The NorduGrid: Building a Production Grid in Scandinavia

P. Eerola*, T. Ekelöf†, M. Ellert†, J. R. Hansen‡, A. Konstantinov§, B. Kónya*, J. L. Nielsen‡, F. Ould-Saada§, O. Smirnova*, A. Wäänänen‡

*Particle Physics, Institute of Physics, Lund University, Box 118, 22100 Lund, Sweden

†Department of Radiation Sciences, Uppsala University, Box 535, 75121 Uppsala, Sweden

‡Niels Bohr Institutet for Astronomi, Fysik og Geofysik, Blegdamsvej 17, DK-2100, Copenhagen Ø, Denmark

§University of Oslo, Department of Physics, P. O. Box 1048, Blindern, 0316 Oslo, Norway

***Abstract*— The aim of the NorduGrid project is to build and operate a production Grid infrastructure in Scandinavia and Finland. The innovative middleware solutions allowed setting up a working testbed, which connects a dynamic set of computing resources by providing uniform access to them. The resources range from small test clusters at academic institutions to large farms at several supercomputer centers. This paper reviews the architecture and describes the Grid services, implemented via the NorduGrid middleware. As an example of a demanding application, a real use case of a High Energy Physics study is described.**

I. 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™ [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.

This paper gives an overview of the NorduGrid architecture, followed by a description of the Grid services provided by the NorduGrid middleware. A High Energy Physics use case is described as an example of a demanding application benefiting from the Grid technology. Finally, the future development plans are laid out.

II. Resources

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.

A. Hardware resources

The NorduGrid environment consists of five core sites, mostly dedicated to development and testing of the middleware, and several production clusters, scattered across the Nordic countries. The core sites run small test clusters (a few CPUs) and comprised the initial testbed, put up in May 2002. Since this testbed was shown to be stable, reliable and capable of executing production-level tasks, more sites have joined with large production clusters.

B. Users

The user base of the NorduGrid consists predominantly of High Energy Physics researchers, who use the Grid in their work on a daily basis. By end-2002 there were around two dozens of active users in two Virtual Organizations (Section II-C). The NorduGrid operate their own Certification Authority, recognized by other related projects, such as EDG.

C. Authorization in NorduGrid

In a Grid environment such as the NorduGrid, users usually don't have local password-protected user accounts on the computing resources they want to use. They hold instead a certificate issued by a Certification Authority. The access control for the computing resources (authorization) is a matter of a local policy: site administrators retain the full control of choosing which Grid user is allowed to use the resources. The local authorization process is done by mapping the accepted set of Grid users onto the local user accounts.

NorduGrid have set up a collective authorization method, the NorduGrid Virtual Organization (VO), following practices of other Grid testbeds. This VO maintains a list of people which are authorized to use the NorduGrid resources. The VO tools provide an automatic method for the sites to easily maintain the "VO member"↔"local Unix user" mappings. The tools periodically query the VO user database and automatically generate the local grid-mapfiles following the local policy formulated in the VO-tools configuration file. The automatic mapping does not violate the site autonomy, because the site administrators retain a full control over their systems thanks to the possibility of denying access to "unwished" Grid users in the NorduGrid VO-tools configuration file.

The database of the VO is maintained by the VO managers. Their responsibility is to add, delete or modify user entries. The NorduGrid VO supports the creation of groups – they can be created or removed only by the VO managers. With the existence of the user groups, the site administrators can implement group based mappings, such that all the members of a certain group are mapped to the same local Unix user, in addition to the default user-based mappings.

Technically, the VO database is stored in an LDAP [6] database. For this purpose, a GSI [7] enabled OpenLDAP server is used. The built-in SASL (Cyrus Simple Authentication and Security Layer [8]) and GSI-GSSAPI mechanisms of the OpenLDAP server provide an entry and attribute level access control, based on the Grid certificates. The NorduGrid sites periodically run the *nordugridmap* utility in order to query the VO LDAP server and automatically create/update local user mappings according to a site policy.

III. 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:

- 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. Fig. 1 depicts the basic components of the NorduGrid architecture and schematic relations between them. In what follows, a detailed description of the components is given.

