Building a production grid in Scandinavia

Eerola, Paula; Konya, Balazs; Smirnova, Oxana; Ekelof, T; Ellert, M; Hansen, JR; Nielsen, JL; Waananen, A; Konstantinov, A; Ould-Saada, F

Published in:
IEEE Internet Computing

DOI:
10.1109/MIC.2003.1215657

2003

Building a Production Grid in Scandinavia

Innovative middleware solutions are key to the NorduGrid testbed, which spans academic institutes and supercomputing centers throughout Scandinavia and Finland and provides continuous grid services to its users.

Paula Eerola, Balázs Kónya, and Oxana Smirnova
Lund University, Sweden

Tord Ekelöf and Mattias Ellert
Uppsala University, Sweden

John Renner Hansen, Jakob Langgaard Nielsen, and Anders Wäänänen
Niels Bohr Institute, Copenhagen

Aleksandr Konstantinov and Farid Ould-Saada
University of Oslo

Academic researchers in the Nordic countries participate in many common projects that process large amounts of data. To function effectively, such collaborations need a grid computing infrastructure that works across a wide area and uses distributed resources efficiently. Initial evaluations of existing grid solutions, however, showed that they failed to meet this requirement. The Globus Toolkit (www.globus.org), for example, had no resource brokering capability, and the European Union's DataGrid project (EDG; www.edg.org) did not satisfy stability and reliability demands. Other grid projects had even less functionality.

In May 2001, researchers at Scandinavian and Finnish academic institutes launched the NorduGrid project (www.nordugrid.org), with the goal of building a Nordic testbed for wide-area computing and data handling. We first developed a proposal for an original architecture and implementation.[1,2] We then developed middleware based on the Globus libraries and API, adding a set of completely new services such as resource brokering and monitoring. The NorduGrid middleware thus preserves Globus compatibility and permits interoperability with other Globus-based solutions. It also meets our goal of providing a lightweight, yet robust solution that is noninvasive, portable, and requires minimal intervention from system administrators.

We launched our testbed in May 2002, and it has been continuously operating since August of that year, providing reliable, round-the-clock services for academic users. NorduGrid spans several countries and incorporates several national computing centers, making it one of the largest operational grids in the world.

NorduGrid Overview

We built the NorduGrid testbed using the resources of national supercomputer centers and academic institutes, and based it on the Nordic countries' academic networks. Currently, the testbed uses more than 900 CPUs in 20 clusters that range in size from 4 to 400 processors. Except for a few test clusters, the resources are not grid-dedicated. Among the active NorduGrid users are high-energy physics researchers who use the Grid daily and physics theorists who use it to perform complex computational tasks.[3,4]

Guiding Philosophy

We planned and designed the NorduGrid architecture to satisfy the needs of both users and system administrators. These needs constitute a general philosophy:

- Start with something simple that works.
- Avoid single points of failure.
- Give resource owners full control over their resources.
- Ensure that NorduGrid software can use the existing system and, eventually, other preinstalled grid middleware, such as different Globus versions.
- Leave installation details (method, operating system, configuration, and so on) up to system administrators.
- Pose as few restrictions on site configuration as possible — for example, permit computing nodes on private as well as public networks, let clusters select the amount of resources they'll dedicate to the Grid, and install NorduGrid software only on front-end machines.

We thus designed the NorduGrid tools to handle job submission and management, user area management, and minimal data management and monitoring capacity.

System Components

The NorduGrid architecture's basic components are the user interface, information system, computing cluster, storage element, and replica catalog.

The NorduGrid's *user interface* is a new service that includes high-level functionality not completely covered by the Globus Toolkit — namely, resource discovery and brokering, grid job submission, and job status querying. NorduGrid thus does not require a centralized resource broker. The user interface communicates with the NorduGrid grid manager and queries the information system and replica catalog. Users can install the user interface client package on any machine, using as many interfaces as they need.

