The NorduGrid Architecture And Middleware for Scientific Applications

O. Smirnova[1], P. Eerola[1], T. Ekelöf[2], M. Ellert[2], J. R. Hansen[3], A. Konstantinov[4], B. Kónya[1], J. L. Nielsen[3], F. Ould-Saada[5], and A. Wäänänen[3]

[1] *Particle Physics, Institute of Physics, Lund University, Box 118, 22100 Lund, Sweden*
[2] *Department of Radiation Sciences, Uppsala University, Box 535, 75121 Uppsala, Sweden*
[3] *Niels Bohr Institutet for Astronomi, Fysik og Geofysik, Blegdamsvej 17, DK-2100, Copenhagen Ø, Denmark*
[4] *Vilnius University, Institute of Material Science and Applied Research, Saulėtekio al. 9, Vilnius 2040, Lithuania*
[5] *University of Oslo, Department of Physics, P. O. Box 1048, Blindern, 0316 Oslo, Norway*

Abstract. The NorduGrid project operates a production Grid infrastructure in Scandinavia and Finland using own innovative middleware solutions. The resources range from small test clusters at academic institutions to large farms at several supercomputer centers, and are used for various scientific applications. This talk reviews the architecture and describes the Grid services, implemented via the NorduGrid middleware.

1 Introduction

The NorduGrid project [1] started in May 2001, initiated by academic institutes in Scandinavia and Finland, aiming to build a Nordic testbed for wide area computing and data handling. After evaluating the existing Grid solutions (such as the Globus Toolkit^{TM} [2] and the EU Datagrid (EDG) [3]), the NorduGrid developers came up with an original architecture and implementation proposal [4, 5]. The NorduGrid testbed was set up accordingly in May 2002, and is in continuous operation and development since August 2002. By now it has grown into one of the largest operational Grids in the world, including not only test sites, but also several large production clusters. The NorduGrid operates its own Certification Authority, recognized by other related projects, such as EDG.

2 The NorduGrid Architecture

The NorduGrid architecture was carefully planned and designed to satisfy the needs of a generic user, as well as those of a system administrator. The chosen philosophy can be outlined as follows:

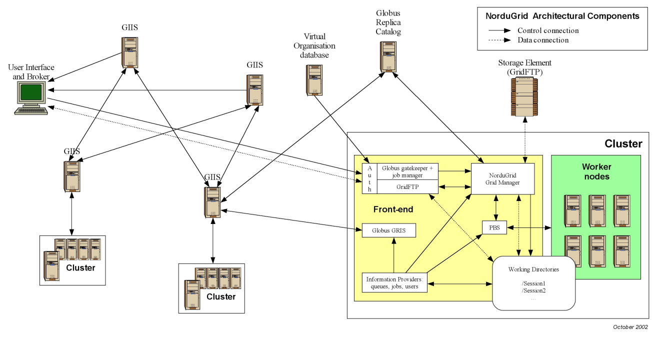


Fig. 1. Components of the NorduGrid architecture.

- Avoid architectural single points of failure
- Resource owners retain full control over their resources
- Installation details (method, operation system, configuration *etc.*) should not be dictated
- As little restriction on the site configuration as possible
  - Computing nodes should not be required to be on the public network
  - Clusters need not be dedicated
  - NorduGrid software should be installed only on a front-end machine
- NorduGrid software should be able to use the existing system and eventual pre-installed Globus versions
- Start with something simple that works and proceed from there

The NorduGrid tools are designed to handle job submission and management, user area management, a minimal necessary data management, and monitoring. Basic components of the NorduGrid architecture, as shown in Fig. 1, are:

- Computing Cluster, consisting of a Gatekeeper (the front-end) and Worker Nodes (back-end)
- Storage Element
- Replica Catalog
- Information System
- User Interface (client)

There are no special requirements imposed on computing clusters, apart of the presence of a shared filesystem (*e.g.* NFS) between the front-end and the back-end nodes. Storage Elements are basically a GridFTP [6] servers. Replica Catalog for locating and registering data sources is the one developed by the Globus project [7], with the only significant change: added possibility to perform securely authenticated connections. Information System is based on the Monitoring and Discovery Services (MDS) of Globus [8]. User Interface is just a client package, which can be installed on any machine.