A. Computing Cluster

A *cluster* is the natural computing unit in the NorduGrid architecture. It consists of a front-end (”master”) node which manages several back-end (”worker”) nodes, normally via a private closed network. The software component currently consists of a standard batch system setup plus extra components to interface with the Grid. The operation system of choice is Linux in its many flavors, however, other Unix-like systems (*e.g.* HP-UX, Solaris) can be used as well.

The configuration of the batch system is very liberal, respecting local setup and configuration policies. Much effort has been put into ensuring that the NorduGrid tools should not change current installations or impose any restrictions, but rather try to be an ”add-on” component which hooks the local resources onto the Grid and allows for Grid jobs to run along with the conventional jobs.

1) Front-end, the gatekeeper: The responsibility of the front-end machine is to handle job requests from the Grid and translate them into requests to the local resource management system (LRMS). This task is shared such that the authentication and authorization of requests is handled by the gatekeeper which speaks the GridFTP protocol [9], while the Grid Manager (Section IV-A) takes authorized requests and does the job management.

The gatekeeper (or the GridFTP server as it is normally called, since it is its essence) is expandable through plugins. It uses the job submission plugin to submit jobs and acts as a normal GridFTP file server. More details can be found in Sections III-B andIV-A.

Once a request from the Grid has been accepted, the control is passed to the Grid Manager. The Grid Manager (Section IV-A) from now on manages the job, from actually retrieving input files for the job and submitting the job to the LRMS, to cleaning up after the job execution and uploading output files to the user-specified locations.

The LRMS which is normally used in NorduGrid is either OpenPBS or a PBSPro [10], eventually with *e.g.* the Maui [11] scheduler. However, there are no specific design limitations as to which batch system to use.

2) Worker nodes: The *worker node* (WN) is the back-end of the system, where the execution of an application is performed. WNs are managed entirely through the LRMS, and no NorduGrid middleware has to be installed on them.

An important part of the NorduGrid architecture is that the WNs are not required to have a direct access to the Internet. Thus applications can not expect to be able to retrieve files via *e.g.* WWW or from global file systems like AFS. This concept is a part of the philosophy that as few restrictions as possible should be imposed on a site.

The only requirement for the cluster setup is that there should be a shared filesystem (*e.g.* NFS) between the front-end and the back-end nodes. This so-called *session directory* is then available to the job during the execution and to the Grid Manager for input and output file staging.

B. Storage Element

A concept of a Storage Element (SE) is not fully developed by the NorduGrid at this stage. So far, SEs are implemented as plain GridFTP servers. A software used for this is either the GridFTP server provided as a part of the Globus Toolkit$^{\mathrm{TM}}$, or the one delivered as a part of the NorduGrid Grid Manager (see Section IV-A). The latter is preferred at the moment, since it allows configuring the access based on the user Grid identity (Distinguished Name of a certificate) instead of the local identities to which users are mapped. At present, NorduGrid operates two high-capacity Storage Elements situated at Oslo, Norway, and several test set-ups.

