Grid Computing with NorduGrid-ARC

Balázs Kónya, Lund University,
NorduGrid Collaboration,

Dapsys 2004,

Budapest, 19 September 2004

Nordic Testbed for Wide Area Computing and Data Handling

outline

- *(Introduction to Gridcomputing)*
- Quick Introduction
- Overview of the architecture and the middleware
- First steps on the Grid with ARC
  - Logging into the Grid: dealing with certificates
  - User Tools
    - Obtaining the software
  - What is on the Grid?
  - Grid jobs: Overview of a Grid session
  - Exercises (demos)

for the inpatients: www.nordugrid.org/documents/ngclient-install.html

Nordic Testbed for Wide Area Computing and Data Handling

Quick introduction

- NorduGrid is a collaboration by universities in Denmark, Estonia, Finland, Norway and Sweden (so far)
- NorduGrid developed and implemented a ***real*** Grid system based on the **ARC middleware,** working non-stop since May 2002
- To the date, this Grid spreads from Norway to Australia to Canada to Japan
- This Grid is used for real problem solution (physics, meteorology, genomics, chemistry etc): ***not a test project, but a true system***

Nordic

Grid Monitor - Mozilla

2004-09-16 CEST 10:06:05

Processes: Grid Local

Country	Site	CPUs	Load (processes: Grid+local)	Queueing
Australia	Charm (UniMelb)	19	0+0	0+0
	Alfred (UniMelb)	88	0+37	2+1
Denmark	DistLab (DIKU)	10	0+0	0+0
	Aalborg Grid Gateway	46	46+0	301+0
	Horseshoe (DCSC/SDU)	540	3+220	6+141
	HEPAX1	1	0+0	0+0
	Morpheus	18	17+0	83+0
	VCR (VideoRecorder)	1	1+0	1+0
Estonia	UT IMCB Anakonda clus>	15	0+0	0+0
	CMS on CERN Linux	1	0+0	0+0
	CMS Production server	5	0+0	0+0
	UT DOUG Cluster	2	0+0	0+0
	CMS test cluster	1	0+0	0+0
	EENet cluster	6	0+0	0+0
	UT Physics Cluster	0		0+0
Finland	CSC Kirppu	1	0+0	0+0
	Mill (Physicum)	52	0+7 (queue down)	6+4
	Alpha (HIP)	1	0+0	0+0
	Testbed0 (HIP)	1	0+0	4+1
Germany	FZK cluster	884	0+701	0+0
	LRZ cluster	234	0+224	0+207
Norway	Oslo Temp Cluster	13	0+0	2+0
	Parallab IBM Cluster	58	9+10	0+0
	Oslo Grid Cluster	41	15+5	20+0
	UiO Grid	99	65+34	35+59
Slovakia	UPJS GRID	1	0+0	0+0
Slovenia	SiGNET	51	1+50	9+0
Sweden	Bluesmoke (Swegrid,NS>	98	95+0	528+0
	Kosufy farm	60	27+0	0+0
	ISV	4	4+0	1+0
	Hagrid (SweGrid, Uppm>	100	100+0	583+0
	Hive (Swegrid, UNICC)	98	3+17	0+0
	Ingrid (SweGrid,HPC2N)	101	101+0	364+0
	Monolith (NSC)	400	0+366 (queue down)	0+109
	Quark Cluster	7	2+0	0+0
	Beppe (SweGrid PDC KT>	96	91+0	62+0
	Sigrid (SweGrid, Luna>	100	0+49	0+48
	Toto7/Whenim64 (Lunar>	192	0+188	0+256
Switzerland	Test cluster at DPNC	2	0+0	0+0
TOTAL	39 sites	3447	580 + 1908	2007 + 826

A realistic picture of a true Grid

Hands-on Experience on a real Grid

Nordic Testbed for Wide Area Computing and Data Handling

ARC: general overview

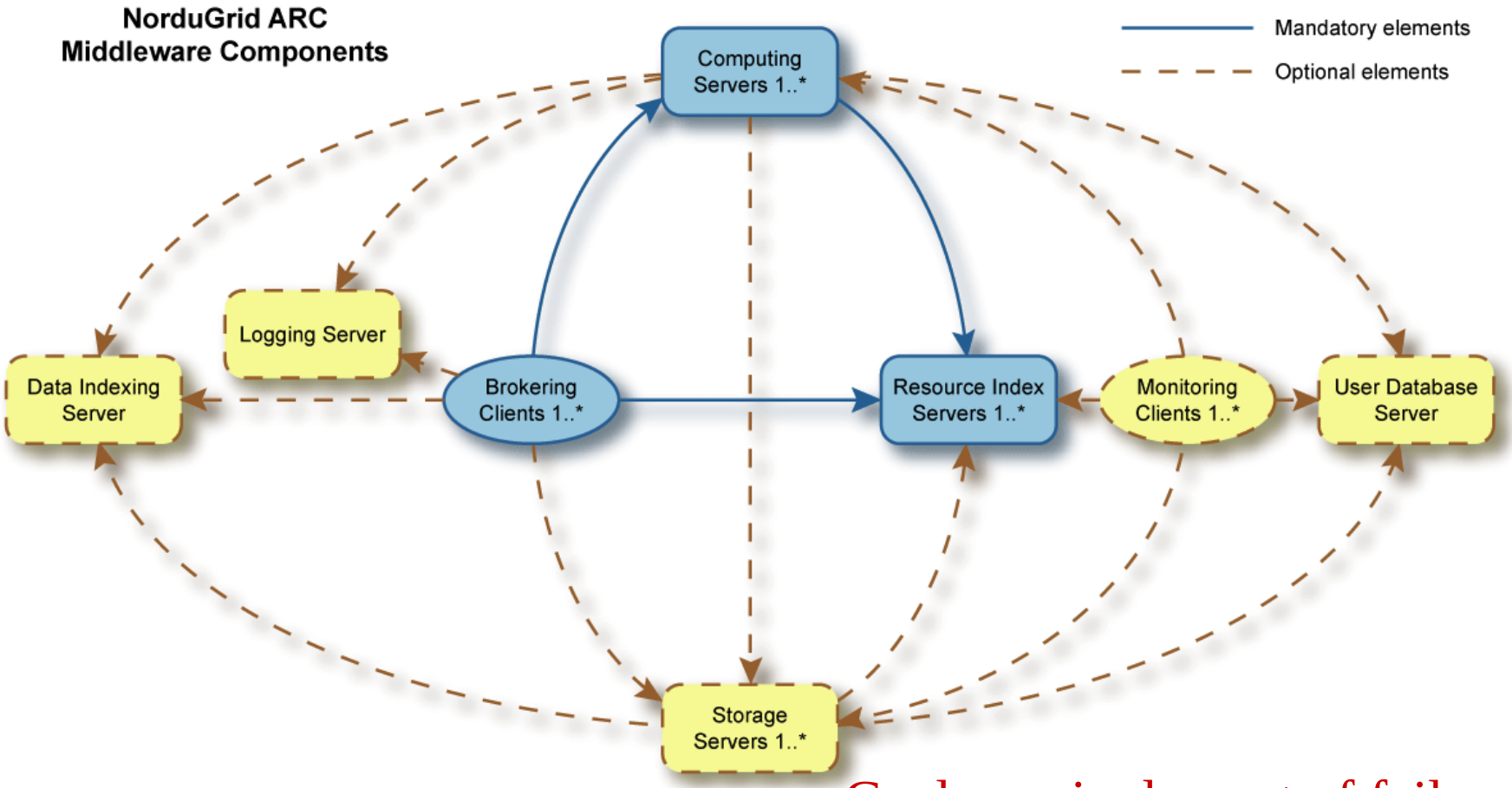
- Provides reliable implementation of fundamental Grid services:
  - Job submission (direct or via brokering), job management and monitoring
  - Information services: resource aggregation, representation, discovery and monitoring
  - Logging service
  - Data management functionality
  - integrates computing and storage resources via a secure common Grid layer
- Built upon standard open source solutions, makes use of standard protocols
  - Relies on Globus Toolkit® 2 API and libraries but makes minimal use of GT2 provided services and utilities
  - OpenLDAP, OpenSSL, SASL, SOAP, GridFTP, GSI