Job submissions are performed via User Interfaces, which use the Information System and Replica Catalog information about Storage Elements contents to discover best resources (clusters).

3 The NorduGrid Middleware

The NorduGrid architecture is realized through innovative middleware, which has to be installed on a front-end (server part) and a user's machine (client part).

3.1 Grid Manager

The NorduGrid Grid Manager (GM) [10] software acts as a smart front-end for job submission to a cluster. It runs on the cluster's front-end and provides job management and data pre- and post-staging functionality, with a support for meta-data catalogues.

Since data operations are performed by an additional layer, the data are handled only at the beginning and end of a job. Hence the user is expected to provide complete information about the input and output data. This is the most significant limitation of the used approach.

The GM is based on the Globus Toolkit^{TM} libraries and services. It was written because the services available as parts of the Globus Toolkit^{TM} did not have the required functionality to implement the architecture developed by the NorduGrid project.

To provide a GridFTP interface for submission of specialized jobs, a more Grid-oriented GridFTP server (GFS) was developed. It has the following features:

- Virtual directory tree, configured per user.
- Access control, based on the Distinguished Name stored in the user certificate.
- Local file system access, implemented through loadable plugins (shared libraries). There are two plugins provided with GFS:
  - The local file access plugin implements an ordinary FTP server-like access,
  - The job submission plugin provides an interface for submission and control of jobs handled by the GM.

The GFS is also used by NorduGrid to create relatively easily configurable GridFTP based storage servers (Storage Elements).

The GM accepts job-submission scripts described in Globus RSL (Section 3.4). For every job, the GM creates a separate directory (the *session directory*) and stores the input files specified in the job submission script into it. There is no single point (machine) that all the jobs have to pass, since the gathering of input data is done directly on a cluster front-end.

Then the GM creates a job execution script and launches it using the LRMS. Such a script can perform additional actions, such as data staging to a computing node, and setting the environment for third-party software packages requested by a job.

After a job has finished, all the specified output files are transferred to their destinations, or are temporarily stored in the session directory to allow users to retrieve them later.

Additionally, the GM can cache input files requested by one job and make them available to another job. If a file is taken from the cache, it is checked (if the protocol allows that) against the remote server containing that file, whether a user is eligible to access it. To save disk space, cached files can be provided to jobs as soft links.

The GM is written mostly in C/C++. It uses a *bash-script* Globus-like implementation of an interface to an LRMS. This makes it easy to adapt the GM to inter-operate with any new LRMS. Currently, the GM comes with the interface to the OpenPBS only.

The GM uses the file system to store the state of handled jobs. This allows it to recover safely from most system faults, after a restart.

The GM also includes user utilities to handle data transfer and registration in meta-data catalogues.

3.2 Information System

The NorduGrid information system is a distributed dynamic system which provides information on the status of the Grid resources. The information is generated on a resource upon request from a user or an agent when a status query is performed (pull model). The generated information can then be cached at different levels in the distributed database. The implementation consists of local information providers, local databases (first level cache), information indices with caching mechanism (higher level caches), a soft registration mechanism and an information model (schema) for representing the Grid resources. The dynamic soft registration mechanism makes it possible to construct different topologies of information indices and resources. The most typical Grid topology is a tree-like hierarchy. The User Interface with the built-in broker (Section 3.3) and the monitoring system (Section 3.5) are the main consumers of the dynamic information produced and stored in the information system.

The NorduGrid designed its own information model (schema) [11] for Globus MDS in order to properly represent and serve its environment. A working information system, as a part of the NorduGrid architecture, has been built and put into operation already in May 2002.

