# Grid Deployment Experiences: The path to a production quality LDAP based grid information system

L. Field, M. W. Schulz, CERN, Geneva, Switzerland

*Abstract*

This paper reports on the deployment experience of the de-facto grid information system, Globus MDS (Meta-data Directory Service) [1], in a large scale production grid and how this experience led to the development of an information caching system based on a standard OpenLDAP (Lightweight Directory Access Protocol) [2] database. The paper then describes how this caching system was developed further, from the results of performance and scalability tests, into a production quality information system. The generic information provider is also introduced and the reasons for its development explained.

## INTRODUCTION

The Globus Project is the self defined de-facto standard for grid computing [3]. Many grid projects around the world are based on the Globus Tool Kit 2 (GTK2) from Globus [4]. Once such project was the EU Datagrid (EDG) project [5]. The objective of the project was to provide a grid computing infrastructure for intensive computation and distributed data storage, across widely distributed scientific communities. This involved building on top of GTK2, higher level services that included: resource brokering, data management, grid monitoring and distributed mass storage.

GTK2 contains four core components; Grid Resource Allocation Manager (GRAM), GridFTP, Grid Security Infrastructure (GSI) and the Meta-data Directory Service (MDS) [3]. MDS is the grid information service. The data model for the information service is based on LDAP and the information that can be used in the information system is defined by an LDAP schema. The information system is made of three parts; information providers, Grid Resource Information Services (GRIS), Grid Information Index Services (GIIS) [1].

An information provider is a script that obtains static information from a configuration file and dynamic information about local services. This information is formated into LDAP Data Interchange Format (LDIF) and printed to stdout.

The GRIS is deployed on the same node as the information provider. The GRIS can be queried via an ldapsearch with a base dn of mds-vo-name=local,o=grid. When the GRIS is queried, it will execute the information provider, obtain the LDIF and return the result of the query. The GRIS can register itself with a GIIS.

A GIIS can be on the same machine as the GRIS but is usually found on another machine. The GIIS can be queried via an ldapsearch with a base dn set to the GIIS name. When the GIIS is queried, the back end will query all GRISes that have registered to the GIIS. A GIIS can register itself with another GIIS. All information can be found from one point by building up a hierarchy GIIS structure. To make the system more efficient, there is a caching mechanism built into the GIIS and GRIS. This makes the system more efficient but will also result in the information being slightly stale.

## EDG DEPLOYMENT

The EDG project had a development testbed consisting of five sites: CERN Switzerland, Ruthford Appleton Laboratory UK, CNAF Italy, NIKHEF Netherlands and IN2P3 France. The main building blocks, nodes, were the Computing Element (CE) and the Storage Element (SE). The CE is the interface to computing resources and the SE is the interface to storage. Each site in the testbed contained one CE and one SE. The CE and SE are the main sources of information in the grid information system. Both the CE and SE had a GRIS installed and an information provider. Each site ran a site GIIS and a region GIIS on the CE. The top level GIIS was located at CERN. (see Fig. 1).

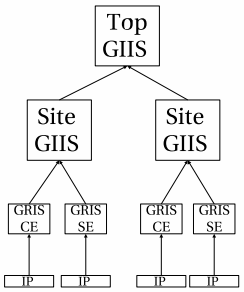


Figure 1: Initial Deployment

### *Initial Deployment Problems*

The information system is the central nervous system of any grid and without a working information system, the other grid middleware can not function.

There was one major problem with the information system that would stop the grid from functioning, queries to the top level GIIS would hang. This occurred if there were any problems in the lower levels of the hierarchy. Due to

the levels in the information system hierarchy not being completely decoupled, queries could to the top level GIIS could depend on an information provider being executed. If the information provider hung or took a long time to return, the whole information system would wait. MDS had a number of configurable timeout parameters to deal with such situation but if was found that no of these worked. Over a six month period the MDS code was investigated and a number of bugs were found that help fix some of these timeout problems.