Nordic Testbed for Wide Area Computing and Data Handling

ARC: it is not Globus

- ARC is built upon the GT2 (pre-WS) libraries and partially makes use of the GT2 framework, BUT
  - ARC implements its own set of core Grid services, original GT2 solutions are replaced!
    - No **GRAM**!, no Globus-Gatekeeper, no Globus-jobmanager, no GT2 information model (MDS schema), no Globus Gridftp-server, no GT2 usertools
  - Innovative ARC solutions:
    - Grid-manager, ARC Gridftpd, SSE, Userinterface & Broker, Information model and providers, Monitoring, Logging, XRSL
  - ARC is a Globus library-based middleware therefore it heavily depends on GT2 as an external software
    - Actually this limits our portability
    - Nordugrid contributed a lot of fixes to pre-WS GT

Architecture: ARC functional components

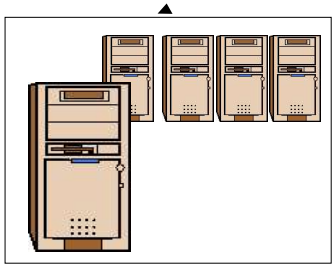


Goal: no single pont of failure

Architecture explained

- Dynamical, heterogeneous set of resources
  - **Computing**: Linux clusters (pools) or workstations, SMPs
    - Oriented towards batch jobs
    - a gateway solution permits the addition of exotic resources too
  - **Storage**: disk storage (no tape storages offered so far)
- Each resource is connected to the Grid via services running on the front-end (preserved local autonomy behind the frontend)
  - Custom **GridFTP server** for all the communications (including job submission!)
  - **Grid-Manager,** an interface to the local system
  - Local **information service**: a special LDAP Database (so-called *GRIS*)
- Resources are dynamicaly linked together via Indexing Services
  - Hierarchical multi-rooted customised tree topology implemented via LDAP registrations and a stripped-down special LDAP-backend (so-called GIISes)
  - Data indexing services (Metadata or Replica catalogues)
- Lightweight brokering clients perform resource discovery, matchmaking and job submission independently
- Auxiliary management services: User, Usage or resource Allocation

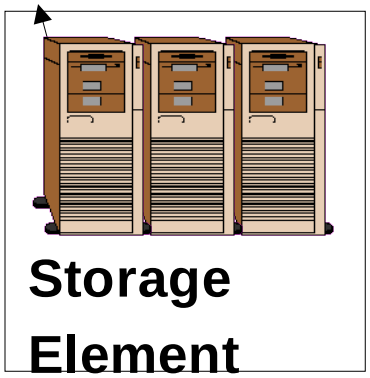
ARC components: Grid layer on a computing resource



- Computing resource is usually a cluster of PCs managed by a batch system (PBS, SGE, Condor, Fork...)
- Grid jobs are submitted through a custom **gridftpd plugin**
- Runs a service (grid daemon) called **grid manager** responsible for local job management (e.g. Job submission to local batch system). It is capable to manage pre- and post-stageing of Grid data, optionally using Metadata Catalogs.
- Provides a scratch disc space "**session directory**" and "**cache**" for grid job's data
- Grid jobs are isolated in their "**session directory**", this directory is available through **gridftpd!**
- **Runtime environment** support
- Runs a **local information service** which (LDAP) and registers to some Resource Index Service.
- Grid services are only installed on the frontend!