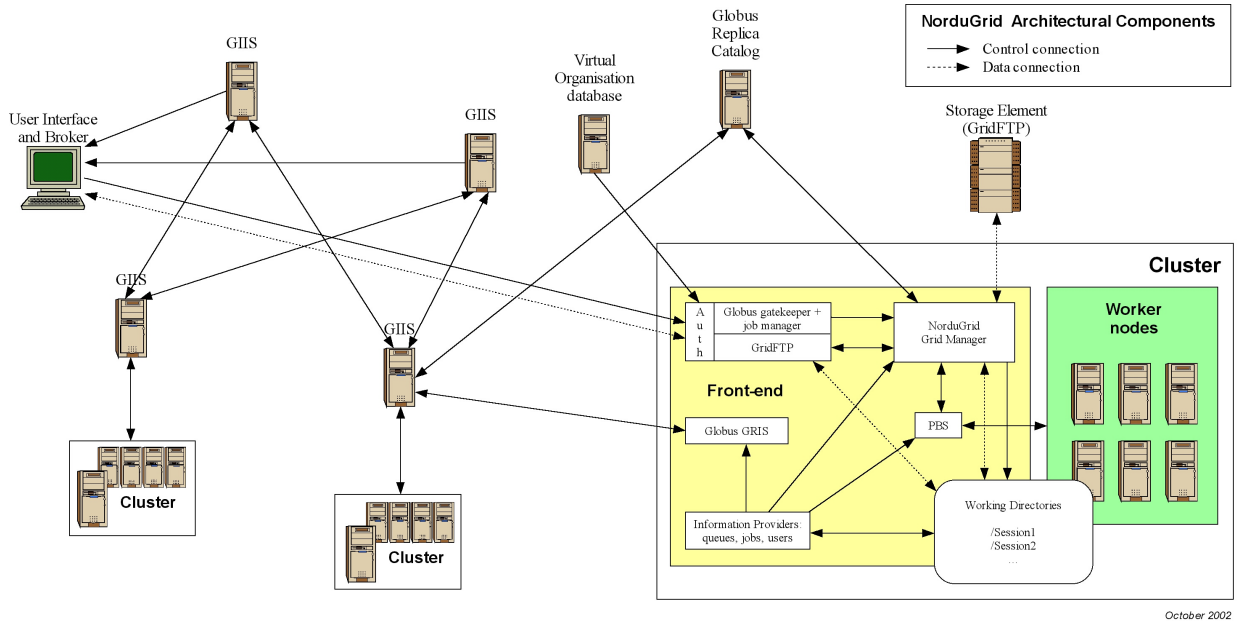


Fig. 1
COMPONENTS OF THE NORDUGRID ARCHITECTURE.

C. Replica Catalog

The NorduGrid project uses the Globus Replica Catalog (RC) developed by the Globus project [2] for locating and registering data sources.

The RC implementation in Nordugrid follows closely the one from Globus. The RC server is based on an OpenLDAP [6] server with an LDBM back-end. There are no significant changes introduced into the original objects schema. Since the applications are not expected to perform heavy concurrent writing to the RC, some advanced features, like rollbacks, are not used.

Unfortunately, the OpenLDAP has appeared to be unstable under heavy load during production runs, and is overall somewhat problematic. One of the problems appeared as the size of the catalogue grew up. It appeared that the OpenLDAP has problems with transferring relatively big amounts of data over an authenticated/encrypted connection. Some problems were fixed by applying appropriate patches, and some by automatic restart of the server. Together with a fault tolerant behavior of the client part, these fixes made the system usable.

To manage the information in the RC server, the Globus Toolkit^{TM} API and libraries were used. The only significant change that was made, was to add the possibility to perform securely authenticated connections based on the Globus GSSAPI-GSI plugin for SASL.

IV. THE NORDUGRID MIDDLEWARE

A. Grid Manager

The NorduGrid Grid Manager (GM) [12] 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. The following parts of Globus are used:

- **GridFTP**: fast and reliable data access for Grid with integrated security,
- **GASS Transfer**: support for HTTP(S)-like data access protocol,
- **Replica Catalog**: meta-data storage [13],
- **Replica Location Service**, designed by a collaboration between Globus and EDG [14],
- **RSL**: extendable Resource Specification Language [15].

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 (often called

Storage Elements).

The GM accepts job-submission scripts described in Globus RSL (Section IV-D). Since RSL only defines a syntax and does not put any restrictions on the used attributes, some additional attributes were introduced to reflect the features of the GM.

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 therefore provides the following functionality:

- Unified interface for job submission and data staging from a user machine and retrieval of results (GridFTP is used).
- Support for GridFTP, FTP, HTTP or HTTPS-like servers for data staging with a possibility to lookup/register information in meta-data catalogues.
- Fault tolerant transfers.
- Caching of downloaded data.
- Email notification of job status. The job execution log is sent to a user in case a problem was encountered, and can also be provided to a user upon request.
- Support for software packages runtime environment configuration.
- Handling of session directories (creation, cleaning, staging to/from a computing node).

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 PBS 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. These are *ngmove* for transferring/registering data and *ngremove* for removing/unregistering.

B. 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 IV-C) and the monitoring system (Section IV-E) rely totaly on the Information System.

