EVOLUTION OF LCG-2 DATA MANAGEMENT

J-P. Baud, J. Casey, CERN, Geneva, Switzerland

Abstract

The LCG-2 middleware was used in the spring 2004 data challenges by all four LHC experiments. This produced the first useful feedback on scalability and functionality problems in the middleware, especially with regards to data management.

In light of the feedback from the data challenges, and in conjunction with the LHC experiments, a strategy for the improvements required in the data management area was developed. The aim of these improvements was to allow both easier interaction and better performance from the experiment frameworks and other middleware such as POOL.

In this paper, we will cover the problems and issues highlighted by the data challenges, as well as the required improvements to allow LCG-2 to handle effectively data management at LHC volumes. In particular, we will highlight the new and improved services provided.

EXPERIMENT FEEDBACK FROM DATA CHALLENGES

The data challenges in 2004 was the first real experiment production use of LCG-2. During the course of the data challenges, many issues and problems were exposed that had not shown up before in more limited tests. The deployment, service and development teams worked closely with the experiments to understand these issues. Some of the problems were solved during the data challenges, with new versions of software being deployed in agreement with the experiments during the data challenge. Other problems required more work to resolve, and indeed some exposed fundamental problems with the middleware as deployed in LCG-2.

The problems within the data management software can be broken down into two main areas - performance problems and missing functionality.

Performance Problems

The EDG Data Management tools [1] were mainly written in Java, both on the client and server side. Indeed, although some C++ APIs were provided for the Local Replica Catalog (LRC) and Replica Metadata Catalog (RMC), the command line tools for the catalog were in Java. This lead to large start up overheads on the client side, that meant the command line tools, while useful for interactive work, were effectively useless for production usage.

Also, the EDG Replica Manager, a pure client-side application, was written in Java. The slow start up times, along with the lack of support for bulk operations, again meant that it was often too slow for production usage.

The catalog servers also showed performance issues. There were issues with missing indexes on some columns in the database, and some common but expensive queries that needed to be cached on the server. The rest of the server performance problems were due to missing functionality and architectural problems, described in the next section.

Missing functionality

The EDG catalogs were designed with a flat logical namespace. This meant that the only effective way to select subsets of the Logical File Name (LFN) namespace was via wildcards (e.g. 'lfn:/foo/*'). This did not scale, since it implied a full table scan of the database table which held the logical file names. Also, due to the stateless nature of the Web Service interface, queries which returned a large number of result which had to 'paged' through, required the same query to be run on the database backend many times (in fact it is run once per ‘page’ of results). This multiplied the load on the database significantly.

Additionally, the catalogs did not provide user-controllable transactions to the user applications. This meant that any rollback of a partially successful multi-file operation had to be carried out manually by the user application.

Problems were also seen in replication of files, especially files used by many jobs on different worker nodes. Since the replication tools had no way of tracking the state of ongoing transfers, some files would be transferred many times to the same site by concurrent jobs.

Finally, there was no managed storage solution in LCG-2 - The EDG 'Classic SE' was the standard storage element. The 'Classic SE' is a gridftp server, with some metadata stored in the information system which defines mount points for different Virtual Organisations (VO) in the file system. This had scaling problems since all requests to a storage element go through a single machine running a single gridftp server.

GFAL AND LCG_UTIL

The Grid File Access Library (GFAL) originally was introduced into the LCG middleware as a low-level IO interface to Grid Storage, specifically Storage Resource Managers (SRM) [2]. It is written in C, and provides a set of POSIX like methods for file system interaction. In order to interact with grid storage, it also needed to interface with the EDG grid catalogs (LRC & RMC) as well as the grid information system.

Once we had a requirement to replace the replica management tools, GFAL was a good base for the development. The solution took the form of a set of command line tools, collectively referred to as 'lcg_utils', which provided the same command line arguments and functionality as the Replica Manager. The same

functionality was also provided as a C API for direct integration into experiment tools.

GFAL (and thus lcg_utils) was extended to communicate with other storage element types, such as the EDG 'Classic SE' and the EDG Storage Element, deployed as an interface to the Atlas Data Store (ADS) RAL.

During the data challenges, experiments asked for extra features in the replica management tools. These were generally added only to lcg_utils, which now has a richer feature set than the EDG replica manager, as well as better performance. GFAL and lcg_utils are still being actively developed; for instance we have recently released a thread safe version at the request of the ATLAS experiment.

