UNIVERSITY
of
GLASGOW

Department of Physics & Astronomy
Experimental Particle Physics Group
Kelvin Building, University of Glasgow,
Glasgow, G12 8QQ, Scotland
Telephone: +44 (0)141 339 8855 Fax: +44 (0)141 330 5881

GLAS-PPE/2005-06
1st July 2005

LCG Data Management: From EDG to EGEE

Jean-Philippe Baud[2], James Casey[2], Sophie Lemaitre[2],
Caitriana Nicholson[1], David Smith[2], **Graeme Stewart**[1]

[1] University of Glasgow, Glasgow, G12 8QQ, Scotland

[2] CERN, European Organisation for
Nuclear Research, 1211 Geneva, Switzerland

Abstract

The Large Hadron Collider (LHC) at CERN, the European Organisation for Nuclear Research, will produce unprecedented volumes of data when it starts operation in 2007. To provide for its computational needs, the LHC Computing Grid (LCG) is being deployed as a worldwide computational grid service, providing the middleware upon which the physics analysis for the LHC will be carried out.

Data management middleware will be a key component of the LCG, enabling users to analyse their data without reference to the complex details of the computing environment.

In this paper we review the performance tests of the LCG File Catalog (LFC) and make comparisons with other data management catalogs. We also survey the deployment status of the LFC within the LCG.

UK e-Science All Hands Meeting 2005
Nottingham, UK

1 Introduction

The Large Hadron Collider (LHC) at CERN, the European Organisation for Nuclear Research, will produce unprecedented volumes of data when it starts operation in 2007. To provide for its computational needs, the LHC Computing Grid (LCG) is being deployed as a worldwide computational grid service, providing the middleware upon which the physics analysis for the LHC will be carried out.

Data management middleware will be a key component of the LCG, enabling users to analyse their data without reference to the complex details of the computing environment.

In this paper we review the performance tests of the LCG File Catalog (LFC) and make comparisons with other data management catalogs. We also survey the deployment status of the LFC within the LCG.

2 EDG Replica Location Service

The European Data Grid (EDG) produced data management middleware consisting of the Replica Metadata Catalog (RMC) and the Local Replica Catalog (LRC). Together these formed the Replica Location Service (RLS), which was deployed in the LCG in 2003-4.

The 2004 series of LHC experiment data challenges were the first to use the LCG-2 set of middleware tools in a realistic environment [1]. Feedback from the experiment groups after the data challenges highlighted various problems and limitations, as well as differences between the expected and actual usage patterns.

During these challenges it became apparent that the file catalog infrastructure was too slow both for inserts and for queries [2]. Queries involving both the LRC and RMC were particularly slow [3]. Missing functionality identified included lack of support for bulk operations and transactions. It also became clear that queries were generally based on metadata attributes and were not simple lookups of a file's physical location. On the other hand, users did not use the web services approach in the way which had been anticipated when the EDG components were developed. They were implemented such that a remote procedure call (RPC) was performed for each low-level operation; users, however, wanted to send higher-level or multiple commands in a single RPC. As this was not available, the cumulative overheads from a large number of low-level RPCs led to considerable loss of performance. Also, although a C++ API was available, command-line tools were available only in Java which led to added loss of performance due to the overhead in starting up the Java Virtual Machine with each call.

3 LCG File Catalog

To address the problem of the EDG RLS the LHC Computing Grid has designed a new data management component, the LCG File Catalog (LFC). The LFC moves away from the Replica Location Service model used in previous LCG releases, towards a hierarchical filesystem model and provides additional functionality.

The LFC has a completely different architecture from the RLS framework. Like the EDG catalog, it contains a GUID (Globally Unique Identifier) as an identifier for a logical file, but unlike the EDG catalog it stores both logical and physical mappings for the file in the same database. This speeds up operations which span both sets of mappings [4]. It also treats all entities as files in a UNIX-like filesystem. The API is designed to mimic a UNIX filesystem API, with calls which are intuitive to the user.

The main entities of the LFC design are shown in Figure 1. There is a global hierarchical namespace of Logical File Names (LFNs) which are mapped to the GUIDs. GUIDs are mapped to the physical locations of file replicas in storage (Storage File Names, SFNs). System attributes of the replicas (such as creation time, last access time, file size and checksum) are stored as attributes on the LFN, but user-defined metadata is restricted to one field. Multiple LFNs per GUID are allowed as symbolic links.