NorduGrid have chosen the hierarchical LDAP-based information system, and designed its own information model (schema) 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. While some other information models have been proposed recently (*e.g.* CIM [16], Glue [17]), the NorduGrid schema is up to now the only one which has been extensively tested and used in an actual production.

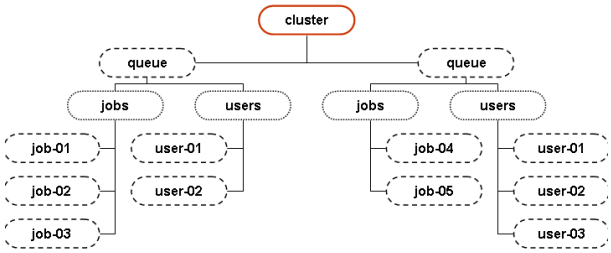


Fig. 2
THE LDAP SUBTREE CORRESPONDING TO A CLUSTER RESOURCE.

1) The information model: The NorduGrid production environment (Section III) consists of different resources located at the different sites. The list of implemented Grid services involves the computing resources (Section III-A), Storage Elements (Section III-B) and Replica Catalogs (Section III-C). The NorduGrid information model [18] naturally maps these resources onto an LDAP-based 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. This 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-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 Catalogs respectively.

2) The software: The information system makes use of the MDS (Monitoring and Discovery Services) of Globus [19]. 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. The information providers are small efficient programs which interface to and gather information from the Grid resources, like clusters, Storage Elements, *etc*. NorduGrid has 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) [20] 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).

C. 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 (see Section III-B) and replica catalogues (see Section III-C) and for removing files from storage elements and replica catalogs. The full list of commands is given in Table I. More detailed information about these commands can be found in the NorduGrid toolkit User Interface user's manual [21].

When submitting a Grid job using `ngsub`, the user should describe the job using extended RSL (xRSL) syntax (Section IV-D). 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 IV-B): 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.

D. Resource specification language

The NorduGrid architectural solution was designed primarily to suit specific High Energy Physics tasks. While using extensively the Globus Toolkit^{TM}, this solution requires certain extensions to the core Globus RSL 1.0. 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 IV-C), and after the necessary modifications is parsed to the Grid Manager (GM, Section IV-A)
- **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" [22], in order to avoid possible confusion. xRSL uses the same syntax conventions as the core Globus RSL,

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 I
THE NORDUGRID USER INTERFACE COMMANDS.

although changes the meaning and interpretation of some attributes.

Most notable changes are those related to the file movement. The implementation of the Grid Manager implies that while most of the GRAM RSL attributes can be re-used, they have be interpreted differently. For example, the standard *executable* attribute becomes obsolete, although it is preserved in the xRSL to preserve compatibility with Globus. By using specific syntax conventions, the main executable will be either uploaded by the UI from the submission node, or will be located by the GM on the execution node.

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. If no output location is specified, the files are expected to be retrieved by a user via the UI. Any of the input files can be given an executable permission by listing it in the *executables* attribute. Output file location can be specified as a Replica Catalog pseudo-URL, such that the file will be moved to a location specified in a logical collection, and registered in the RC.

It should be mentioned that the most recent implementation of the Globus GRAM adds some new attributes to the RSL, similar to those introduced in xRSL. However, they were not available at the time of the GM and xRSL development.

Several other attributes were added in xRSL, for convenience of users. Using the *cluster* attribute, a user can explicitly select or exclude a list of clusters on which the resource matching will be performed. The complete GM job log can be retrieved by specifying the *stdlog* attribute. Standard output and standard error can be merged into one file by requesting it in the *join* attribute. In order to enable matching based on a specific cluster configuration and setup, such attributes can be used as *middleware*, *architecture*, *runTimeEnvironment etc*. Since most file movement is done via GridFTP, a user can request a specific number of parallel FTP threads by using the *ftpThreads* attribute.

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.

E. 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 NorduGrid Information System (Section IV-B) provides a robust and dynamic model for accessing not only quasi-static information about resources and services, but also about such rapidly changing parameters such as queue and job status. Being based on OpenLDAP, it can easily be interfaced to any browsing or monitoring tool, giving thus a user-friendly overview of all the resources.