The *information system* is a distributed service that serves information for other components, such as monitors and user interfaces. The information system consists of a dynamic set of distributed databases that are coupled to computing and storage resources to provide information on a specific resource's status. The information system operates on a pull model: when queried, it generates requested information on the resource locally (optionally caching it afterward). Local databases register to a global set of indexing services via a soft-state registration mechanism. For direct queries, the user interface or monitoring agents contact the indexing registries to find contact information for local databases.

The *computing cluster* consists of a front-end node that manages several back-end nodes, typically through a private closed network. The software component is a standard batch system, with an extra layer that acts as a grid interface and includes the grid manager, the GridFTP server,[5] and the local information service. Although Linux is the operating system of choice, Unix-like systems, including Hewlett-Packard's UX and Tru64 Unix, can be used as well. The NorduGrid does not dictate batch system configuration; its goal is to be an add-on component that hooks local resources onto the Grid and lets grid jobs run along with conventional jobs, respecting local setup and configuration policies. The cluster has no specific requirements beyond a shared file system (such as Network Filesystem) between the front- and back-end nodes. The back-end nodes are managed entirely through the local batch system; no grid middleware is required on them.

We've yet to fully develop NorduGrid's *storage element*; so far we've implemented storage as plain GridFTP servers, which come as either part of the Globus or the NorduGrid Toolkit. We prefer the latter because it allows access control based on users' grid certificates rather than their local identities.

To register and locate data sources, we modified the Globus project's *replica catalog* to improve its functionality. The catalog's records are primarily entered and used by the grid manager and the user interface. The user interface can also use the records for resource brokering.

How It Works

Users access NorduGrid resources and related projects, such as EDG, using certificates issued by the NorduGrid certification authority. Like other grid testbeds, NorduGrid uses a central service that maintains a list of authorized users. The list is stored in an open-source implementation of the lightweight directory access protocol database (Open-LDAP; www.openldap.org), which uses the grid security infrastructure[6] mechanism for secure authentication. The Open-LDAP server's built-in security mechanisms control access at the entry and attribute levels, based on the grid certificates. We also developed the NorduGrid mapping utility to periodically query the LDAP server and automatically create and update local user mappings according to site policy.

The NorduGrid task flow includes four basic steps. First, the user prepares a job description using the extended Globus Resource Specification Language (RSL). This description might include application-specific requirements, such as input and output data descriptions, as well as other options used in resource matching, such as the architecture or an explicit cluster. These options are needed only if the user has specific preferences and wants to direct a job to a known subset of resources. The user can also request email notifications about the job status.

Next, the user interface interprets the job description and brokers resources using the information system and replica catalog. It then forwards the job to the grid manager on the chosen cluster and eventually uploads the specified files. The grid manager handles preprocessing, job submission to the local system, and post-processing, on the basis of the job specifications. It manipulates input and output data with the help of the replica catalog and storage elements.

Finally, users can receive email notifications or simply follow the job's progress through the user interface or monitor. Once the job is complete, users can retrieve the files specified in the job description. If the files are not fetched within 24 hours, the local grid manager erases them.

NorduGrid Middleware

We based the NorduGrid middleware almost entirely on the Globus Toolkit's API, libraries, and services. To support the NorduGrid architecture, however, we used several innovations such as the grid manager and the user interface, and we extended other components, including the information model and RSL.

Grid Manager

The grid manager software runs on the cluster's master node and acts as a smart front end for job submission to a cluster, job management, data pre- and post-staging functionality, and metadata catalog support.

We wrote the grid manager as a layer above the Globus Toolkit libraries and services. At that time, the existing Globus services didn't meet the NorduGrid architecture's requirements, including integrated replica catalog support, sharing cached files among users, and staging input and output data files. Because the system performs data operations at an additional layer, it handles data only at a job's beginning and end. Hence, we expect the user to provide complete information about the input and output data. This is our approach's most significant limitation.