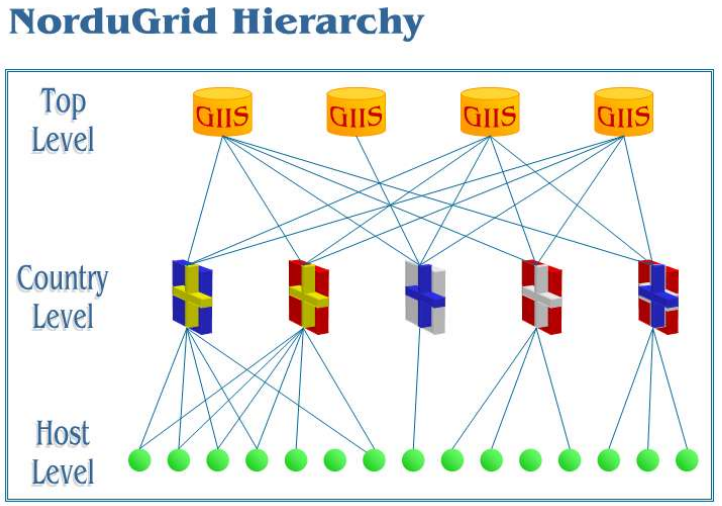
ARC components: Grid layer on storages

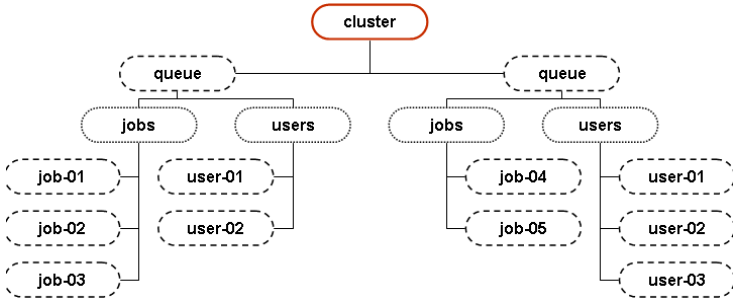
- **"Classical" Storage Element**
  - Usually GridFTP server.
  - Any other protocol supported by available tools can be used.
  - It's just a shelf where users put their files.
  - Several authorization solution: “unix file permission based, Grid Access Control List (GACL) based
- **"Smart" Storage Element (SSE)**
  - Curently being developed
  - More standard protcols: HTTPS/G, SOAP
  - Flexible access control
  - Data integrity between resources
  - Support for data replication
- Storages can be registered to Information or Metadata Indices



ARC components: Information System

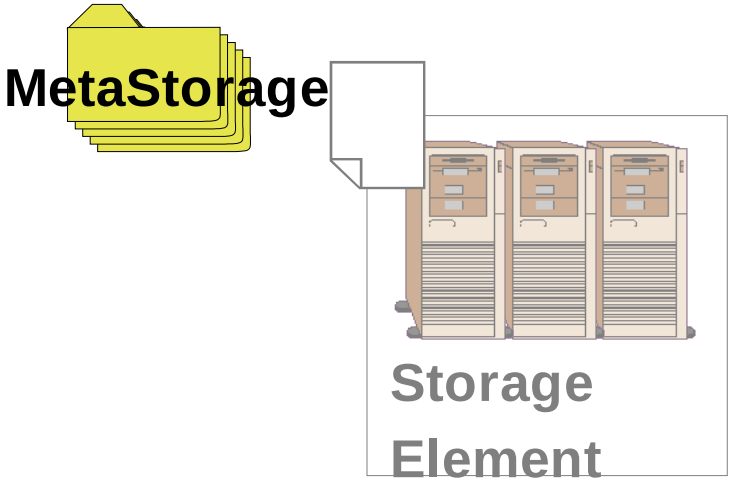
- Built upon Openldap and Globus' GIIS/GRIS backends
  - Planning to use native Openldap
- **Information indices** form a redundant hierarchical topology
  - Store the contact URLs of local information services
- **Local information service**
  - Information model (schema) represents
    - Clusters, Grid jobs, Grid users
  - Efficient **Information collectors** fill the information model with data
  - Runs on every resource (cluster and SE):
  - pull model with cacheing





ARC Components: support for metadata catalogues

- Metadata Catalogues or Data Indexing services are Databases to store information about distributed data instances
- Currently ARC supports the following two Globus products:
  - **Replica Catalog**: scalability and stability problems, but fairly reliable (LDAP DB)
  - **Replica Location Service**: was very unstable and unreliable (fixed by NorduGrid) but fashionable, requested by users (MySQL DB)