LCG FILE CATALOG

The main lesson from the data challenges was that the catalogs deployed in LCG-2 did not meet either the performance or functionality requirements of the experiments. Thus it was agreed to upgrade the catalogs in LCG-2, as an interim measure in order to allow experiments to carry out future intensive work using LCG-2. The aim was to undergo a rapid development and deployment cycle, in order to quickly provide a short-term solution to the experiments.

The catalogs are based on an existing and mature code base, and supports both Oracle and MySQL as database components. The main improvements that the new catalogs have over the EDG catalogs are:

- Cursors for large queries, reducing database load
- Timeouts and retries are implemented in the client. This reduces the impact of momentary loss of contact to the service for remote sites
- There is a user exposed transaction API. This allows the experiment frameworks to explicitly rollback transactions, as well as transactions rolling back automatically on error
- A true hierarchical namespace is provided on Logical File Names along with the corresponding namespace operations.
- Integrated GSI Authentication and Authorization. There will no longer be any insecure access to the grid catalogs. A user certificate will be required for all interactions.
- Access Control Lists (ACLs) We provide both standard UNIX permissions and POSIX compliant ACLs
- Checksums are provided to aid in maintenance of replica consistency

A prototype of the catalog has been created and undergone functional testing. Also, integration with GFAL and lcg_utils has been completed. Integration with both ROOT [3] and the POOL File Catalog Interface [4] is in progress, and will be provided during autumn 2004.

Initial performance and scalability testing is currently underway, and is looking promising. A first version will be deployed for certification during October 2004, with migration of experiment data in the existing catalogs happening in October and November 2004.

Performance Comparison

A first performance comparison with the EDG catalogs has been carried out. The tests consisted of recording the average insert time of a Logical File Name and GUID with varying number of entries in the catalog, and varying number of client threads. In the tests, the number of server threads in the LFC was fixed at 20. The results for the EDG catalog and the LFC are shown in Figure 1, 2 and 3.

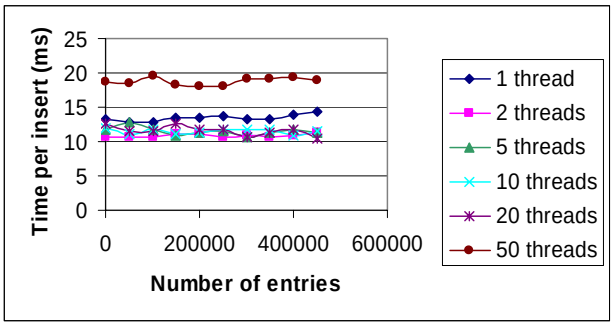

Figure 1: LCG File Catalog – average insert time

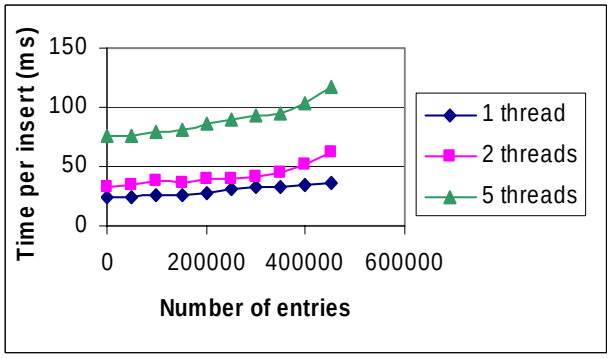

Figure 2: EDG Metadata Catalog – average insert time

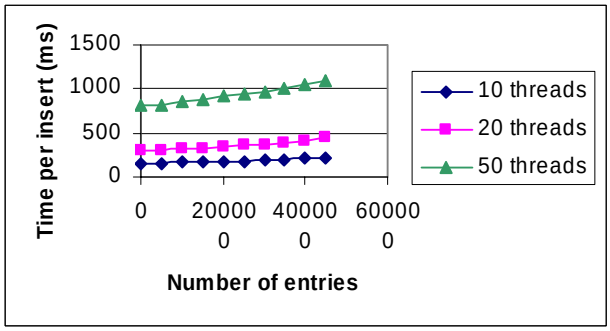

Figure 3: EDG Metadata Catalog – average insert time

We see that for the EDG catalog, the average insert time is less than 30ms for 1 or 2 threads, but that it degrades rapidly with both the number of entries and the number of threads. The new catalog shows insert times of less than 25ms for up to 50 concurrent client threads. We note that the new catalog seems to show no degradation in performance when we increase the number of entries in

the database. Also it scales much better with increasing numbers of clients.

ROBUST DATA TRANSFER

The need for a highly performing and reliable data transfer service is driven by LCG 'Robust Data Transfer' Service Challenge.