After these timeout fixes, stress testing the information system caused the information system to hang again. With a query load on the top level GIIS of three queries per second and three sites, the information system would function. However, when a fourth site was added, the information system would hang. A test was conducted by David Groep, to compare MDS with the performance of a standard OpenLDAP database. Each site GIIS was queried and the returned LDIF inserted in to the OpenLDAP database. It was found that with all five sites in the OpenLDAP database, there were no problems, even with a query load of over ten queries per second.

### *Introducing the BDII*

As the standard OpenLDAP database had proved successful in the tests, it was decided that it should be used to replace of the top level GIIS. This was named the Berkley Database Information Index (BDII). Periodically, the BDII would query each site GIIS and use the returned LDIF to populate the OpenLDAP database. A timeout was included in the search in case a GIIS did not respond. The refresh time for the BDII was 20mins. In this mode of operation the BDII was viewed as a caching mechanism for the information system and although this was not an ideal solution, it produced stable information that could be used for testing the other grid system components.

## LCG DEPLOYMENT

LCG (LHC Computing Grid) [6], is the largest user of the EDG middleware. The goal of LCG is to deliver the computing infrastructure that is required by the four experiments in LHC: Alice, Atlas, CMS and LHCb. The production run will start in 2007 when the LHC accelerator is turned on. LCG will ramp up to this production by participating in a number of data challenges. LCG inherited the EDG code base and is endeavouring to run a production grid system with this code. LCG has fixed bugs found in the software and re-engineer's some of the code so that it will meet the production requirements.

### *BDII Re-engineering*

The BDII had not changed much since it had initially be written as a test. A few small modifications were required, however, it was decided that more time should be spent on re-engineering the BDII and change the BDII from a prototype to a production quality component.

The additional functionality that was added during this re-engineering were: the automatic update of the configuration and support for information provider scripts. The automatic update enables the configuration for the BDII to be updated via a web page. The configuration contains a list of LDAP URLs for the BDII to query. The automatic update will check a web page for an updated version of this configuration. The BDII also supports information providers. If the URL of an information provider is in the configuration file then the BDII will run it and obtain the LDIF output. This means that the BDII can also act as a GRIS as well as a GIIS.

Before the re-engineered BDII was deployed in the production system, a series of tests were conducted. These tests had two main objectives: firstly to ensure that the BDII was ready for the production system and secondly to understand its limits. The tests used a dual 1GHz Intel Pentium III machine with 512Mb of Ram.

### *Performance Testing*

The performance test measured the time to took to insert information into the BDII under different query loads. Information providers scripts were created by doing LDAP searches on the LCG1 production grid and writing the output to a file. A wrapper script would then print out the contents on the file thus simulating the real information in the grid information system. The ldapsearch was not used directly on the GRIS due to the varying time delays that occur when querying an MDS based grid information system. Three different entry points to the grid information were used: The top level (1 1.8Mb file), The regional level (3 600k files), The site level (24 75k files). All entry points produced the same 1.8M of information, 658 ldap entries for 24 sites. For each test two times were measured. The time it takes to add all the entries to an empty database and the time it takes to update a populated database.

Table 1: No query load

| **Level** | **Add** | **Modify** |
|---|---|---|
| Top | 20s | 7s |
| Region | 29s | 7s |
| Site | 16s | 9s |

Table 2: 5 query streams

| **Level** | **Add** | **Modify** |
|---|---|---|
| Top | 21s | 12s |
| Region | 40s | 28s |
| Site | 20s | 15s |

Table 3: 10 query streams

| Level | Add | Modify |
|---|---|---|
| Top | 24s | 16s |
| Region | 50s | 39s |
| Site | 24s | 17s |

### *Stress Testing*

The stress test involved populating the BDII whilst simultaneously querying the database with 10 parallel streams. The information provider used to populate the BDII was the same the 1.8Mb file used in the performance testes. The information provider was queried by the BDII every 30 seconds. The BDII query load process would fork off 10 queries and wait for them to return and then rest for 1 second before querying again. The test ran for over two weeks, in which time over 2 million queries on the database had be done with no corruption of the database.