Unlike the Globus resource allocation manager,[7] the grid manager uses a GridFTP interface for job submission. To allow this, we developed a NorduGrid GridFTP server based on Globus libraries.[5] The main features that distinguish our server from the Globus implementation are:

- a virtual directory tree, configured for each user;
- access control, based on the distinguished name stored in the user certificate;
- a local file access plug-in, which implements ordinary FTP-server-like access; and
- a job submission plug-in, which provides an interface for submission and control of jobs that the grid manager handles.

NorduGrid also uses the GridFTP server to create relatively easy-to-configure GridFTP-based storage elements.

The grid manager accepts job-submission scripts written in Globus RSL with few new attributes added. For each job, the grid manager creates a separate session directory to store the input files. Because a cluster front end directly gathers the input data, there is no single point (machine) through which all the job data must pass. The grid manager then creates a job-execution script and launches it using a local resource management system. Such a script can perform other actions as well, including setting the environment for third-party software packages that a job requests.

After a job has finished, the grid manager transfers all specified output files to their destinations or temporarily stores them in the session directory so that users can retrieve them later. The grid manager can also cache input files that jobs and users can share; if the protocol allows it, the manager checks for authorization of user access requests against a remote server. To save disk space, the grid manager can provide cached files to jobs as soft links.

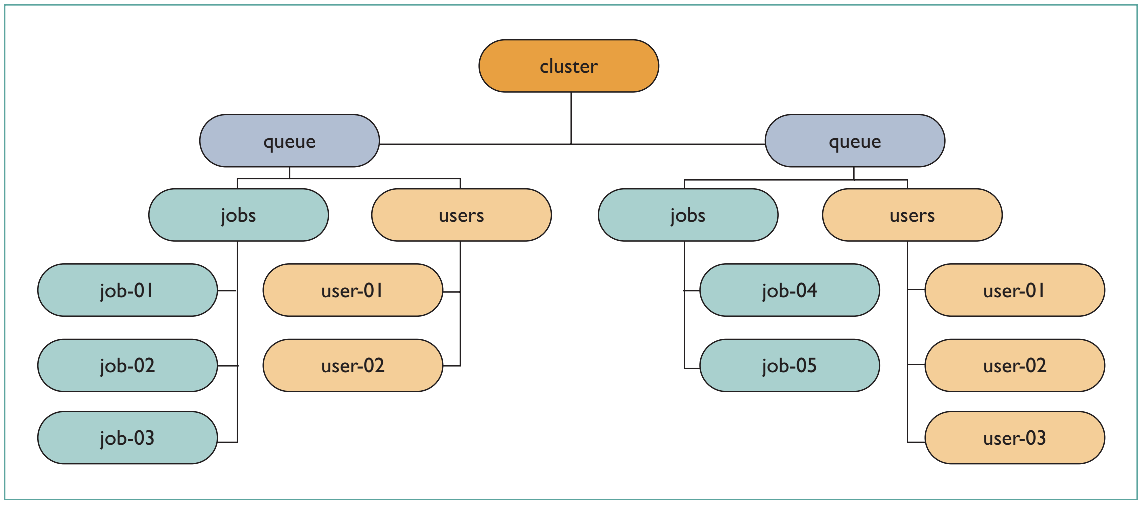


Figure 1. The LDAP subtree corresponding to a cluster resource. A queue entry represents grid-enabled queues and branches job entries, and authorized user entries, which include information such as each user's available resources.

As in Globus, the grid manager implements a bash-script interface to a local batch system, which enables easy interoperation with most local resource management systems. To store the state of all jobs, the grid manager uses a file system, which lets it recover safely from most system faults after a restart. The grid manager also includes user utilities for data transfer and metadata catalog registration.

Replica Catalog

NorduGrid uses the Globus replica catalog, which is based on an Open-LDAP server with the default LDAP database manager back end. We fixed Open-LDAP's problems with transferring large amounts of data over an authenticated or encrypted connection by applying appropriate patches and automating server restart. These modifications, combined with fault-tolerant client behavior, made the Globus system usable for our needs.