ARC components: User Interface & Broker

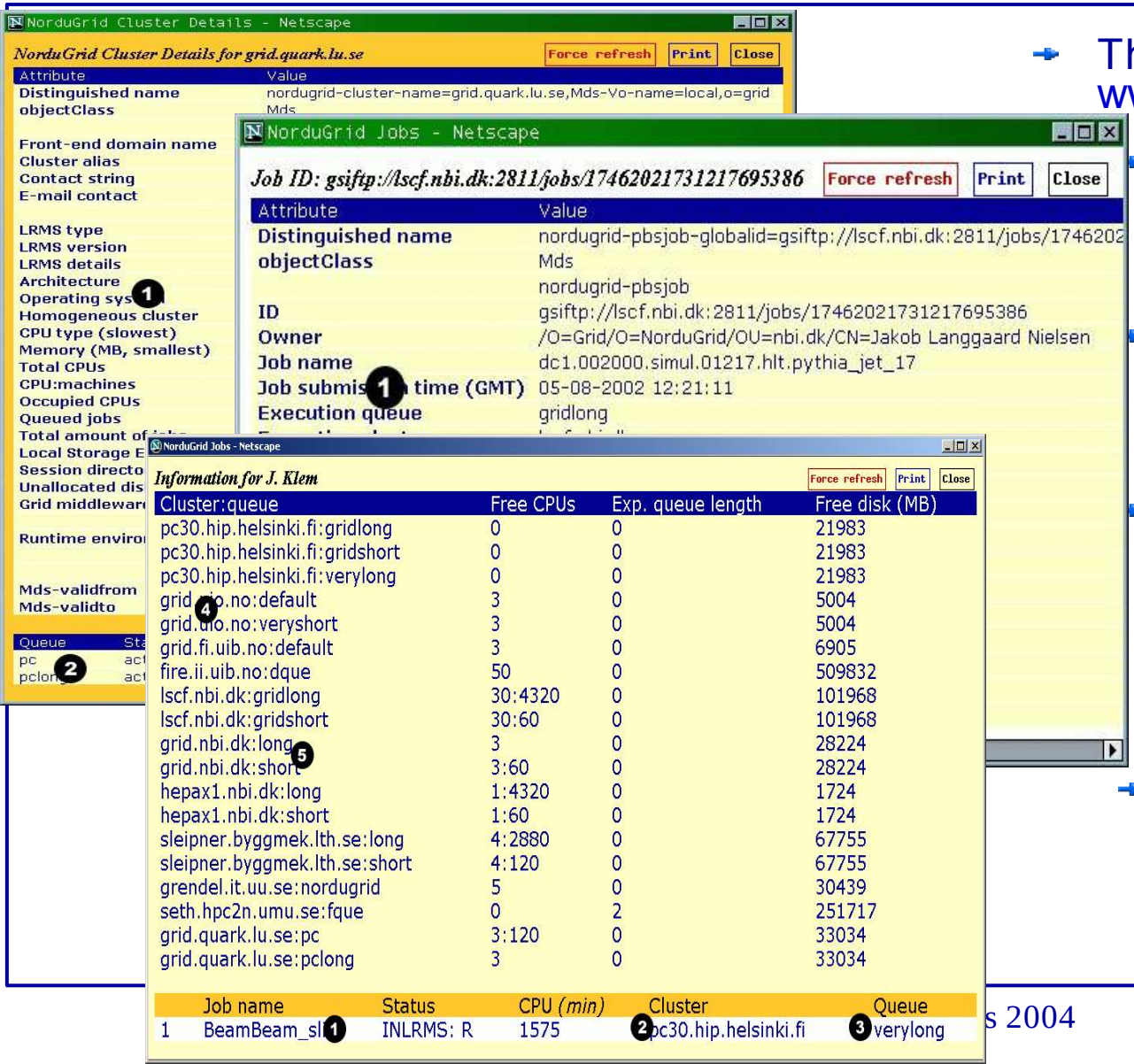
User Interface

- Provides a set of utilities to be invoked from the command line:

ngsub	to submit a task
ngstat	to obtain the status of jobs and clusters
ngcat	to display the stdout or stderr of a running job
ngget	to retrieve the result from a finished job
ngkill	to cancel a job request
ngclean	to delete a job from a remote cluster
ngrenew	to renew user's proxy
ngsync	to synchronize the local job info with the MDS
ngcopy	to transfer files to, from and between clusters
ngremove	to remove files

- Contains a personal broker that polls Infosys and decides to which queue at which cluster a job should be submitted
- Fully **decentralized model**, no central broker, no central UI
- Light-weight set of commands, collection of tools to control job's execution from submission to retrieval of results
- Additional tools to handle data files at Storage Elements and MetaStorage, plus a complete test suite (**ngtest**)
- Every user can run his own UI(s), or **switch between UIs**, job information is kept in the Grid and not on the UI
- Communicates via **XRSL**

ARC components: Grid monitor



- The Monitor is available at www.nordugrid.org/monitor
  - PHP4 client, visualization tool for the distributed Information System
  - No caching, real time LDAP queries (try to run it in debug mode)
  - Provides information on grid jobs, status of resources (clusters, storages) and active users.
  - localized so far in 3 languages

ARC components: User management, logging

- User Management:
  - User lists are periodically pulled by the resources in order to generate local synchronized grid-mapfiles
  - The lists can be fetched from anything ranging from an HTTPS-served text file to an LDAP database, to VOMS
  - Currently we have ca 20 user lists in total (over 800 potential users)
- Logging service:
  - job provenance database,
  - Reliably filled by Grid Manager with the job usage record
  - Both the user and the resource owner can specify a logger database

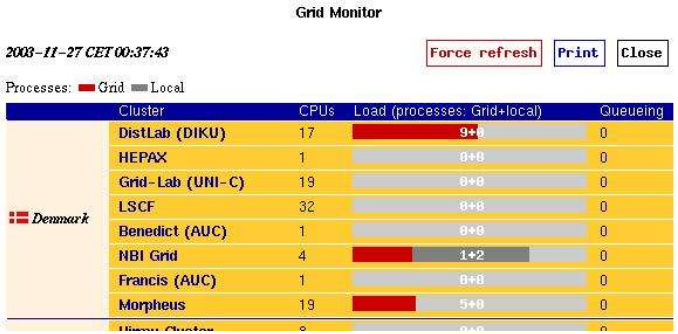
ARC components: XRSL Job Description Language

```
(&(executable="recon.gen.v5.NG")
(arguments="dc1.002000.lumi02.01101.hlt.pythia_jet_17.zebra" "dc1.002000.lumi02.recon.007.01101.hlt.pythia_jet_17.eg7.602.ntuple"
        "eg7.602.job" "999")
(stdout="dc1.002000.lumi02.recon.007.01101.hlt.pythia_jet_17.eg7.602.log")
(stdlog="gridlog.txt")(join="yes")
(
|(&(|(cluster="farm.hep.lu.se")(cluster="lscf.nbi.dk")(*cluster="seth.hpc2n.umu.se"*)(cluster="login-3.monolith.nsc.liu.se"))
(inputfiles=   ("dc1.002000.lumi02.01101.hlt.pythia_jet_17.zebra"
        "rc://grid.uio.no/lc=dc1.lumi02.002000,rc=NorduGrid,dc=nordugrid,dc=org/zebra/dc1.002000.lumi02.01101.hlt.pythia_jet_17.zebra")
        ("recon.gen.v5.NG" "http://www.nordugrid.org/applications/dc1/recon/recon.gen.v5.NG.db")
        ("eg7.602.job" "http://www.nordugrid.org/applications/dc1/recon/eg7.602.job.db")
        ("noisedb.tgz" "http://www.nordugrid.org/applications/dc1/recon/noisedb.tgz"))
)
(inputfiles=   ("dc1.002000.lumi02.01101.hlt.pythia_jet_17.zebra"
        "rc://grid.uio.no/lc=dc1.lumi02.002000,rc=NorduGrid,dc=nordugrid,dc=org/zebra/dc1.002000.lumi02.01101.hlt.pythia_jet_17.zebra")
         ("recon.gen.v5.NG" "http://www.nordugrid.org/applications/dc1/recon/recon.gen.v5.NG")
        ("eg7.602.job" "http://www.nordugrid.org/applications/dc1/recon/eg7.602.job"))
)
(outputFiles=   ("dc1.002000.lumi02.recon.007.01101.hlt.pythia_jet_17.eg7.602.log"
        "rc://grid.uio.no/lc=dc1.lumi02.recon.002000,rc=NorduGrid,dc=nordugrid,dc=org/log/dc1.002000.lumi02.recon.007.01101.hlt.pythia_jet_17.
        eg7.602.log")
         ("histo.hbook"
        "rc://grid.uio.no/lc=dc1.lumi02.recon.002000,rc=NorduGrid,dc=nordugrid,dc=org/histo/dc1.002000.lumi02.recon.007.01101.hlt.pythia_jet_1
        7.eg7.602.histo")
        ("dc1.002000.lumi02.recon.007.01101.hlt.pythia_jet_17.eg7.602.ntuple"
        "rc://grid.uio.no/lc=dc1.lumi02.recon.002000,rc=NorduGrid,dc=nordugrid,dc=org/ntuple/dc1.002000.lumi02.recon.007.01101.hlt.pythia_jet_
        17.eg7.602.ntuple"))
(jobname="dc1.002000.lumi02.recon.007.01101.hlt.pythia_jet_17.eg7.602")
(runTimeEnvironment="ATLAS-6.0.2")
(CpuTime=1440)(Disk=3000)(ftpThreads=10))
```