The information model The NorduGrid production environment (Section 2) consists of different resources located at the different sites. The NorduGrid information model [11] naturally maps these resources onto an LDAP-based [9] tree, where each resource is represented by a subtree. The LDAP-tree, or DIT, of a cluster is shown in Fig. 2.

The clusters are described by the *nordugrid-cluster* LDAP objectclass. The objectclass has several attributes to characterize the hardware, software and middleware properties of a cluster. The model supports both dedicated and non-dedicated Grid clusters and queues, as it contains attributes which make it possible to distinguish between the Grid-shared and non-Grid-shared resources.

A cluster provides access to Grid-enabled queues which are described by the *nordugrid-queue* objectclass. Under a queue entry, the *nordugrid-authuser* and *nordugrid-job* entries can be found grouped in two distinct subtrees (the branching is accomplished by the *nordugrid-info-group* objectclass). Every Grid user who is authorized to submit jobs to a queue should have an entry under the queue. Similarly, every Grid job submitted to the queue is represented by a job entry.

The *nordugrid-authuser* entry contains all the user-dependent information of an authorized Grid user. Within the user entries the Grid users can find out, among other things, how many CPUs are available for them in that specific queue, what is the available disk space their job can consume, *etc.* The NorduGrid philosophy is that the user-dependent information is crucial on the Grid, since the Grid users are more interested in their personal resources rather than the "general" resources of a cluster.

The *nordugrid-job* entries describe the Grid jobs submitted to a cluster. The detailed job information includes the job's unique Grid identifier, the certificate subject of the job's owner, the job status, the jobs submission machine, *etc.* The job monitoring is done solely by querying the information system. The job entries are kept on the execution cluster, thus implementing a distributed status monitoring system.

The schema contains the *nordugrid-se* and the *nordugrid-rc* objectclasses for describing Storage Elements and Replica Catalogues respectively, although these resources are not fully implemented yet.

The software The information system makes use of the Globus MDS [8]. The standard MDS package is an extensible toolkit for creating a Grid information system. It is built upon and heavily relies on the OpenLDAP software. Such an information system is implemented as a pseudo-distributed LDAP database, with the Grid information being stored in the attribute-value pairs of the entries of the LDAP trees.

The LDAP entries are generated "on the fly" by the information providers of a Grid resource upon a search request from a user or agent. NorduGrid has

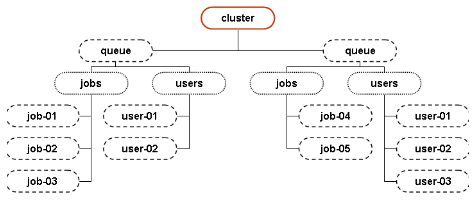


Fig. 2. The LDAP subtree corresponding to a cluster resource.

created its own set of information providers for populating the entries of the NorduGrid schema.

The local database of a resource is responsible for implementing the first-level caching of the output of the providers and presenting the formulated LDAP entries through the LDAP protocol. For the local database, the GRIS (Grid Resource Information Service) [12] LDAP back-end of the Globus MDS is used.

The local databases are linked together via the GIIS (Grid Information Index Services) information indices. These indices are implemented as a special GIIS LDAP back-end of the Globus MDS. Local resources register dynamically to the indices, which themselves can further register to other indices. Through the usage of the dynamic registration it is possible to create different topologies of indexed resources. The NorduGrid operate a multi-level tree hierarchy, based upon the geographical location of resources. In order to avoid any single points of failure, NorduGrid operate a multi-rooted tree with several top-level indices.

Besides the indexing function, the GIIS indices are capable of performing queries by following the registrations and caching the results of these search operations. However, the present implementation of the caching capability of the GIIS back-ends were found to be unreliable and unscalable in the everyday usage, therefore the NorduGrid operates with disabled GIIS caching. In such a layout, the GIISes are merely used as resource indices, while the query results are obtained directly from the resources (local databases).