The aim of the Robust Data Transfer service challenge is to prototype the data movement services that will be needed for LHC. Many of the components that are required exist at the current time, but have not yet been shown to work together at the required performance and reliability. It is important to start building up the knowledge of how the entire system performs when exposed to the level of usage we expect during LHC running.

The required data transfer rates for Tier-0 to Tier-1 traffic for the LHC are large; current estimates are upwards of 50 Gb/s in total to all Tier-1s. It is very important that we achieve these levels well in advance of the real data so we are sure that the system works, is manageable and maintainable.

The project will work towards testing the interaction of the full set of services at all levels:

- network
- disk to disk file transfer
- reliable file transfer service
- mass store to mass store file transfer
- grid components - catalog, replication tools, etc.

Initially, we will start with a simple system to test the disk to disk file transfer systems and the reliable file transfer service. Once this is working at the required levels of reliability and performance, we can add in additional components (e.g. MSS interaction). The service will be managed and coordinated by LCG Grid Deployment at CERN.

Status

The initial sites involved are CERN, FNAL, BNL, NIKHEF/SARA, IN2P3 and FZK. DESY are also interested in collaborating in the project, and have started to set up the required infrastructure.

At CERN, the setup currently consists of, currently, 10 dual Itanium2 machines each with a 1Gb link that gets aggregated into a 10Gb switch. From this switch we have a 10 Gb connection to GEANT (and another 10Gb link to Chicago). Some sites are connected directly with dedicated bandwidth to the CERN cluster; FNAL have a 10Gb link and NIKHEF/SARA have 1Gb currently, with the possibility of 10Gb.

Transfers will be done using gridftp, with either plain gridftp servers or a SRM. First transfers were carried out from CERN to Fermi in the last week of September 2004, with the other participating sites starting transfers during October 2004.

DISK POOL MANAGER

Recent experience and current thinking leads to using the SRM as a common interface to storage at grid sites. There are three distinct cases:

- Tier-0/Tier-1 sites with hierarchical MSS. These sites usually make the integration with their own MSS. This is currently the case at CERN and FNAL, with CASTOR and dCache/ENSTORE respectively.
- Large Tier-1s sites with large disk pools (10's TBs distributed between many fileservers). These sites need a flexible system which can encompass many different configurations of disk systems and transfer servers. Currently dCache [5] provides a good solution, but it has been seen that it needs effort to integrate and manage.
- Sites with smaller disk pools (1–10 TBs) and with less available management effort. Currently no such solution exists that is lightweight to both install and manage.

In order to solve the problem for the third class of sites, it was decided to design and develop a lightweight Disk Pool Manager (DPM) within the LCG project. This is complementary to dCache as a solution in LCG-2 and we

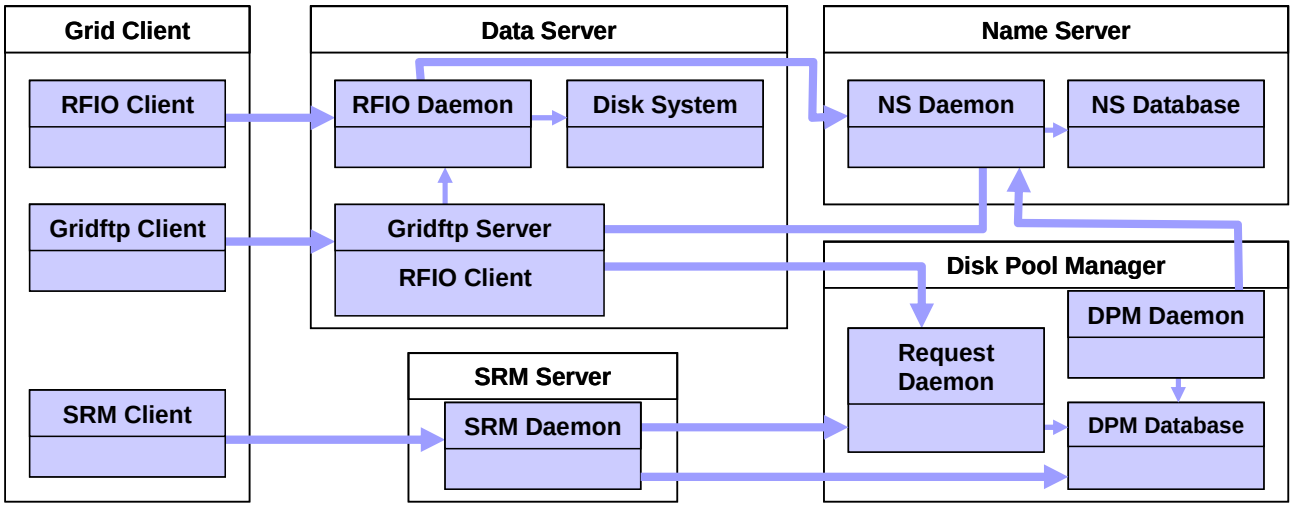

Figure 5: The architecture of the Disk Pool Manager

believe that having both available allows sites to choose an appropriate solution their particular organisation.

