt")
 (cputime="1 minute")
 (jobname="Hello Grid")
```

Job submission

Jobs are submitted using the `ngsub` command:

```
$ ngsub -f hello.xrsl
Job submitted with jobid gsiftp://bluesmoke.nsc.liu.se:2811/
jobs/29255108609821818820569 69
```

Checking on your job

You can check the progress of your job with the `ngstat` command:

```
$ ngstat "Hello Grid"
Job gsiftp://bluesmoke.nsc.liu.se:2811/jobs/2925510860982181
882056969
  Jobname: Hello Grid
  Status: ACCEPTED
```

[... *twiddling thumbs* ...]

```
$ ngstat "Hello Grid"
Job gsiftp://bluesmoke.nsc.liu.se:2811/jobs/2925510860982181
882056969
  Jobname: Hello Grid
  Status: FINISHED 2004-06-01 15:58:07
```

Retrieving results

In this case, we didn't specify any automatic upload of output files to a storage element (SE) for permanent storage:

```
$ ngstat -l "Hello Grid"
        :
        :
Results must be retrieved before: 2004-06-06 15:58:07
        :
```

Use `ngget` to download output files from temporary storage:

```
$ ngget "Hello Grid"
ngget: downloading files to /home/nixon/tmp/29255108609821818
ngget: download successful - deleting job from gatekeeper.

$ cat 29255108609821818 82056969/out.txt
Hello grid!
```

Automatic upload

We can modify the XRSL file to get the output automatically uploaded to permanent storage:

```
&(executable="/bin/echo")
 (arguments="Hello grid!")
 (stdout="out.txt")
 (cputime="1 minute")
 (jobname="Hello Grid")
 (outputfiles=
   ("out.txt"
    "gsiftp://bluesmoke.nsc.liu.se/se1/nixon/out.txt"))
```

Automatic upload

Let's try it:

```
$ ngsub -f hello.xrsl
Job submitted with jobid gsiftp://benedict.aau.dk:2811/jobs/
26661108610007320835927 57
```

[... *twiddling thumbs* ...]

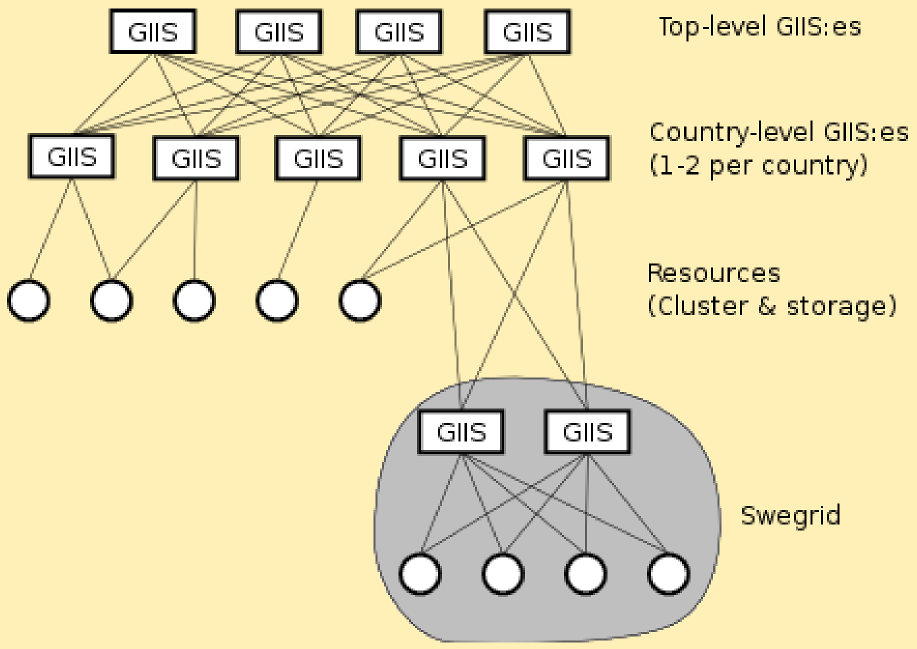
```
$ ngstat "Hello Grid"
Job gsiftp://benedict.aau.dk:2811/jobs/266611086100073208359
2757
  Jobname: Hello Grid
  Status: FINISHED 2004-06-01 16:29:29

$ ngget "Hello Grid"
ngget: downloading files to /home/nixon/tmp/266611086100073208
ngget: download successful - deleting job from gatekeeper.

$ ls 266611086100073208359275 7/
$
```

SWEGRID

A side note: Swegrid vs. Nordugrid

The last job went to a cluster in Denmark, i.e. not a Swegrid cluster!

This is because Swegrid actually is a subset of the larger Nordugrid facility. As long as the conditions in the XRSL are fulfilled, the job can wind up at any resource[1].



[1] As long as you are authorized to use it. More about that later.

Automatic upload

To access files on an SE you can use, for example, `ngcopy`:

```
$ ngcopy gsiftp://bluesmoke.nsc.liu.se/se1/nixon/out.txt \
file:///tmp/out.txt

$ cat /tmp/out.txt
Hello grid!
```

Types of jobs

There are – basically – two types of jobs, those that run *preinstalled applications*, and those that run *user-supplied programs*.

Preinstalled applications are announced as *runtime environments*.

Runtime Environments

For example, you can run Gaussian '03 in parallel on 4 nodes with an XRSL like this:

```
&(runtimeenvironment="GAUSSIAN-03")
 (executable="$g03root/g03/bsd/g03l")
 (arguments="SC4H4.com")
 (count=4)
 (inputfiles=("SC4H4.com" ""))
 (outputfiles=("SC4H4.log" "")
             ("SC4H4.chk" ""))
 (cpuTime="2 hours")
 (jobname="SC4H4")
```

The runtime environment information is used both at submission time to find a suitable resource to run on, and at job start time to set up the job's environment appropriately.

SWEGRID

Running your own code

If you want to run your own program, you can either ask the site administrators to preinstall it for you and announce it as a runtime environment, or you can send the binary together with the job.

Say you have a small program, `hello`, that just prints “Hello grid!” to stdout and exits. You can then submit a modified Hello Grid job:

```
&(executable="hello")
 (stdout="out.txt")
 (cputime="1 minute")
 (jobname="Hello Grid")
```

If the executable is given as a plain file name it is assumed to be located in the directory where you do `ngsub` and is uploaded together with the job.

Running your own code

Since you don't know where your job will end up, you will need to make sure that the program is portable.

Often it is enough to just link the binary statically. If you have complex library dependencies, things can get a bit complicated, though.

Authentication and access

Brief overview[2]:

You don't have an ordinary user account and password on the grid. Instead, a system based on public key cryptography is used, where you have a certificate and a private key, which together prove your identity.

Certificates are issued by a Certificate Authority (CA). A Nordugrid CA for the Nordic countries is operated by Anders Wäänänen at the Niels Bohr Institute in Copenhagen.

To manage authorization, users are grouped into Virtual Organizations (VOs). Resource owners can grant access to a specific VO, without having to keep track of individual users.

[2]Details will follow later during the course

Finally: What's missing?

Swegrid is still in its infancy. Some things that we currently are missing:

- Good MPI support – how to make portable MPI binaries
- Automated account handling system
- Storage organization – each user should have a basic "home directory". There should probably be tape-backed long term storage.
- Framework for handling large numbers of jobs – "job babysitter".
- Good accounting