The Grid Monitor makes use of the LDAP module of PHP4 [23] 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. The basic objects are defined by the NorduGrid schema objectclasses, such as *nordugrid-cluster*, *nordugrid-queue*, *nordugrid-job* and *nordugrid-authuser*. The Grid Monitor also uses the NorduGrid Virtual Organisation (VO) objectclass *organisationalPerson* and the corresponding attributes.

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. There are seven major modules in the Grid Monitor:

1. An overall Grid Monitor
2. Cluster Description
3. Queue Details
4. Job Information
5. User Information
6. Attributes Overview
7. List of Users

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

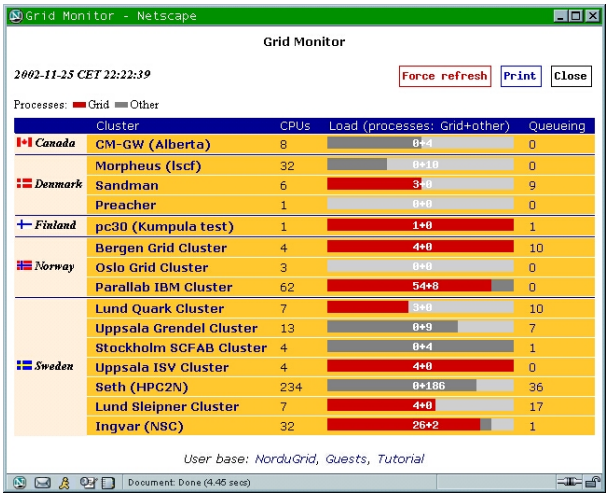


Fig. 3
THE GRID MONITOR.

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 Grid Monitor proves to be a very useful and reliable tool, being the most frequently requested service of the NorduGrid, both by users and developers. 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).

F. Software configuration

Since the Grid is supposed to be deployed on many sites, it implies involvement of many site administrators, of which not all may be Grid experts. Therefore the configuration of the NorduGrid Toolkit was made as simple as possible. It basically requires writing two configuration files: `globus.conf` and `nordugrid.conf`. The `globus.conf` is used by the *globus-config* package, which configures the information system of the Globus Toolkit™ from a single file. This configuration scheme is not specific to NorduGrid and is also used by other projects, such as the EDG. The `nordugrid.conf` file is used for configuring the various NorduGrid Toolkit components.

This approach proved to be very convenient and allowed to set up sites as remote from Scandinavia as Canada or Japan in a matter of hours, with little help from the NorduGrid developers.

G. Distribution

The NorduGrid Toolkit is freely available via the Web site [1]. One can find there source tar-balls as well as CVS snapshots and nightly builds. There are binary RPM distributions for several Linux distributions. Furthermore, there is a stand-alone local client installation for users without root

```
& (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")
```

Fig. 4
A TYPICAL JOB DESCRIPTION.

privileges. It is distributed as a tar-ball and is designed to be a NorduGrid entry point, working out-of-the-box. Since it contains the necessary Globus components, it can be used to interact with other Grids as well.

For completeness, NorduGrid software repository contains some required third-party software, such as the Globus Toolkit™.

V. USE CASE: ATLAS DATA CHALLENGE

The ATLAS Experiment [24] at the Large Hadron Collider is a huge High Energy Physics detector, at the moment under construction at the CERN physics laboratory in Geneva, Switzerland. Its goal is to explore the fundamental nature of the matter and forces that shape our Universe.

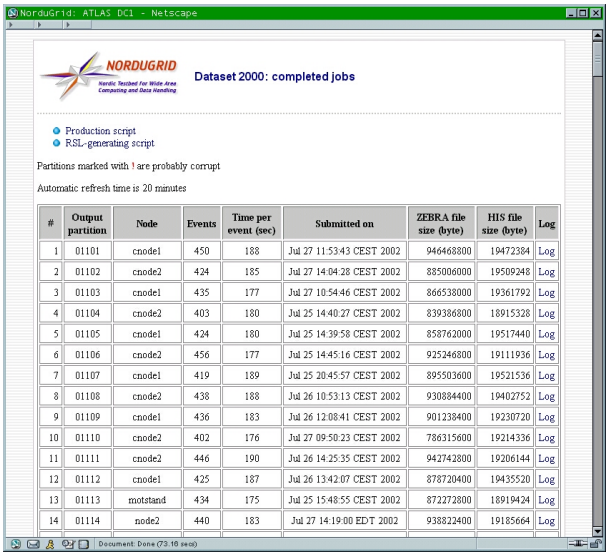
Vast quantities of data from the experiment are to be collected and interpreted by computers and stored on disks and tapes. The amount of data to be stored is expected to be around 3 PB per year. The computing challenge faced by ATLAS is thus enormous, so to be ready in 2007, when the experiment is planned to start operating, ATLAS has planned a series of computing challenges of increasing size and complexity to test its computing model and the complete software suite.