Aims and Objectives

As stated above, the aim of the DPM is to provide a managed disk solution for the small Tier-2s in LCG-2. This implies scaling to between 1 to 10 TB of storage, with the disk space possibly spread over several disk servers at the site. There is a strong requirement to focus on manageability, with the DPM being both easy to install and easy to configure.

We also want to improve over the problems seen with the 'Classic SE'. This implies that we should provide space reservation so that space on the storage element can be reserved at the start of the job and guaranteed to be available at the end of the job. Also important is support for multiple replicas of a file within the disk pools, in order to avoid 'hotspots' on particular disks.

Also we provide two different types of storage space – volatile and permanent, as defined by the SRM specification. This allows for both long term storage and scratch space close to the worker nodes.

Manageability

In order to make the service easy to install and manage, there are few daemons to install. An installation will consist of

- Disk Pool Manager daemon
- Name Server daemon
- SRM daemon
- IO Daemon (e.g. gridftp, RFIOD, XROOTD)

There are no central configuration files, and disk nodes send messages to the Disk Pool Manager daemon when they want to add their disk to the pool. This makes it easy to both add and remove disks and partitions to the simple. For instance, an administrator can temporarily remove file systems from the DPM if a disk has crashed and is being repaired. Also, the DPM automatically configures a file system as "unavailable" when it is not contactable

Features

The DPM provides secure GSI authenticated and authorized access to data via several different interfaces:

- Direct Socket interface
- SRM v1
- SRM v2 Basic

It also offers a large part of SRM v2 Advanced, including space reservation, namespace operations, permissions, copy and remote get/put. The I/O access to the data itself can be done via several protocols: Gridftp, RFIO and ROOT I/O. The overall architecture of the DPM is shown in Figure 4.

It should be noted that the DPM allows for the possibility of its catalog acting as a 'Local Replica Catalog' in a distributed catalog system, without requiring an additional local site catalog to be deployed.

INTERACTION OF LCG-2 COMPONENTS WITH GLITE

File Catalog

The LCG File Catalog provides an immediate solution to experiment needs for LCG-1. In order to not disrupt ongoing experiment grid activities, the EDG Catalogs will be migrated to LFC Catalog for LCG-2 and both systems will be run in parallel to allow the experiments to test the interoperability of their tools.

This migration will help the later migration to gLite, since it will tackle how to map existing Logical File Names into a hierarchical namespace. Alternatively, a gLite web service interface could be provided on top of the new catalog to allow for interoperability with both LCG-2 and gLite data management tools.

GFAL and lcg_utils

The abstractions inside lcg_utils and GFAL allow for easy addition of alternative components. This is seen in that the next version will interact with both EDG and LCG File Catalog. We believe that adding support for the gLite catalogs should be similarly straightforward.

Robust Data Transfer

The architecture of the transfer management system used in the transfer challenge was created in conjunction with gLite team. Both systems share a common database schema for interoperability, and gLite client tools will work with the underlying infrastructure developed for the Service Challenge. We see the work from the two teams is complementary - The service challenge focuses on performance and scalability while gLite focuses on user-visible functionality and VO policies.

Disk Pool Manager

The DPM will be used for new storage at Tier-2 level. We will provide tools for 'Classic SE' migration. This will only require transferring the file system metadata into the DPM catalog. This data would then be available from within gLite via SRM interfaces.

REFERENCES

[1] D. Cameron et al, "Replica Management in the European DataGrid Project," Journal of Grid Computing 2004, In Print.

[2] SRM Working Group home page. http://sdm.lbnl.gov/srm-wg/

[3]The ROOT home page http://root.cern.ch/

[4] C. Cioffi et al, "POOL File Catalog, Collection and Metadata Components," CHEP 2003, La Jolla California, March 24-28 2003.

[5] P. Fuhrmann, "dCache, the commodity cache," Twelfth NASA Goddard and Twenty First IEEE Conference on Mass Storage Systems and Technologies Spring 2004, Washington DC.