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

L. Field, M. 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 the results of 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[3] is the self defined de-facto standard for Grid computing. 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 objectives of the project was to provide a Grid computing infrastructure for intensive computation and distributed data storage, across widely distributed scientific communities. The main goal of the project was to build on top of GTK2, higher level services that included: resource brokering, data management, Grid monitoring and distributed mass storage. The resulting middleware, GTK2 and the EDG middleware, would enable large scale Grids to be used by experiments in the three main application areas, High Energy Physics (HEP), Earth Observation (EO) and Bioinformatics (BIO). As the EDG middleware was built upon GTK2, the quality of GTK2 would directly affect the quality of the EDG middleware.

The GTK2 contains four core components; Grid Resource Allocation Manager (GRAM), GridFTP, Grid Security Infrastructure (GSI) and the Meta Directory Service (MDS). MDS is the Grid information service. The data model for the information service is based on LDAP. 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.

An information provider is a script that obtains static information from a configuration file and also dynamic information about other local services. This information arranged into the ldif format and the ldif is printed to stdout.

The GRIS is deployed on the same node as the information provider. The GRIS will execute the information provider and obtain the resulting ldif from stdout. The information stored in the GRIS can be queried by an ldapsearch on the machine running the GRIS, ensuring that the correct port is used and the bind dn is mds-vo-name=local,o=grid. The GRIS can register itself with a GIIS.

A GIIS can be on the same machine but is usually found on another machine. A GIIS is given an mds-vo-name other than local. If a GIIS is queried by using an ldapsearch, ensuring the correct port and bind dn, the back end in the GIIS will then query the GRIS, the information provider will be executed and the information will be flow back through the components to the client that queried the GIIS. A GIIS can register itself with another GIIS. This enables a hierarchy to be built so that all information can be found by querying one point. To make the system more efficient, there is a cache mechanism built into the GIIS and GRIS. If another query is executed before the cache time-out period, then the information is returned from the cache and thus makes the system more efficient. However, this will also result in the information being slightly stale.

INITIAL 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, for the EDG project were the Computing Element (CE) and the Storage Element (SE). The CE is the interface to computing resources eg. a gateway to a batch system. The SE is the interface to storage, either a Mass Storage System (MSS) or a distributed storage system. Each site in the testbed contained at least one CE and one SE. The CE and SE are the main source 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 country GIIS, both of these on the CE. The top level GIIS was located at CERN. (see Fig. 1).

Hanging Problems

The first noticeable problem that occurred was that the top level GIIS would always hang when queried. It was found that this was due to problems in the lower levels of the hierarchy. When the top level GIIS is queried, it subsequently queries all the GIIS's that are registered to it and waits for the response. As one or more of the lower GIIS's was not responding the query would just hang. This prob-

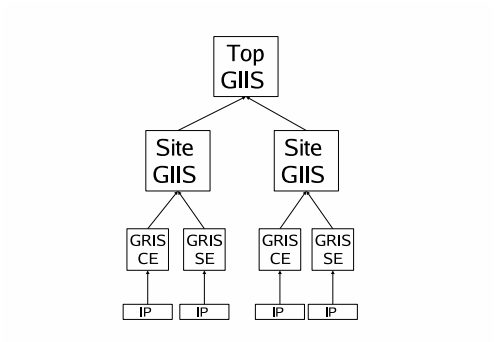

Figure 1: Deployment

lem was also observed at all levels in the hierarchy. If an information provider hung, the GRIS would not respond and all higher level GIISs would also hang. MDS had a number of configurable timeout parameters. One such parameter was the timeout for the query. It was found that this timeout did not work. For over six months the MDS code was investigated and a number of bugs were found that help fix some of the hanging problems.

