

# Data Management Services in the European DataGrid Project

David Cameron<sup>2</sup>, James Casey<sup>1</sup>, Leanne Guy<sup>1</sup>, Peter Kunszt<sup>1</sup>,  
Sophie Lemaitre<sup>1</sup>, Gavin McCance<sup>2</sup>, Heinz Stockinger<sup>1</sup>, Kurt Stockinger<sup>1</sup>,  
Giuseppe Andronico<sup>3</sup>, William Bell<sup>2</sup>, Itzhak Ben-Akiva<sup>4</sup>, Diana Bosio<sup>1</sup>,  
Radovan Chytracsek<sup>1</sup>, Andrea Domenici<sup>3</sup>, Flavia Donno<sup>3</sup>, Wolfgang Hoschek<sup>1</sup>,  
Erwin Laure<sup>1</sup>, Levi Lucio<sup>1</sup>, Paul Millar<sup>2</sup>, Livio Salconi<sup>3</sup>,  
Ben Segal<sup>1</sup>, Mika Silander<sup>5</sup>

<sup>1</sup> CERN, European Organization for Nuclear Research, 1211 Geneva, Switzerland

<sup>2</sup> University of Glasgow, Glasgow, G12 8QQ, Scotland

<sup>3</sup> INFN, Istituto Nazionale di Fisica Nucleare, Italy

<sup>4</sup> Weizmann Institute of Science, Rehovot 76100, Israel

<sup>5</sup> University of Helsinki, Finland

## Introduction

The European DataGrid project [2] was charged with providing a Grid infrastructure for the massive computational and data handling requirements of several large scientific experiments. The size of these requirements brought the need for scaleable, robust data management services and within the European DataGrid project, creating these services was the task of Work Package 2.

The data management services were designed to follow the web services paradigm, and were implemented in Java. The following services make up Work Package 2's replica management system: Replica Location Service, Replica Metadata Catalog, and the Replica Optimization Service. The primary interface between users and these services is the Replica Manager client [1]. In this paper we discuss the architecture and functionality of these components and analyse their performance. Results show that they can handle user loads as expected and scale well.

## Replication Services

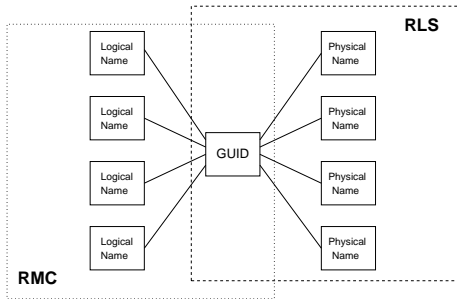
The design of the replica management system is modular, with several independent services interacting via the Replica Manager, a logical single point of entry to the system for users and other external services. Each service has been designed and deployed as a web service and runs on Apache Axis inside a Java

servlet engine. The Replica Manager coordinates the interactions between all components of the systems and uses the underlying file transport services for replica creation and deletion. Query functionality and cataloging are provided by the Replica Metadata Catalog and Replica Location Service (RLS). Optimized access to replicas is provided by the Replica Optimization Service, which aims to minimize file access times by directing file requests to appropriate replicas.

Grid Unique IDentifiers (GUIDs) are guaranteed unique identifiers for data on the Grid. In the RLS each GUID is mapped to one or more Physical File Names (PFNs), which represent the physical location of each replica of the data. However the GUIDs stored in the RLS are neither intuitive nor user friendly. The Replica Metadata Catalog (RMC) allows the user to define and store Logical File Name (LFN) aliases to GUIDs and store GUID metadata such as file size, owner and creation time. Many LFNs may exist for one GUID but the LFN must be unique within the RMC. The relationship between LFNs, GUIDs and PFNs and how they are stored in the catalogs is summarised in Figure 1.

## Performance Results

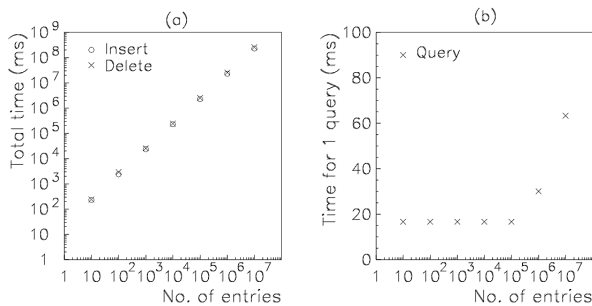
Work Package 2 services were individually tested for performance and scalability under stressful conditions and some results are presented in this section. Clients for the services are available in three forms: C/C++



**Figure 1. Relationship between LFNs, GUIDs and PFNs.**

API, Java API, and a Command Line Interface (CLI).

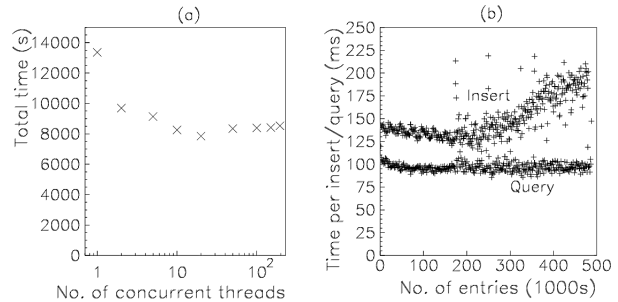
Firstly, the C++ client was tested using a test suite which inserts a number of GUID:PFN mappings into the RLS, queries for one GUID and then deletes the mappings. This tests how each of these operations on the RLS scales with the number of entries in the catalog. Figure 2(a) shows the total time to insert and delete up to 10 million mappings, and Figure 2(b) shows how the time to query one entry varies with the number of entries in the RLS.



**Figure 2. (a) Total time to add and delete mappings and (b) query the RLS.**

The results show that insert and delete operations have stable behaviour, in that the total time to insert or delete mappings scales linearly with the number of mappings inserted or deleted. A single transaction takes about 25 - 29 ms with the tendency that delete operations are slightly slower than inserts. The query time is independent of the number of entries in the catalog up to around 1 million entries, then it slowly increases by about a factor of 3 up to 10 million entries. This is due to the underlying database, which takes longer to query the more entries it contains.

Taking advantage of the multiple threading capabil-



**Figure 3. (a) Total time to add 500,000 mappings and (b) comparison of insert and delete times in the RLS.**

ities of Java, it was possible to simulate many concurrent users of the catalog and monitor the performance of the Java API. The total time to insert 500,000 mappings was measured for different numbers of concurrent threads (Figure 3(a)). The time falls rapidly with increasing numbers of threads, bottoming out at 10 - 20 threads. Although the time for an individual operation is slower the more concurrent operations are taking place, the overall throughput increases.

Figure 3(b) compares insert time and query time for a RLS storing between 0 and 500,000 entries. This test was done with 10 concurrent threads, where at any given moment 5 threads would be inserting a mapping and 5 threads would be querying a mapping. The plot shows the insert time rising from 140 ms to 200 ms but the query time stays at a constant 100 ms and does not vary with the number of entries.

## Conclusion

We have briefly described the design and architecture and examined the performance of the data management services provided to the European DataGrid project by Work Package 2. The web services model was used to create a set of independent replication, cataloging and optimization services which performance analysis has shown can cope under stressful conditions and scale well with increasing user load.

## References

- [1] Diana Bosio et al. Next-Generation EU DataGrid Data Management Services. In *Computing in High Energy Physics (CHEP 2003)*, La Jolla, California, USA, March 2003.
- [2] The European DataGrid Project. <http://www.edg.org>.