ARC: User Work flow

- The User:
  - prepares her job-description in the xRSL job-description language.
  - submits the xRSL to the NorduGrid resources using the user-interface
  - While the job is running, she can query the status of her jobs.
  - When the jobs have finished, she can download the output of jobs -- or the output can be placed on permanent storage directly.
- Meanwhile the components of the Grid do their job:
  - The brain of the Grid, the client "UserInterface" does resource discovery, brokering, Grid job submission and monitoring
  - The Information system, the nervous system of the Grid answers the queries of the UI and the monitoring tools
  - The "heart"(s) of the Grid, the **Grid Manager**(s) perform data movement, keeps track of job status, manages and controls session directories, prepares preinstalled software, accepts job submissions from the clients

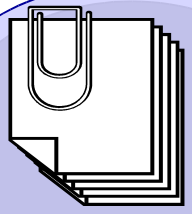
One more glimpse on ARC



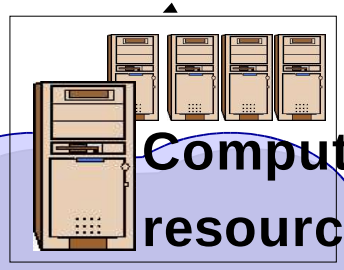
User Interface & Broker

User Interface & Broker

User Interface & Broker



Information Service

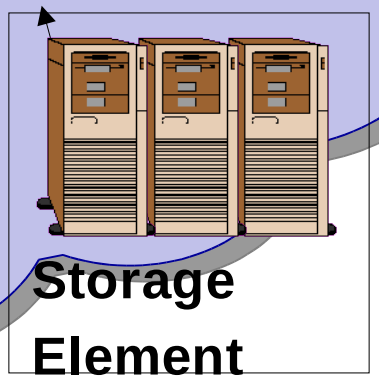


Computing resource

GSI everywhere

GSI everywhere

MetaStorage



Storage Element

Live demonstration

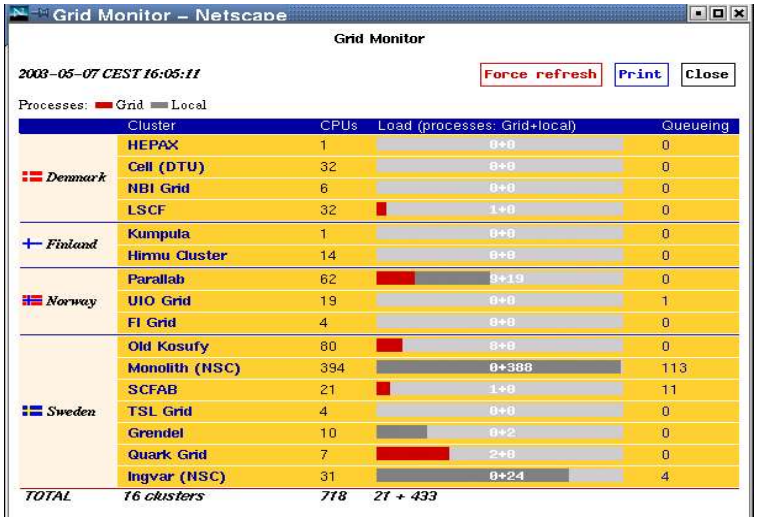
First Steps on the Grid

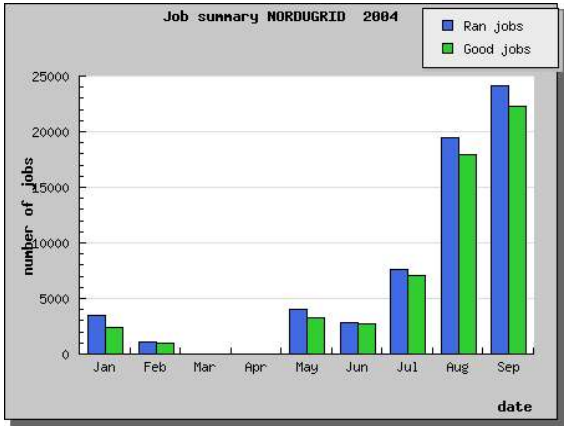
ingredients of a Grid sesion

- a functional Grid
- authorized Grid credentials
- user Toolkit or access to a Grid Portal
- description of a Grid job

ARC-based Production Grid

- Components:
  - Clusters
  - Storage Elements
  - Metadat Catalogues (file catalogues)
- What is on the Grid?
  - Grid Monitor
- What happened on the Grid?
  - Logger interface





exercise: Grid discovery

- Fire up the Grid Monitor
  - entries are clickable, clicking an entry performs an LDAP search over the Grid with respect to that attribute
  - check out the free resources for a particular user
  - Run the monitor in debug mode (www.nordugrid.org -> site registry)
- browse the NorduGrid LDAP Information Tree
  - look into entries, check attributes, walk the tree
- Use the `ngstat -q -l` UI command for getting information on clusters
- try out an `ldapsearch` command:

```
ldapsearch -h quark.hep.lu.se -p 2135 \
-b "mds-vo-name=local,o=grid" 'objectclass=nordugrid-cluster' -x dn
```

authorized credentials

Who can use the Grid?