We used the Globus Toolkit API and libraries to manage the replica catalog's server information. Our only significant change here was to add the Globus grid security infrastructure mechanism for securely authenticating connections.

Information System

We created a dynamic, distributed information system[8] by extending the Globus Toolkit's monitoring and discovery services.[9] MDS is an extensible framework for creating grid information systems built on Open-LDAP software. An MDS-based information system consists of an information model (schema), local information providers, local databases, a soft registration mechanism, and information indices. Adding NorduGrid extensions and a specific setup gave us a reliable information system backbone.

An effective grid information model results from a delicate design process that shows how to best represent resources and structure resource information. In an MDS-based system, we store information as attribute-value pairs of LDAP entries, organized into a hierarchical tree (see Figure 1). The information model is thus technically formulated through an LDAP schema and LDAP-tree specification.

Our evaluation of the original MDS and EDG schemas[9,10] showed their unsuitability for simultaneously describing clusters, grid jobs, and grid users. We thus designed our own information model.[8] Unlike Globus and other grid projects that keep developing new schemas (see for example, the Glue schema at www.hicb.org/glue/glue-schema/schema.htm and the CIM-based Grid Schema Working Group at www.isi.edu/~flon/cgs-wg/) we've deployed, tested, and used our model in a production facility.

NorduGrid's information model describes the main grid components: computing clusters, grid users, and grid jobs. Figure 1 shows the LDAP-tree of a cluster. The cluster entry describes its hardware, software, and middleware properties. A queue entry represents grid-enabled queues, and branches represent authorized user and job entries. Authorized user entries include information on each user, such as free CPUs and available disk space. Similarly, each grid job submitted to the queue is represented by a job entry, which is generated on the execution cluster. We thus implement a distributed job status monitoring system. The schema also describes storage elements and replica catalogs, albeit in a simplistic manner.

The information providers are small programs that generate LDAP entries when a user or service enters a search request. Our custom information providers create and populate the local database's LDAP entries by collecting information — about grid jobs, grid users, and the queuing system — from the cluster's batch system, the grid manager, and so on.

The information system's local databases provide clients with the requested grid information using LDAP and first-level caching of the providers' output. To this end, Globus created the grid resource information service, an LDAP back end. We use this back end as NorduGrid's local information database, and configure it to temporarily cache the output of NorduGrid providers.

The local databases use MDS's soft-state registration mechanism to register their contact information into the registry services, which can in turn register with other registries. This soft-state registration makes the grid dynamic, letting resources come and go. It also lets us create a specific topology of indexed resources. The registries, or index services, maintain dynamic resource lists with contact information for the soft-state-registered local databases. They might also perform queries by following the registrations and using a higher-level caching mechanism to cache the search results. For an index service, NorduGrid uses Globus's grid information index service, an LDAP back end.

In the NorduGrid testbed, we organized local databases and index services into a multilevel tree hierarchy through the soft-state registration to upper-level indices. The registries do not use the higher-level caching; they work as simple, dynamic "link catalogs," which reduces the overall system load. Clients connect to the (higher-level) index services only to find local databases' contact information, then query the resources directly.

Our goal with NorduGrid's hierarchy was to follow a natural geographical organization that grouped resources belonging to the same country together and registered them with the country's index service. These indices are further registered with NorduGrid's top-level index services. To avoid single points of failure, NorduGrid operates a multirooted tree with several top-level indices.

```
 &(executable="ds2000.sh")
(arguments="1101")
(join="yes")
(rsl_substitution=("TASK" "dc1.002000.simul"))
(rsl_substitution=
 ("LNAM"
  "dc1.002000.simul.01101.hlt.pythia_jet_17"))
(rsl_substitution=
 ("LC"
  "lc=DC1,rc=NorduGrid,dc=nordugrid,dc=org"))
(replicaCollection=ldap://grid.uio.no/$(LC))
(stdout=$(LNAM).log)
(inputfiles=("ds2000.sh"
 http://www.nordugrid.org/$(TASK).NG.sh))
(outputFiles=
($(LNAM).log
 rc://dc1.uio.no/2000/log/$(LNAM).log)
(atlas.01101.zebra
 rc://dc1.uio.no/2000/zebra/$(LNAM).zebra)
(atlas.01101.his
 rc://dc1.uio.no/2000/his/$(LNAM).his)
(jobname=$(LNAM))
(runTimeEnvironment="DC1-ATLAS")
```

Figure 2. An xRSL file. Extensions to RSL1.0 include `inputFiles` and `outputFiles` attributes that list URL pairs or the local–remote file name.