Transactions and cursors are supported for handling large query results, with bulk operation support planned. As there is only one catalog, transactions are possible across both LFN and SFN

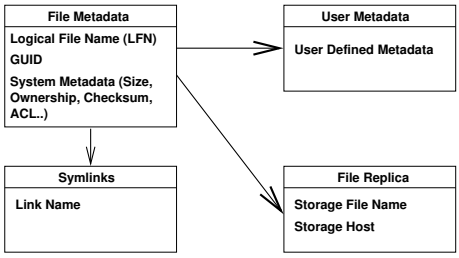

Figure 1: Components of the LFC.

operations, which was impossible with the EDG RLS. In case of momentary loss of connection to the catalog, timeouts and retries are supported. Authentication is by Grid Security Infrastructure (GSI), which will allow single sign-on to the catalog with users' Grid certificates.

The LFC has been intergrated with the Pool Of Persistant Objects for LHC (POOL) framework.

4 LFC Performance Tests

4.1 Test Setup

Performance tests of the LFC were written in `C`, using the LFC's API. For each test the client forked the required number of threads. A test harness was written in `perl` to run each batch of tests and collate the results.

The LFC server was running on a dual 1GHz PIII machine with 512MB of RAM, connected to an Oracle database server running on a dual 2.4GHz P4 machine. Clients ran on a single processor PIII and all machines were networked through 100Mb switches.

4.2 Insert

The mean time to insert a single entry into the LFC was measured as the number of entries in the catalogue was increased, for a single client thread. Starting from a low number of entries in the LFC new entries were inserted and the mean of every 1000 entries was then taken and plotted as shown in Figure 2.

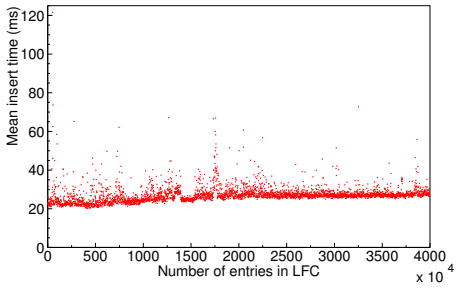

Figure 2: Mean insert time with increasing catalogue size.

There is very little dependence of insert time on the number of entries in the catalogue. Up to about 5 million entries, the mean insert time is about 22 ms, after which it increases slightly to a plateau of about 27 ms. Ocassionaly delays on the network are interpreted as the cause of the long tail on the distribution.

4.3 Transactions

In the LFC, the transaction methods are exposed in the API so that the user can explicitly start a transaction, perform a number of operations then commit to the database (or abort and roll back if there is an error). To test transaction performance, the number of operations inside a single transaction was varied and the mean insert time measured for different numbers of client threads and 20 server threads. Figure 3 shows the results.

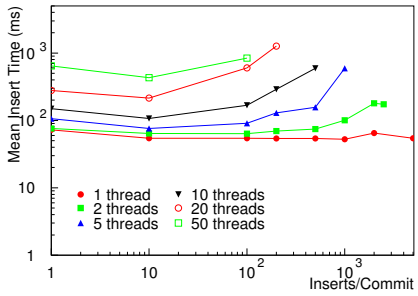

Figure 3: Mean insert time when using transactions, with increasing number of inserts per transaction.

This shows that explicitly using transactions increases the insert time by at least a factor of two; with a single client thread at the optimal number of operations per transaction, the mean insert time is about 54 ms. Analysis of transactions indicates that performace loss occurs in the Oracle database.

4.4 Inserting with `chdir`

An important feature of the LFC is the ability to change directories using a `chdir` operation, and in particular to change into the current working directory. When performing a large number of operations within the same directory, it should be faster to change into that directory before performing all the operations, as permissions would only be checked once.

The mean of 5000 inserts was measured for increasing numbers of subdirectories in the path of the LFN to be inserted, with and without performing the directory change, and for various numbers of client threads.

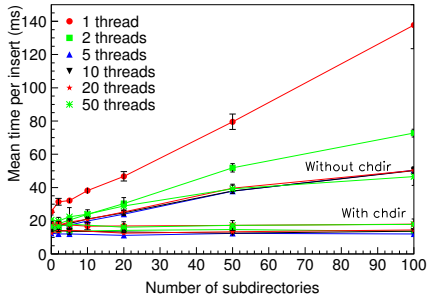

Figure 4: Mean insert time with and without chdir, with increasing number of subdirectories in the path.

From Figure 4, we see that the `chdir` is working as intended. When the absolute path name is used the mean insert time increases fairly linearly with the number of subdirectories in the path.

When `chdir` is used, the mean insert time is independent of the number of subdirectories. The results show that the performance gain from using `chdir` is significant.

4.5 Queries

Querying the catalogue for entries will be the most common operation after the initial production phase at the LHC so query performance is very important. In the following tests, a query is defined as finding a given LFN in the catalogue and performing a `stat()` operation on it. This returns all the associated metadata such as file size, checksum, last modification time and so on, as well as checking permissions.

The query rate was measured with an increasing number of client threads. As in the insert rate and delete rate experiments, the server was running with 20 threads and there were about 1 million entries in the LFC and as Figure 5 shows, the pattern of results is also similar. A maximum rate of about 275 queries per second is reached, which is slightly higher than the insertion rate.