- possess a recognized certificate
  - certificate is the Grid-ID card

```
"/O=Grid/O=NorduGrid/OU=quark.lu.se/CN=Balazs Konya"
```

  - Public Key Infrastructure (PKI X.509)
    - Certificate mini-howto
  - NorduGrid issues its own certificates but accepts certificates from other Grid projects too
- being authorized on the Grid resources
  - member of a recognized user group (Virtual Organization)
  - www.nordugrid.org --> Users

exercise: credentials

1) Check out your credentials

```
ls -l .globus/
```

2, Generate a certificate request

```
grid-cert-request -dir certdir
```

3, Modify the passphrase of your private key

```
grid-change-pass-phrase
```

4, Check the content of your credentials

```
grid-cert-info & grid-proxy-info
```

5, Log into the Grid: create your proxy

```
grid-proxy-init
```

6, Destroy your proxy and create a longer one

```
grid-proxy-destroy; grid-proxy-init -valid 48:0
```

7, Check out the NorduGrid User Info page:

```
http://www.nordugrid.org/...
```

NorduGrid client middleware

NorduGrid standalone package:

- precompiled binaries in a single tarball (~5MB)
- comes with the required Globus components and does all the necessary initial setup/configuration
- NorduGrid + Globus command line tools:
  - `ngsub, ngclean, ngget, ngremove, ngcat, ngcopy, ngkill, ngstat, ngsync`
  - `grid-proxy-*, grid-cert-*, globus-url-copy, gsincftp`

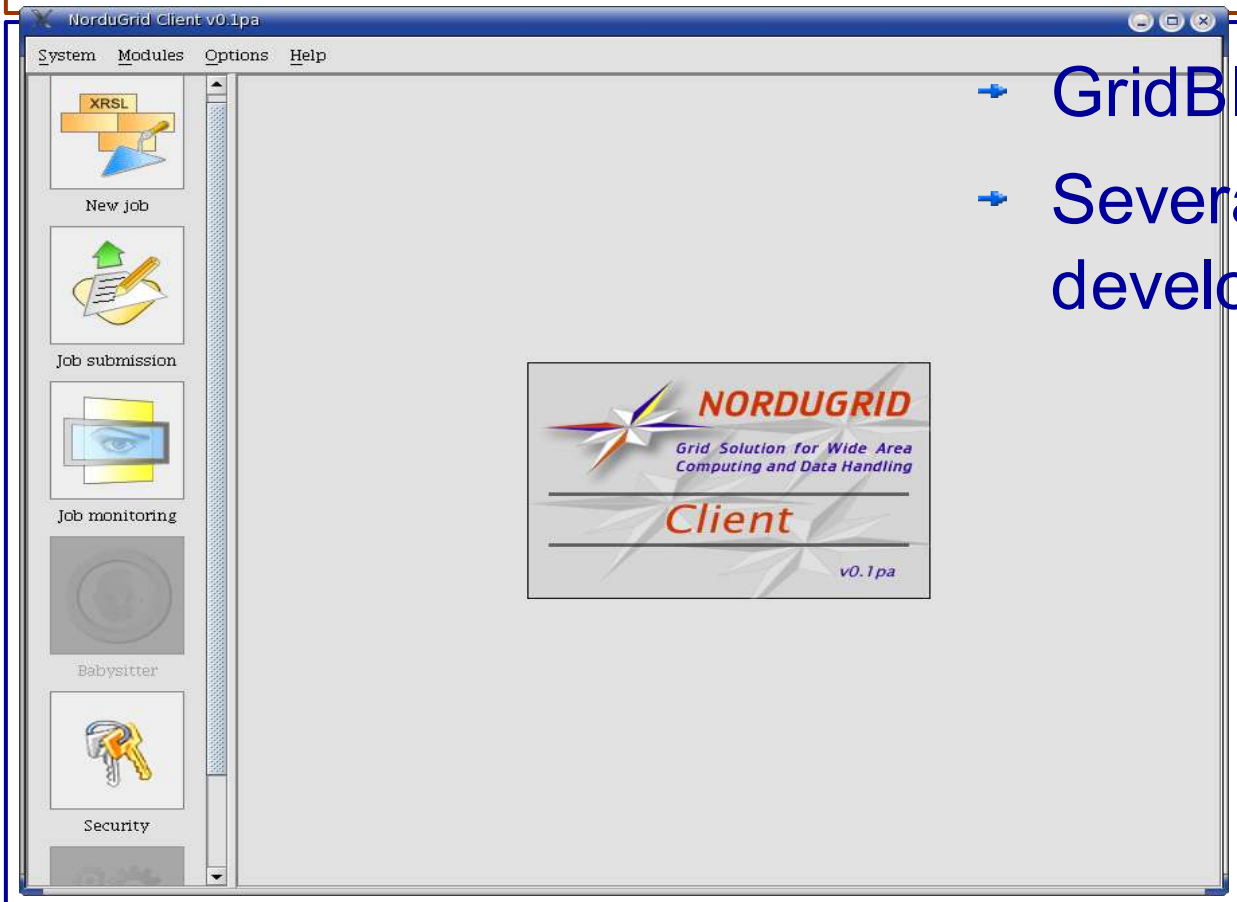
Installation steps (3 minutes):

- get the package from our download area
- unpack the tarball (~14MB), cd directory, source the setup.sh
- you are ready to fire up your certificate or generate a cert. request