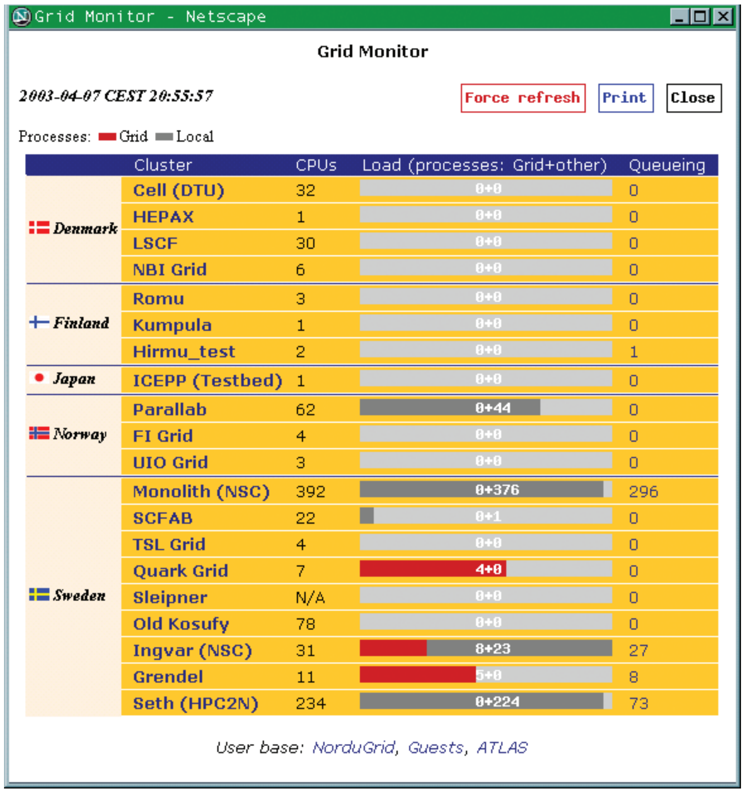
User Interface and Resource Brokering

The user interacts with NorduGrid through a set of command-line tools:

- `ngsub`: job submission
- `ngstat`: show status of jobs and clusters
- `ngcat`: display a running job's `stdout` or `stderr`
- `ngget`: retrieve output from a finished job
- `ngkill`: kill a running job
- `ngclean`: delete a cluster's output
- `ngsync`: recreate the user interface's local information about running jobs
- `ngcopy`: copy files to, from, and between storage elements and replica catalogs
- `ngremove`: delete files from storage elements and replica catalogs

(For detailed information about each command, see the user interface's user manual at our Web site.)

To submit a grid job using `ngsub`, the user describes the job using extended RSL syntax. This xRSL contains all required job information (the executable's name, the arguments to be used, and so on) and cluster requirements



Grid Monitor

2003-04-07 CEST 20:55:57

Processes: 🟥 Grid ▨ Local

	Cluster	CPUs	Load (processes: Grid+other)	Queueing
Denmark	Cell (DTU)	32	0+0	0
	HEPAX	1	0+0	0
	LSCF	30	0+0	0
	NBI Grid	6	0+0	0
Finland	Romu	3	0+0	0
	Kumpula	1	0+0	0
	Hirmu_test	2	0+0	1
Japan	ICEPP (Testbed)	1	0+0	0
Norway	Parallab	62	0+44	0
	FI Grid	4	0+0	0
	UIO Grid	3	0+0	0
Sweden	Monolith (NSC)	392	0+376	296
	SCFAB	22	0+1	0
	TSL Grid	4	0+0	0
	Quark Grid	7	4+0	0
	Sleipner	N/A	0+0	0
	Old Kosufy	78	0+0	0
	Ingvar (NSC)	31	8+23	27
	Grendel	11	5+0	8
	Seth (HPC2N)	234	0+224	73

User base: NorduGrid, Guests, ATLAS

Figure 3. The grid monitor. The monitor interlinks objects of the NorduGrid's information system, making them easily accessible.