3.3 User Interface and resource brokering

The user interacts with the NorduGrid through a set of command line tools. There are commands for submitting a job, for querying the status of jobs and clusters, for retrieving the data from finished jobs, for killing jobs *etc.* There are also tools that can be used for copying files to, from and between the Storage Elements and replica catalogues and for removing files from storage elements and replica catalogs. The full list of commands is given in Table 1. More detailed information about these commands can be found in the NorduGrid toolkit User Interface user's manual [13].

command	action
`ngsub`	Job submission
`ngstat`	Show status of jobs and clusters
`ngcat`	Display stdout or stderr of a running job
`ngget`	Retrieve the output from a finished job
`ngkill`	Kill a running job
`ngclean`	Delete a the output from the cluster
`ngsync`	Recreate the user interface's local information about running jobs
`ngmove`	Copy files to, from and between Storage Elements and Replica Catalogs
`ngremove`	Delete files from Storage Elements and Replica Catalogs

Table 1. The NorduGrid User Interface commands.

When submitting a Grid job using `ngsub`, the user should describe the job using extended RSL (xRSL) syntax (Section 3.4). This piece of xRSL should contain all the information needed to run the job (the name of the executable, the arguments to be used, *etc.*) It should also contain a set of requirements that a cluster must satisfy in order to be able to run the job. The cluster can e.g. be required to have a certain amount of free disk space available or have a particular set of software installed.

When a user submits a job using `ngsub`, the User Interface contacts the Information System (see Section 3.2): first to find available resources, and then to query each available cluster in order to do requirement matching. If the xRSL specification states that some input files should be downloaded from a Replica Catalog, the Replica Catalog is contacted as well, in order to obtain information about these files.

The User Interface then matches the requirements specified in the xRSL with the information obtained from the clusters in order to select a suitable queue at a suitable cluster. If a suitable cluster is found, the job is submitted to that cluster. Thus, the resource brokering functionality is an integral part of the User Interface, and does not require an additional service.

3.4 Resource specification language

The NorduGrid architectural solution requires certain extensions to the core Globus RSL 1.0 [14]. This concerns not only the introduction of new attributes, but also the differentiation between the two levels of the job options specifications:

- **User-side RSL** – the set of attributes specified by a user in a job description file. This file is interpreted by the User Interface (Section 3.3), and after the necessary modifications is parsed to the Grid Manager (GM, Section 3.1)
- **GM-side RSL** – the set of attributes pre-processed by the UI, and ready to be interpreted by the GM

This differentiation reflects the dual purpose of the RSL in the NorduGrid architecture: it is used not only by users to describe job requirements, but also by the UI and GM as a communication language. Such a functional separation, as well as re-defined and newly introduced attributes, prompted NorduGrid to refer to the used resource specification language as "xRSL" [15], in order to avoid possible confusion. xRSL uses the same syntax conventions as the core Globus RSL, although changes the meaning and interpretation of some attributes [15].

Most notable changes are those related to the file movement. The major challenge for NorduGrid applications is pre- and post-staging of considerable amount of files, often of a large size. To reflect this, two new attributes were introduced in xRSL: *inputFiles* and *outputFiles*. Each of them is a list of local-remote file name or URL pairs. Local to the submission node input files are uploaded to the execution node by the UI; the rest is handled by the GM. The output files are moved upon the job completion by the GM to a specified location

(Storage Element). If no output location is specified, the files are expected to be retrieved by a user via the UI.