Client install instructions for alternative installations:

- http://www.nordugrid.org/documents/ng-client-install.html

Graphical clients



- GridBlock Web Portal
- Several GUIs are under development

exercise: NorduGrid Middleware

1) get the NorduGrid Standalone binary
   - `nordugrid-standalone-xyz.tgz`
   - `ftp://ftp.nordugrid.org/nordugrid/releases/`
   - `or www.nordugrid.org -> Downloads -> latest release, standalone`
2) install the package
   - `tar xvzf nordugrid-standalone-xyz.tgz`
   - `cd nordugrid-standalone-xyz`
   - `source ./setup.sh`
3) get to know more about the ng-commands
   - `i.e. man ngsub`
   - `ng-command -h`

job description (XRSL*)

eXtended Resource Specification Language describes the Grid jobs:

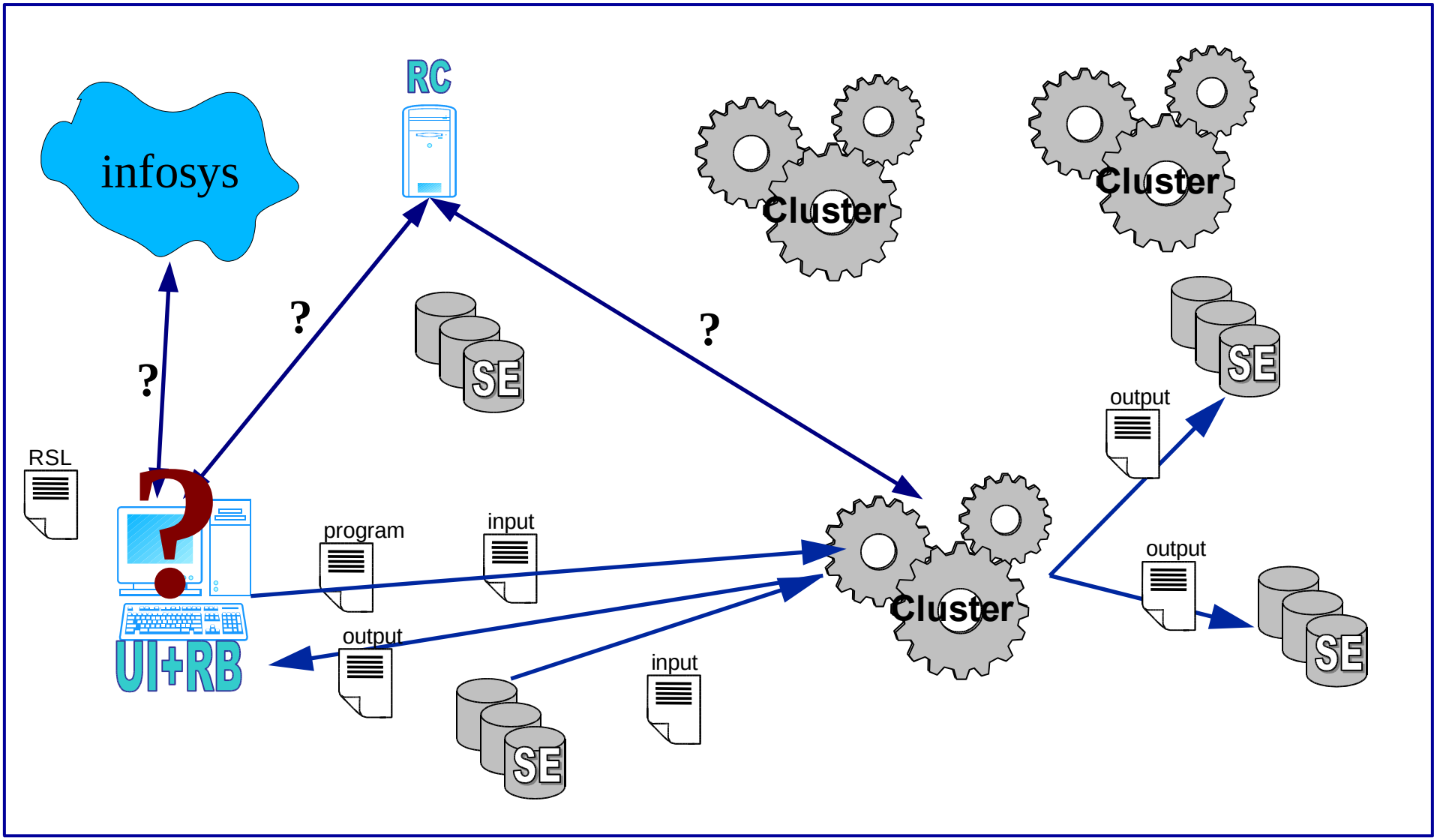
- on what kind of platform?
  - Linux cluster or something else (architecture, operating system)
  - how much memory, disk space, CPUtime is needed?
- what kind of program to run?
  - do I have my own binary what I want to upload?
  - do I request preinstalled/configured software? (RuntimeEnvironment)
- what about input files (stdin), required datasets?
- what to do with results, output files (stdout,stderr,gmlog)?

*www.nordugrid.org/documents/xrsl.pdf

overview of a Grid session

- user formulates the job requirements by editing an xrsl file
- having a valid proxy submits the job with ngsub
- the broker (built in the UI) selects the target cluster, passes the job to the GridManager (via the Gridftpd jobplugin), uploads the requested files from the submission machine
- after successful submission, a job handle (ID) is returned
  `gsiftp://seth.hpc2n.umu.se:2811/jobs/8632436256385296`
- The GM takes care of the Grid job on the cluster:
  - creates a dedicated session directory for the job
  - collects the requested input data files from the Storage Elements
  - submits the job to the Cluster Management System (PBS)
  - after job execution the GM uploads (if requested) the files to an SE
- Meanwhile the user may continuously monitor the status of the job & Grid
- after job completion the user retrieves the output from the session directory on the cluster (only those files which were not uploaded to an SE)

Grid session (animated)