(required disk space, software, and so on). The user interface then contacts the information system, first to find the available resources, and then to query each available cluster to match requirements. If the xRSL specification requires that input files be downloaded from a replica catalog, that catalog is contacted to obtain file information. Next, the user interface matches the xRSL-specified requirements with cluster information to select a suitable queue at a suitable cluster. When one is found, the user interface submits the job. Resource brokering is thus an integral part of the user interface, and does not require an additional service.

Resource Specification Language

NorduGrid uses Globus RSL 1.0 as the basis for communication between users, the user interface, and the grid manager.[7] Our RSL extensions include both new attributes and the ability to differentiate between two sets of attributes:

- *User-side RSL.* A user specifies the attribute set in a job description file, which the user interface interprets, modifies as necessary, and passes to the grid manager.
- *Grid manager-side RSL.* The grid manager interprets the preprocessed user interface attribute set.

xRSL uses the same syntax conventions as the core Globus RSL, although we changed some attributes' meaning and interpretation. The most notable changes are those related to file movement. The major challenge for NorduGrid applications is that they must pre- and post-stage many (often large) files. We thus added the `inputFiles` and `outputFiles` attributes, each of which lists the local–remote file name or URL pairs. The user interface uploads to the execution node the `inputFiles` that are local to the submission node; the grid manager handles the rest. Upon job completion, the grid manager moves `outputFiles` to the specified storage element. If no storage element is specified, the system expects users to retrieve the files through the user interface.

We also added several xRSL attributes for the convenience of users. Figure 2 shows a typical xRSL job-submission script. So far, xRSL seems sufficiently complex for job description, and the ease with which we can add new attributes is particularly appealing. We plan to use xRSL in further NorduGrid development.

Monitoring

Our grid monitor is realized as a Web interface to the NorduGrid information system (see Figure 3). The monitor lets users browse through all published information about the system, offering them both real-time monitoring and a primary debugging tool.

The grid monitor follows the information system's structure, displaying either a subset of object classes or the whole list, making them easily accessible to users as a set of windows associated with a corresponding module. Figure 3 shows the main grid monitor window. Most of the objects are linked to appropriate modules, so users can simply click the mouse to launch another module window and expand the available information about an object or attribute. Each new window is linked to other modules as well, which makes browsing intuitive.

Because the grid monitor's server runs on an independent machine, it imposes no extra load on NorduGrid, apart from the frequent LDAP queries. (The default refresh time for a single window is 30 seconds.)

Software Configuration and Distribution

The Grid's goal is multisite deployment, and many site administrators are involved. Because we can't expect all of these administrators to be grid experts, we made the NorduGrid Toolkit's configuration as simple as possible. Basically, it requires writing two configuration files: `globus.conf` and `nordugrid.conf`. The globus-config package uses `globus.conf` to configure the Globus Toolkit's information system from a single file. We developed this configuration scheme in collaboration with EDG, and it is not NorduGrid-specific. The `nordugrid.conf` file configures various NorduGrid Toolkit components.

Our approach has proven convenient, letting administrators set up sites as remote from Scandinavia as Canada or Japan in a matter of hours, with little help from NorduGrid developers.

Use Case: The Atlas Data Challenge

We initially designed the NorduGrid architecture for data-intensive tasks, such as those in experimental high-energy physics projects like the Atlas experiment (http://atlasexperiment.org). Atlas is scheduled to begin collecting data in 2007, at a rate of about 3 Petabytes per year. To prepare, Atlas is running a series of increasingly complex computing challenges to test its computing model and software suite, as well as those of existing grid solutions.