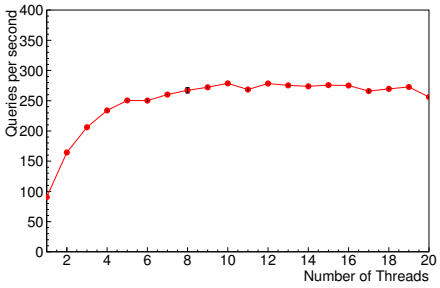

Figure 5: Query rate for increasing number of client threads, LFC with 1 million entries.

4.6 Reading directories

Reading through files in a directory is an important use case. The total time to perform such a `readdir` operation was therefore measured for increasing numbers of files in a directory, from 1 to 10000, with the results shown in Figure 6. The time it takes to read the whole directory is directly proportional to the number of files in the directory, until the number of client threads matches the number of server threads. At this point, the time spent by a thread waiting for another thread to finish dominates, until the directory is sufficiently large (about 2000 files) for the read time to dominate.

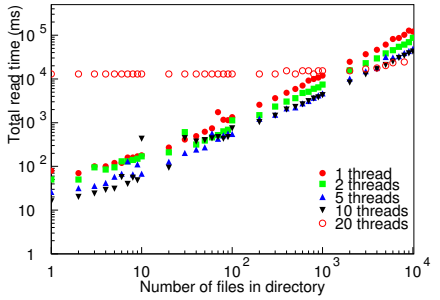

Figure 6: Total read time with increasing directory size, for varying number of client threads, 20 server threads.

5 Comparison with Globus and EDG File Catalogs

The EGG and Globus Replica Location Services (RLS) are built on a different model to that of the LFC. Local file catalogs (LRCs) store information local to a site, and publish this information to a higher level catalog, the Globus Replica Location Index (RLI). (N.B. the EDG RLS made use of a Replica Metatdata Catalog, RMC, and was never deployed with an RLI)

It should be emphasised that these different catalog designs address different issues, the RLS catalogs can be deployed in a distributed fashion (which can scale better), however this comes at the expense of making queries involving both catalogs very slow [3]. The LFC also offers features considered essential by the LHC community, such as cursors and transactions, not offered by the RLS catalogs.

5.1 Inserts

It is considered essential to maintain data integrity for LHC experiments, which requires flushing the database after each insert. In this case the Globus LRC can achieve 84 inserts per second [5]. For the EDG RLS the best insert time is 17ms [6], corresponting to 60 inserts per second for a single thread. Tests with the LFC show insert rates rising from 75/s for a single client up to 220/s for multiple client threads.

5.2 Queries

Queries on the Globus RLS are extremely rapid, reaching rates of 2000/s [5]. EDG single thread queries are slower, 60/s [6]. The LFC can query at 275/s, however this performs a full `stat` on an entry and checks permissions along the directory path, where as the Globus RLS only does a lookup. Analysis of LHC use suggests that lookups alone are rare [3] and that returning metadata is essential.

6 Deployment Status

The LCG File catalog has alreay been deployed as a production file catalog by ZEUS at DESY, SEE-GRID in Greece and TW Grid in Taiwan. A pre-production service for LHC experiments has been installed at CERN to enable performance testing and integration with experiment frameworks. LHCb will use a central catalog at CERN and Atlas are evaluating the LFC for local cataloging within LCG.

As part of LCG Service Challenge 3 LFC has been installed at all Tier 1 and participating Tier 2 centres.

7 Conclusions

The LCG File Catalog is a replacement for the EDG RLS system. It offers the features required by users but unavailable in other replica management systems.

Performance measurements show the robustness and scalability of the LFC up to many millions of entries and hundreds of client threads.

The LFC is now deployed as a production service by several VOs, and is undergoing performace and integration tests by LHC experiments.

References

[1] A. Delgado Peris, *et al.* LCG-2 User Guide. Technical Report CERN-LCG-GEDIS-454439, CERN, Geneva, Switzerland, September 2004.

[2] A. Fanfani *et al.* Distributed Computing Experiences in CMS DC04. In *CHEP*, Interlaken, Switzerland, September 2004.

[3] M. Girone et al. Experience with POOL in the LCG Data Challenges of Three LHC Experiments. In *CHEP*, Interlaken, Switzerland, September 2004.

[4] J-P. Baud and J. Casey. Evolution of LCG-2 Data Management. In *CHEP*, Interlaken, Switzerland, September 2004.

[5] A. Chervenak *et al.* Performance and Scalability of a Replica Location Service. In *HPDC-13*, Honolulu, Hawaii, USA, 2004.

[6] D. Cameron. Replica Management and Optimisation for Data Grids. PhD Thesis, University of Glasgow, 2005.