“Hello Grid” exercise

```
&(executable=/bin/echo)(arguments="Hello Grid" )
(stdout="hello.txt")
(stderr="hello.err")
(gmlog="gridlog")
(jobname="My Hello Grid")
(cputime=300)
(*middleware="nordugrid-0.4"*)
```

```
&(executable=say_hello)(arguments="Hello Grid with uploaded binary" )
(stdout="hello.txt")
(stderr="hello.err")
(gmlog="gridlog")
(jobname="Say_Hello")
(cputime=300)
(*middleware="nordugrid-0.4"*)
```

Nordic Testbed for Wide Area Computing and Data Handling

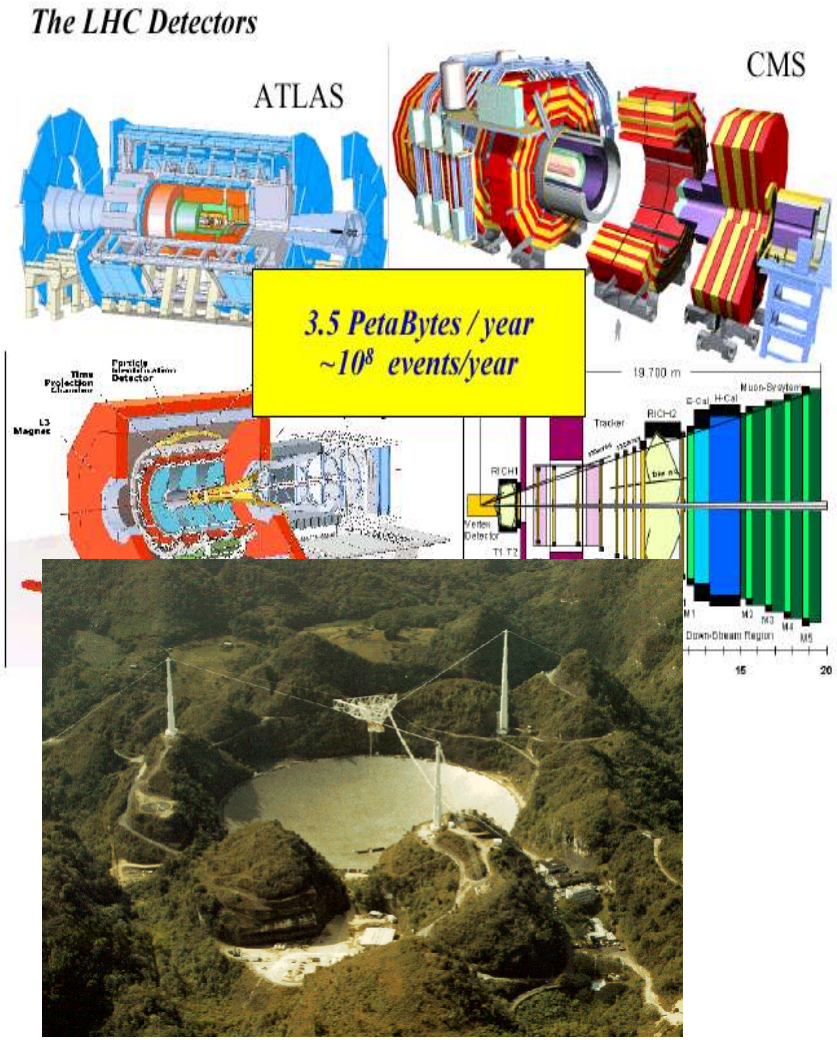
The Mandelbrot exercise

- download the mandel.tgz
- run the small program locally on your machine
  - ./generate_mandel.bin < parameters.inp
  - check out the generated figure
- look at the generated figure:
  - kview figure.ppm
- submit the same job to the Grid
  - ngsub -f mandel.xrsl -d 1
- monitor, your job, peek into the stdout
  - ngstat <jobid> ; ngget <jobid>
- submit several jobs, try to kill some, clean up the mess
  - ngkill <jobid>; ngclean -a

ngtest exercise

- Use the ngtest suite to submit jobs, ngtest -list-cases describes the available test jobs, run the default test case
  - ngtest -d 1
- Use the ngstat, ngkill, ngclean, ngget commands to check job status, kill the job, remove the job from the cluster or fetch the job output, a few example:
  - ngstat -a; ngkill <jobid>; ngget <jobid> -k
- Browse the session directory with a gridftp client:
  - gsincftp <jobid>
- Look into the xrsl files describing the test jobs
  - ngtest -d 1 -s
  - cat ngtest.xrsl

Modelling Grid-enabled science



Main driving force behind Grid:

- Sharing access to scientific instruments
- Sharing access to observed data

Simple model: Griddified Video Recorder

- "Scientific phenomena" to be observed:
  - broadcasted TV program
- "Scientific apparatus" to be used:
  - PC with a TV capture card
- Grid jobs will be used to take measurements
- Grid interface will be used to access collected data

read more: www.imada.sdu.dk/~karlsen/vcrrecord.html

brokering exercise

- imitate the jobsubmission, play with the UI without submitting real jobs (the UI performs a fake jobsubmission)
  - ngtest -d 1 -dumpxrsl -t 15
- try to follow the brokering steps described here
  - http://cvs.nordugrid.org/cgi-bin/cvsweb/nordugrid/doc/ui/brokering.pdf

Getting on the Grid (summary)

- Get a bit familiar with Grid computing, read some documentation, follow a Grid presentation (www.nordugrid.org)
- Install the “standalone” client
- Request a certificate (your grid ID)
- Obtain access to the Grid, apply for grid resources
  - In Sweden contact the SweGrid, www.swegrid.se
- Use the support system, contact your local grid expert in order to get help “gridifying” your application
- You are welcome to join the R & D projects of the NorduGrid collaboration

sources of information

- Documentations, papers, conference presentations, tutorials:

  `www.nordugrid.org -> Documentation`

- Support (ticketing service):

  `nordugrid-support@nordugrid.org`

- NorduGrid overview paper:

  `www.nordugrid.org/documents/ieee-nordugrid.pdf`