Scaling Problems

Once the hanging GIIS problem was fixed another problem in the information system was found. When stress testing the information system it was found that the information system would again hang. With a stress on the system of 3 queries per second and 3 sites in the information system everything work fine. However when a fourth site was added, the top level GIIS would hang. A test was conducted to find out if this was a problem with MDS or a limit in OpenLDAP. A script was produced that would query the site GIIS's and add this information to an OpenLDAP database. It was found that with all five sites in the OpenLDAP database, there were no problems, even when being queried over 10 times per second. However when the same test was tried with MDS, the queries would hang.

Introducing the BDII

As the standard OpenLDAP database had proved successful in the tests, it was decided that an Information Index would be used for the top level MDS node. This piece of software was named the Berkley Database Information Index (BDII). The BDII was hand configured with the site GIIS's. Periodically the BDII would query each site GIIS and populate the OpenLDAP database. A timeout was included in the search in case a country GIIS did not respond. The refresh time for the BDII was 30mins. In this mode of operation the BDII was viewed as a caching mechanism for the information system. Although this was not an ideal solution, it produced stable information that could be used for testing the other Grid system components.

LCG

LCG (LHC Computing Grid), is the largest user of the software delivered by EDG. 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 has the freedom to do what is necessary to provide a production grid that is usable by the experiments. 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.

Further Development

The BDII had not changed much since it had initially be written as a test. A few small modifications were still required, however it was decided that more time should be spent on re-engineering the BDII. The reasons for this were; to ensure the code meet the quality requirements for LCG code development, improve the configuration, simplify the code where possible, improve the logic and generally change the BDII from a prototype to production quality.

The additional functionality that was added during this re-engineering were: the automatic update of the configuration and support for information provider scripts. This 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.

INITIAL TESTING

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

Performance Testing

These tests involve gathering information from the information provider scripts and pushing that data into the OpenLDAP database via the BDII. The information providers scripts were created by doing LDAP searches on the LCG1 production grid and writing this 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 reason why the ldapsearch was not used directly on the GRIS is 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, old BDII: 1 file of 1.8Mb. The regional level, GIIS: 3 files, 780k, 823k and 217k. The site level, GRIS: 24 files, CE 71k and SE 5k.

All the three entry points produced the same 1.8M of information, 658 ldap entries for 24 sites.

For each test two times were measured and recorded. The first and largest time is how long it takes to add all the entries to an empty database. The second is how long it takes to do an update on that 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

This test involved inserting information from an information provider script and pushing the data into the OpenLDAP database via the BDII, whilst simultaneously querying the database by 10 parallel streams. The information provider script was the same the top level script as used in the performance testes, 1.8Mb of data from LCG1. The information provider script was run every 30 seconds and the data pushed into the database. The query 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 LDAP database.

Initial Deployment

The performance tests showed a differences in the time for the different levels are due to two factors. The size of the data and the number of streams. Each stream will read separately the LDIF, then insert the LDIF into the database. 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 on top level BDII should be used that connects to 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 hardend to the production enviorment. As the number of sites reached 50, the BDII information in the BDII no longer seemed to be consistent. The investigation into this showed that LDAP queries were queueing up and due to a configarable limit in the database new queries were being rejected. The reason why the queries were being queue was due to the time that it took to update the database. Read and Write operations were occuring simultaneously and the write operation was taking so long that the number of queueing queries would increase. Reading and writing to the database simultaneously significantly decrease performance.

The stability of the BDII showed up some instabilty within the lower levels of the information system. For the possibilty of replaceing the site GIIS with a site BDII was looked into. After some trials at afew major sites it was decided that all the site GIIS would be replace by site BDII. A site BDII is idential to a top level BDII. A site BDII only contains information about a site. It obtains this information by queryig the GRIS on each resource at the site.

Further imporvements

The performance tests showed that there is a difference between the adding of the data to a clean database and updating the database. This difference was due to the overhead of creating the LDAP nodes in the database. Each entry has a dn which describes the position in the database hierarchy. If a modification is tried on an object that does not exist an error is generated and the object will have to be created. The creation of this object generates many calls to the database as all the parent nodes also need to be created. The database would be updated from an empty database, the entries would be sorted by the length of the dn and inserted sortest first. This way the parents will always be added before the child.