### *BDII Deployment in LCG*

The performance tests showed a difference in the BDII population time for the different levels. This time differs both with the size of the data and the number of streams. There is a finite speed at which data can been added to the database. There is also an overhead for creating the connection to the information source. From the results, it seems that it is best either to read all the data from one source or read small amounts of data from many sources. As such it was decided that for deployment the top level BDII should query each site GIIS.

The BDII was deployed in a production environment. This showed up a few minor bugs which were fixed and the BDII was gradually hardened to the production environment. As the number of sites reached 50, the information in the BDII no longer seemed to be consistent. An investigation into this showed that LDAP queries were queueing up and due to a configurable limit in the database new queries were being rejected. The queries were being queued due to the time that it took to update the database. Read and write operations were occurring simultaneously and the write operation was taking so long that the number of queueing queries would increase.

The stability of the BDII showed up some instability within the lower levels of the information system. It was decided that all GIISes should be replaced by BDIIs. A site BDII is identical to a top level BDII but a site BDII only contains information about a site. It obtains this information by querying the GRIS on each resource at the site.

### *Further BDII Improvements*

The performance tests showed that there is a difference between adding data to an empty database and updating the database. In the BDII code, if an entry modification is tried on an entry that does not exist, an error is generated and the entry will have to be created. The creation of this entry generated many calls to the database. To remove this problem, the database would always be updated from an empty database. The entries would be sorted by the length of the dn and inserted shortest first. This way, the parents will always be added before the child.

The performance test also showed that putting a query load on the database would increase the time that it took to update the database. This increase in time caused the queries to queue when the number of sites reached 50. For this reason the BDII was changed to use two databases. One read only and one write only. When the write database has been updated it is be swapped with the read only database. This decouples the reads and writes from the same database and hence removes the problem.

### *Site Scalability Test*

The site scalability test showed how many sites the BDII can support. A script was created that would populate a slapd server with example information (17.1kb) that represented one site. This script was used to start 50 slapd servers using different ports on one machine. The BDII was updated and the time taken for the update measured. If the test was successful another machine was added that also had 50 slapd server running. The results of this test show that the BDII can easily cope with the amount of information produced from 1000 sites. The time taken to update the database increases linearly with the number of sites.

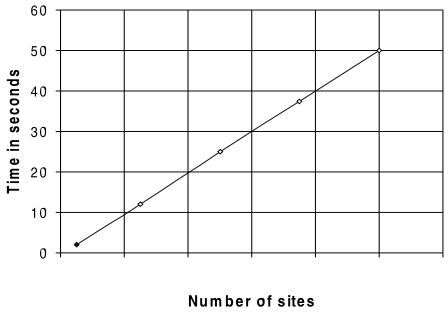


Figure 2: Results of Scalability Tests

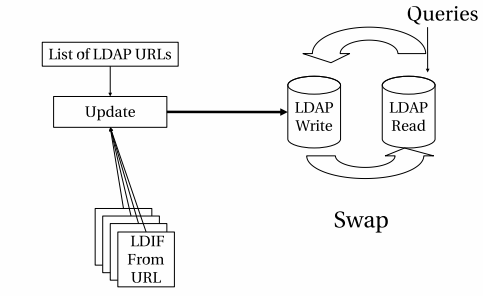


Figure 3: BDII Architecture

## THE GENERIC INFORMATION PROVIDER

The Generic Information Provider (GIP) was developed to help LCG support multiple systems. Using the EDG information providers would require writing a new information provider, including configuration, for each system that had to be supported.