Several other attributes were added in xRSL, for convenience of users. A typical xRSL file is shown below:

```
& (executable="ds2000.sh") (arguments="1101")
(stdout="dc1.002000.simul.01101.hlt.pythia_jet_17.log") (join="yes")
(inputfiles=("ds2000.sh"
"http://www.nordugrid.org/applications/dc1/2000/dc1.002000.simul.NG.sh"))
(outputFiles=
   ("dc1.002000.simul.01101.hlt.pythia_jet_17.log"
"rc://dc1.uio.no/2000/log/dc1.002000.simul.01101.hlt.pythia_jet_17.log")
   ("atlas.01101.zebra"
"rc://dc1.uio.no/2000/zebra/dc1.002000.simul.01101.hlt.pythia_jet_17.zebra")
   ("atlas.01101.his"
"rc://dc1.uio.no/2000/his/dc1.002000.simul.01101.hlt.pythia_jet_17.his")
   ("dc1.002000.simul.01101.hlt.pythia_jet_17.AMI"
"rc://dc1.uio.no/2000/ami/dc1.002000.simul.01101.hlt.pythia_jet_17.AMI")
   ("dc1.002000.simul.01101.hlt.pythia_jet_17.MAG"
"rc://dc1.uio.no/2000/mag/dc1.002000.simul.01101.hlt.pythia_jet_17.MAG")
) (jobname="dc1.002000.simul.01101.hlt.pythia_jet_17")
(runTimeEnvironment="DC1-ATLAS")
```

Such an extended RSL appears to be sufficient for job description of desired complexity. The ease of adding new attributes is particularly appealing, and NorduGrid is committed to use xRSL in further development.

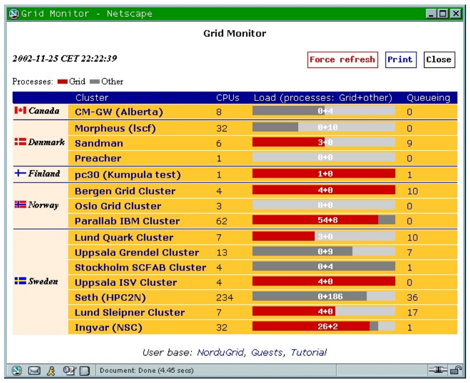
3.5 Monitoring

The NorduGrid provides an easy-to-use monitoring tool, realized as a Web interface to the NorduGrid Information System. This Grid Monitor allows browsing through all the published information about the system, providing thus a real-time monitoring and a primary debugging for the NorduGrid.

The Grid Monitor makes use of the LDAP module of PHP4 [16] to provide a Web interface to the information infrastructure. The structure of the Grid Monitor to great extent follows that of the NorduGrid Information System. For each objectclass, either an essential subset of attributes, or the whole list of them, is presented in an easily accessible inter-linked manner. This is realized as a set of windows, each being associated with a corresponding module.

The screen-shot of the main Grid Monitor window, as available from the NorduGrid Web page, is shown in Fig. 3. Most of the displayed objects are linked to appropriate modules, such that with a simple mouse click, a user can launch another module window, expanding the information about the corresponding object or attribute. Each such window gives access to other modules in turn, providing thus a rather intuitive browsing.

The Web server that provides the Grid Monitor access runs on a completely independent machine, therefore imposing no extra load on the NorduGrid, apart of very frequent LDAP queries (default refresh time for a single window is 30 seconds).



Grid Monitor

2002-11-25 CET 22:22:39

Processes: 🟥 Grid ▨ Other

	Cluster	CPUs	Load (processes: Grid+other)	Queueing
Canada	CM-GW (Alberta)	8	0+4	0
Denmark	Morpheus (lscf)	32	0+10	0
	Sandman	6	3+0	9
	Preacher	1	0+0	0
Finland	pc30 (Kumpula test)	1	1+0	1
Norway	Bergen Grid Cluster	4	4+0	10
	Oslo Grid Cluster	3	0+0	0
	Parallab IBM Cluster	62	54+8	0
Sweden	Lund Quark Cluster	7	3+0	10
	Uppsala Grendel Cluster	13	0+9	7
	Stockholm SCFAB Cluster	4	0+4	1
	Uppsala ISV Cluster	4	4+0	0
	Seth (HPC2N)	234	0+186	36
	Lund Sleipner Cluster	7	4+0	17
	Ingvar (NSC)	32	26+2	1

User base: NorduGrid, Guests, Tutorial

Fig. 3. The Grid Monitor.