The performance test showed that puting a query load on the database would increase the time that it took to update the Database. This increase in time cause a problem when the number of sites reasched 50. For this reason the BDII was changed to use two databases. One read only and one write only. When the write database had been updated it would be swapped with the read only database. This would

decouple the reads and writes from the same database and hence remove the problem.

Site Scalability Test

To test that the BDII would work for many more sites than were available a simulated scalibity test was required. The aim of this test was to see how many sites the BDII could support. A script was created that would populate a slapd server with example information (17.1kb) that would be produced from one site. This script could be used to start 50 slapd servers using different ports on one machine. The BDII would then be 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 (in the table below) of this test show that the BDII can easily cope with the amount of information produced from 1000 sites although the time that is taken to update the database increases. The limit on the number of site is therefore dictated by the required freshness information.

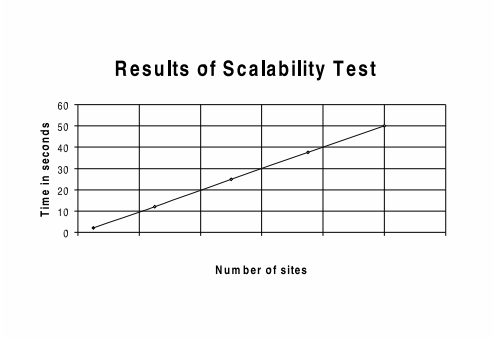

Figure 2: Scalibility

Final BDII Architecture

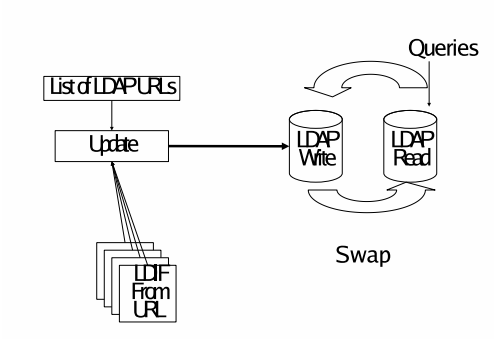

Figure 3: BDII Architecture

THE GENERIC INFORMATION PROVIDER

The Generic Information Provider (GIP) was deveoped for one reason. LCG had to support muliple systems and using the current set of information providers would require writting a new information provider for each system. As most of the information produced by and information provider is static it seemed to be excess work.

The GIP is a framework where by a common configuration can be used from producing the static information and dynamic plug-ins are used to obtain the dynamic values. The static information is created by using a configuration file with the values, a configuration script and templates that correspond to the structure. The configuration is only require to be done once to create the static information. If there is no dynamic information then the GIP will simply read the static information and print the output to standard out. 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 standard out.

On 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 can remain small and system specific.

Results from deployment

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 reson for this was that the underlying query to the batch system would take quite some time. A caching mechanism was built into the GIP that would be used for all dynamic plug-ins. The GIP will fork 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 where if it is too old the static defaults will be used.

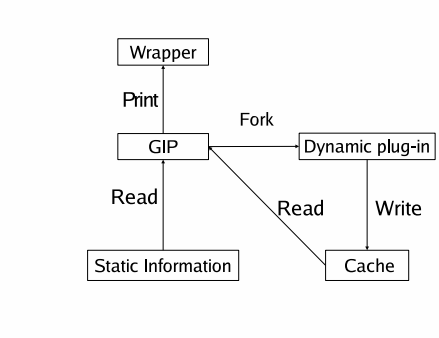

Figure 4: GIP Architecture

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 wrapper script to source the grid information and update the database, it is possible to build a production quality grid information system. This highlights a few points for successful software. Build on a good implementation of established standards. Concentrate the small subset of the core functionality. Follow a good quality control procedure where the results of thourgh testing are fed back into the deveopment of the core functionlity.

The results of the tests that the best way of using the BDII is for it to query directly each site.

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. 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.

REFERENCES

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

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

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

[4] http://www.

[5] http://www.edg.org