One of the stages of these challenges, the so-called ATLAS Data Challenge 1, ran from July to September 2002 and consisted of large-scale physics simulations based on requests from different physics communities. 39 different institutes from around the world participated in the simulations, and NorduGrid were amongst them.

Participants were assigned different data sets, according to their capacities. The NorduGrid was assigned two different data sets: 10 percent of the so-called *Dataset 2000* and the whole of the *Dataset 2003*.

For the Dataset 2000, the input data volume was 18 GB, stored in 15 files, each of which was made available locally. A typical xRSL job submission script is shown in Fig.4.

The notation is almost self-explanatory. The executable, `ds2000.sh`, is a script downloaded from the specified URL, which takes as an argument the processed partition number, which in this case equals to `1101`. `ds2000.sh` calls the ATLAS physics simulation program. To make sure that the ATLAS software is installed on the chosen cluster, the xRSL specifies that the simulation needs the runtime environment `DC1-ATLAS`. A small script, also called `DC1-ATLAS`, is



#	Output partition	Node	Events	Time per event (sec)	Submitted on	ZEBRA file size (byte)	HIS file size (byte)	Log
1	01101	cnode1	450	188	Jul 27 11:53:43 CEST 2002	946468800	19472384	Log
2	01102	cnode2	424	185	Jul 27 14:04:28 CEST 2002	885006000	19509248	Log
3	01103	cnode1	435	177	Jul 27 10:54:46 CEST 2002	866538000	19361792	Log
4	01104	cnode2	403	180	Jul 25 14:40:27 CEST 2002	839386800	18915328	Log
5	01105	cnode1	424	180	Jul 25 14:39:58 CEST 2002	858762000	19517440	Log
6	01106	cnode2	456	177	Jul 25 14:45:16 CEST 2002	925246800	19111936	Log
7	01107	cnode1	419	189	Jul 25 20:45:57 CEST 2002	895503600	19521536	Log
8	01108	cnode2	438	188	Jul 26 10:53:13 CEST 2002	930884400	19402752	Log
9	01109	cnode1	436	183	Jul 26 12:08:41 CEST 2002	901238400	19230720	Log
10	01110	cnode2	402	176	Jul 27 09:50:23 CEST 2002	786315600	19214336	Log
11	01111	cnode2	446	190	Jul 26 14:25:35 CEST 2002	942742800	19206144	Log
12	01112	cnode1	425	187	Jul 26 13:42:07 CEST 2002	878720400	19435520	Log
13	01113	motstand	434	175	Jul 25 15:48:55 CEST 2002	872272800	18919424	Log
14	01114	node2	440	183	Jul 27 14:19:00 EDT 2002	938822400	19185664	Log

Fig. 5

THE WEB PAGE SHOWING THE OUTPUT FILES OF THE DATASET 2000.

then run locally at the chosen cluster, setting up the ATLAS software environment, together with a variable pointing to the local location of the input files. After simulation, all the output files are uploaded to the Storage Element `dc1.uio.no` and registered into the Replica Catalog.

In total there were 300 jobs. They used around 220 CPU-days for the simulations and produced 1500 output files with a total size of about 320 GB. A Web page querying the Replica Catalog every 20 minutes for output files was set up to allow an easy browsing of the log-files of the simulations and an easy check if something had gone wrong during the physics simulation (see Fig. 5).

Dataset 2003 was more challenging, because the input volume was a staggering 158 GB in 100 input files. Some sites in NorduGrid were not able to accommodate such a big data set, and it was therefore decided to distribute only subsets to these sites. Some sites did not have any files available locally and used features of the NorduGrid software to download and cache input data. All the distributed sets were then registered into the Replica Catalog.