The GIP is a framework whereby a common configuration can be used to produce the static information and dynamic plug-ins are used to obtain the dynamic values. The static information is created taking values from a configuration file and using a template that corresponds to the structure, create a static LDIF file. If there is no dynamic information then the GIP will simply read the static information and print the output to stdout. If there is a dynamic plug-in, the GIP will run this to obtain the dynamic information then overwrite the static value with the dynamic value when being printed to stdout.

One of the main advantages of the GIP is that it makes a clear separation between static and dynamic data. This separation, along with the concept of the plug-in, enables the GIP to be easily adapted to produce information about any system. By using a common framework to configure the static information, the plug-ins remain small and system specific.

### *GIP Deployment in LCG*

When running the dynamic plug-ins, the GIP waits for a period of time and if the plug-in did not return it within this period, it would return the static defaults. The results from the deployment showed that on some systems, eg a batch system with 500+ nodes, the dynamic plug-in would take quite some time to run. The reason for this was that the underlying query to the batch system was slow. A caching mechanism was built into the GIP that would be used for all dynamic plug-ins. The GIP forks of a process for the dynamic plug-in and the dynamic plug-in will write its output to a cache. This means that if the dynamic plug-in is taking a long time to return, the GIP can respond much quicker by used the old information that is in the cache. There is a timeout for the cache whereby if it is too old the static defaults will be used.

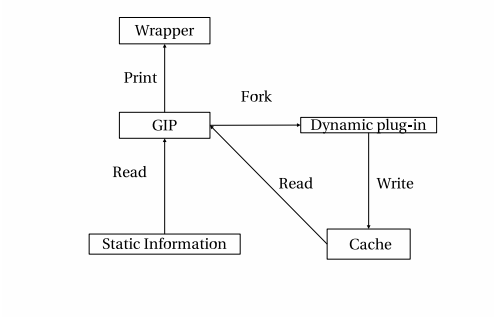


Figure 4: GIP Architecture

## CONCLUSION

From the deployment experience in EDG and LCG, MDS is unusable in a production environment as a grid information system. Using a standard OpenLDAP database and a small script update the database, it is possible to build a production quality grid information system that supports 1000 sites. This highlights a few points for successful software development within a grid environment.

- Build on good implementations of will established standards.
- Concentrate on the small subset of core functionality.
- Follow a good quality control procedure.
- Try an keep everything simple.

The results of the tests show that the best way of using the BDII is for it to query many data sources. The limitation on the number of sites a BDII can support is two fold. As the BDII spawns off a process for querying each site, the number of sites will be limited by the number of processes that the operating system can spawn off itself. The second limitation is the amount of data in the system, as the time it takes will increase as the amount of data increases and hence the period between updates will also increase. This fundamentally means that more sites, and hence information in the grid, the more stale the information will need to be. This is true for all grid information systems. Even if improvements are made to the update speed and hence the data is made less stale, more data will always lead to stale data.

The GIP makes it easy to support multiple systems in an extensible way. The configuration is simplified and only new plug-ins are required to be written to support new systems. The built in caching mechanism ensures that that the dynamic information is always readily available and that the freshness of this information only dependent on the underlying speed of the system that is queried.

The BDII, along with the GIP, have been used to build a production quality information system for LCG.

## REFERENCES

[1] http://www.globus.org/mds

[2] http://www.openldap.org

[3] http://www.globus.org

[4] http://www.globus.org/gt2.4/

[5] http://eu-datagrid.web.cern.ch/eu-datagrid/

[6] http://lcg.web.cern.ch/LCG/

[7] Lee Momtahan and Andrew Martin, “e-Science Experiences: Software Engineering Practice and the EU DataGrid”, in Proc. Asia-Pacifi c Software Engineering Conference, IEEE Press, 2002.

[8] F.Etienne, C.Loomis, S.Traylen, “Evaluation of Testbed Operation”, DataGrid-06-D6.6-0120-1-1, EUDG, 2003.