The first such challenge, Atlas Data Challenge 1 (DC1) consisted of large-scale physics simulations and ran from July through September 2002. Simulation participants included 39 institutes from around the world, including a group of Scandinavian researchers using NorduGrid. The NorduGrid team's task was to process two different data sets. In the first stage, the input data were 15 files (totaling 18 Gbytes) that were stored locally at different NorduGrid sites. The files had to be processed by a total of 300 simulation jobs that used 220 CPU-days for the simulations and produced 1,500 output files totaling about 450 Gbytes. After job execution, NorduGrid automatically uploaded the output files to a storage element in Norway and registered them in the replica catalog.

Figure 2 shows a typical xRSL job-submission script. The `executable` attribute specifies the executable running the simulation. In this case, `ds2000.sh` is a script downloaded from the URL that the `inputFiles` specify. This script calls Atlas's preinstalled physics simulation program, which runs on the input data that the `arguments` attribute specifies. To ensure that such a program exists, xRSL also requests the `runTimeEnvironment DC1-ATLAS`. The job is sent only to those clusters advertising this environment, which indicates that they have the required Atlas DC1 software installed. The grid manager uploads the indicated `outputFiles` to the specified output location — in this case, a physical location registered in the replica catalog collection and defined by the `replicaCollection` attribute so that the replica catalog will resolve on request (`rc://dc1.uio.no/log` to `gsiftp://dc1.uio.no/dc1/2000/log`).

DC1's second task was more challenging: the input data consisted of 100 files with a total volume of 158 Gbytes. Some NorduGrid sites could not accommodate all the files, and the team therefore decided to distribute file subsets. They then registered all distributed input sets into the replica catalog so that, during job submission, the broker could query the catalog for clusters that had the necessary input file for the corresponding physics simulation. However, the jobs were not exclusively data-driven; when requested input files were not found locally, the grid manager used the replica catalog information to download them into local temporary cache.

In all, the second task had 1,000 jobs that used about 300 CPU-days and produced 442 Gbytes of output. NorduGrid uploaded all output files to the dedicated storage element and registered them in the replica catalog.

NorduGrid's success in the Atlas DC1 task showed its reliability, and the Atlas collaboration's validation of the testbed's calculations confirm that the project achieved its goals.[11] As a matter of comparison, the EDG testbed could not reliably perform the task in preliminary tests, and thus was not used in DC1.

Conclusion

We developed NorduGrid in the Nordic countries' highly cooperative and efficient environment, which let us easily overcome various administrative, legal, financial, and technical obstacles. We plan to further extend and develop NorduGrid in keeping with the principles of simplicity, reliability, portability, scalability, and noninvasiveness. Among our future development plans are

- enabling an efficient and scalable distributed data-management system;
- further developing resource discovery and brokering to allow interoperability with complex

local scheduling policies;
- improving the authorization and authentication system, while maintaining its flexibility;
- modifying the information system to accommodate heterogeneous clusters, providing more sophisticated caching mechanisms; and
- developing and implementing bookkeeping and accounting services, as well as a user-friendly grid portal.

The NorduGrid Toolkit is freely available at www.nordugrid.org as RPM distributions, source tar-balls, and as CVS snapshots and nightly builds. We also offer a standalone local client installation, distributed as a tar-ball, which we designed as a NorduGrid entry point that works out-of-the-box.

Acknowledgments

The NorduGrid project is funded by the Nordic Council of Ministers through the Nordunet2 program, and by NOS-N. We are grateful to the system administrators across the Nordic countries for their courage, patience, and assistance in deploying the NorduGrid environment. In particular, we thank Ulf Mjörnmark and Björn Lundberg (Lund University), Björn Nilsson (Niels Bohr Institute, Copenhagen), Niclas Andersson and Leif Nixon (National Supercomputer Centre, Linköping), and Åke Sandgren (High Performance Computing Center North, Umeå).