For this data set, there were 1000 jobs using about 300 CPU-days and producing a total output of 442 GB. All output files were successfully uploaded to the `dc1.uio.no` Storage Element and registered into the Replica Catalog.

VI. CONCLUSION

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.

ACKNOWLEDGEMENTS

The NorduGrid project is funded by the Nordic Council of Ministers through the Nordunet2 programme and by NOS-N. The authors would like to express their gratitude to system administrators across the Nordic countries for their courage, patience and assistance in enabling the NorduGrid environment. In particular, our thanks go to Ulf Mjörnmark and Björn Lundberg of Lund University, Björn Nilsson of NBI Copenhagen, Niclas Andersson and Leif Nixon of NSC Linköping and Åke Sandgren of HPC2N Umeå.

REFERENCES

[1] *Nordic Testbed for Wide Area Computing And Data Handling*. [Online]. Available: http://www.nordugrid.org
[2] *The Globus Project*. [Online]. Available: http://www.globus.org
[3] *The European Union DataGrid Project*. [Online]. Available: http://www.edg.org
[4] A. Waananen, "An overview of an architecture proposal for a high energy physics grid," in *Proc. of PARA 2002, LNCS 2367, p. 76*, J. Fagerholm, Ed. Springer-Verlag Berlin Heidelberg, 2002.
[5] A. Konstantinov, "The nordugrid project: Using globus toolkit for building grid infrastructure," to appear in Proc. of ACAT 2002, Nucl. Instr. and Methods **A**. Elsevier Science, 2002.
[6] *Open source implementation of the Lightweight Directory Access Protocol*. [Online]. Available: http://www.openldap.org
[7] *Grid Security Infrastructure (GSI)*. [Online]. Available: http://www.globus.org/security/
[8] *Simple Authentication and Security Layer*. [Online]. Available: http://asg.web.cmu.edu/sasl
[9] *GSIFTP Tools*. [Online]. Available: http://www.globus.org/datagrid/deliverables/gsiftp-tools.html
[10] *The Portable Batch System*. [Online]. Available: http://www.openpbs.org
[11] *The Maui Scheduler*. [Online]. Available: http://supercluster.org/maui
[12] A. Konstantinov, *The NorduGrid Grid Manager and GridFTP Server. Description and Administrator's Manual*. [Online]. Available: http://www.nordugrid.org/documents/GM.pdf
[13] *Globus Replica Catalog Service*. [Online]. Available: http://www-fp.globus.org/datagrid/replica-catalog.html
[14] *Replica Location Service*. [Online]. Available: http://grid-data-management.web.cern.ch/grid-data-management/replica-location-service/index.html
[15] *The Globus Resource Specification Language RSL v1.0*. [Online]. Available: http://www-fp.globus.org/gram/rsl%5Fspec1.html
[16] *The CIM-based Grid Schema Working Group*. [Online]. Available: http://www.isi.edu/~flon/cgs-wg/index.htm
[17] *The GLUE schema effort*. [Online]. Available: http://www.hicb.org/glue/glue-schema/schema.htm,http://www.cnaf.infn.it/~sergio/datatag/glue/index.htm
[18] B. Kónya, *The NorduGrid Information System*. [Online]. Available: http://www.nordugrid.org/documents/ng-infosys.pdf
[19] *The Monitoring and Discovery Service*. [Online]. Available: http://www.globus.org/mds
[20] *Hierarchical GIIS, Globus documentation*. [Online]. Available: http://www.globus.org/mds/hierarchical%5FGIIS.pdf
[21] M. Ellert, *The NorduGrid toolkit user interface*. [Online]. Available: http://www.nordugrid.org/documents/NorduGrid-UI.pdf
[22] O. Smirnova, *Extended Resource Specification Language*. [Online]. Available: http://www.nordugrid.org/documents/xrsl.pdf
[23] *PHP: Hypertext Preprocessor*. [Online]. Available: http://www.php.net
[24] *The ATLAS Experiment*. [Online]. Available: http://atlasexperiment.org