4 Scientific applications

The NorduGrid project was initiated by the High Energy Physics community in the Nordic countries, therefore the majority of applications initially were related, although not limited, to this particular field.

The CPU-intensive tasks were the first to be submitted by the NorduGrid users, as the NorduGrid harnessed a lot of the CPU power in the Nordic countries, becoming a very attractive distributed resource. Several studies in the area of theoretical physics credited the NorduGrid for providing necessary resources for fast while complex analysis.

Data-intensive tasks are much more demanding for the Grid, and the NorduGrid architecture was designed to suit this kind of tasks, primarily for the purposes of the experimental High Energy Physics. In the framework of the ATLAS Experiment [17], the NorduGrid testbed was used to process large sets of data, producing yet larger output. For example, in the period from August to October 2002, 180 GB of input data was processed in 520 CPU-days and produced 760 GB of output. The input data were distributed over various NorduGrid sites, while the output was collected at a dedicated Storage Element.

5 Summary

The NorduGrid middleware was designed to be a light-weight, non-invasive and portable solution, suitable for any kind of available resources and requiring minimal intervention on the part of system administrators. This solution allows NorduGrid to embrace all kinds of available resources, providing authorized users with a widely distributed continuously operational production environment. The NorduGrid is a dynamic, heterogeneous Grid with fluctuating and inflating number of available resources of different kinds.

The NorduGrid philosophy proves to be a valid one, as the NorduGrid environment spans many sites around the World and makes use of several different operation systems and distributions, such as RedHat, Mandrake, Slackware or Debian Linux. The whole system runs reliably 24/7 without the need for being manned around the clock. The NorduGrid thus operates a production environment, being used by academic researchers in their daily work.

References

1. Nordic Testbed for Wide Area Computing And Data Handling. http://www.nordugrid.org.
2. The Globus Project. http://www.globus.org.
3. The European Union Datagrid Project. http://www.edg.org.
4. A. Waananen et al.. An overview of an architecture proposal for a High Energy Physics Grid. In J. Fagerholm, editor, *Proc. of PARA 2002, LNCS 2367, p. 76.* Springer-Verlag Berlin Heidelberg, 2002.
5. A. Konstantinov. The NorduGrid project: Using Globus Toolkit For Building Grid Infrastructure. In *Proc. of ACAT 2002, Nucl. Instr. and Methods* ***A***. Elsevier Science, 2002.
6. Gsiftp tools. http://www.globus.org/datagrid/deliverables/gsiftp-tools.html.
7. Globus Replica Catalog service. http://www-fp.globus.org/datagrid/replica-catalog.html.
8. The Monitoring And Discovery Service. http://www.globus.org/mds.
9. Open source implementation of the Lightweight Directory Access Protocol. http://www.openldap.org.
10. A. Konstantinov. The NorduGrid Grid Manager and GridFTP Server. Description And Administrator's Manual. http://www.nordugrid.org/documents/GM.pdf.
11. B. Kónya. The NorduGrid Information System. http://www.nordugrid.org/documents/ng-infosys.pdf.
12. Hierarchical GIIS, Globus documentation. http://www.globus.org/mds/hierarchical_GIIS.pdf.
13. M. Ellert. The NorduGrid Toolkit User Interface. http://www.nordugrid.org/documents/NorduGrid-UI.pdf.
14. The Globus Resource Specification Language RSL v1.0. http://www-fp.globus.org/gram/rsl_spec1.html.
15. O. Smirnova. Extended Resource Specification Language. http://www.nordugrid.org/documents/xrsl.pdf.
16. PHP: Hypertext Preprocessor. http://www.php.net.
17. The ATLAS Experiment. http://atlasexperiment.org.