References

1. A. Wäänänen, “An Overview of an Architecture Proposal for a High-Energy Physics Grid,” *Proc. Applied Parallel Computing* (PARA 2002), LNCS 2367, Springer-Verlag, 2002, pp. 76-86.
2. A. Konstantinov, “The NorduGrid Project: Using Globus Toolkit for Building Grid Infrastructure,” *Nuclear Instruments and Methods A*, vol. 502, Elsevier, 2003, pp. 407–410.
3. T. Sjöstrand and P.Z. Skands, “Baryon Number Violation and String Topologies,” *Nuclear Physics B,* vol. 659, nos. 1-2, 2003, pp. 243–298.
4. O.F. Syljuåsen, “Directed Loop Updates for Quantum Lattice Models,” *Condensed Matter*, e-print no. 0211513, 2002; http://arxiv.org/abs/cond-mat/0211513.
5. B. Allcock et al., “Secure, Efficient Data Transport and Replica Management for High-Performance Data-Intensive Computing,” *IEEE Mass Storage Conf.*, 2001; http://storageconference.org/2001/proceedings.html.
6. I. Foster et al., “A Security Architecture for Computational Grids,” *Proc. 5th ACM Conf. Computers and Security*, ACM Press, 1998, pp. 83–92.
7. K. Czajkowski et al., “A Resource Management Architecture for Metacomputing Systems,” Proc. *4th Workshop Job Scheduling Strategies for Parallel Processing.* LNCS 1459, Springer, 1998, pp. 62–82.
8. B. Kónya, “The NorduGrid Information System,” tech. manual, NorduGrid, www.nordugrid.org/documents/ng-infosys.pdf.
9. K. Czajkowski et al., “Grid Information Services for Distributed Resource Sharing,” *Proc. 10th IEEE Int’l Symp. High Perf. Distributed Computing*, IEEE Press, 2001, pp. 181–194.
10. M. Sgaravatto, *WP1 Inputs to the DataGrid Grid Information Service Schema Specification*, tech. report 01-TEN-0104-0 6, DataGrid, 2001.
11. P. Eerola et al., “ATLAS Data-Challenge 1 on NorduGrid,” to be published in *Proc. Computing in High Energy Physics*, World Scientific, 2003.

Paula Eerola is a professor of physics at Lund University, Sweden. Her research interests include B-mesons physics related to the Atlas experiment and physics beyond the standard model, application of grid technologies to high-energy physics data processing, and the Atlas Transition Radiation Tracker. She received a PhD in experimental particle physics from the University of Helsinki, Finland. Contact her at paula.eerola@hep.lu.se.

Balázs Kónya is a lecturer at Lund University, Sweden, and is working on the NorduGrid Project. His recent work in grid computing focuses on information system models and their implementations. He received a PhD in theoretical physics from the University of Debrecen, Hungary. Contact him at balazs.konya@hep.lu.se.

Oxana Smirnova is a docent (associate professor) in the Experimental High Energy Physics Department of Lund University, Sweden. Her recent interests include multiparticle dynamics and porting high-energy physics applications onto the Grid. She is a member of the Delphi and Atlas high-energy physics experiments, and participates in the Large Hadron Collider (LHC) Computing Grid, the EU’s DataGrid, and the NorduGrid projects. She received her PhD in physics from Lund University for experimental studies in quantum chromodynamics and interferometry. Contact her at oxana.smirnova@hep.lu.se.

Tord Ekelöf is a professor at the Department of Radiation Sciences of Uppsala University, Sweden, where he received his PhD in particle physics. He has worked with many experiments at the European Particle Physics Laboratory at CERN in Geneva, including the Delphi and Atlas detectors. He chairs the SweGrid project, aimed at building a grid infrastructure for research in Sweden. His other interests include developing and deploying various Atlas detector components, and searching for the Higgs particle. Contact him at tord.ekelof@tsl.uu.se.