Replica Management in the European DataGrid Project †

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

[1] *CERN, European Organization for Nuclear Research, CH-1211 Geneva 23, Switzerland*

[2] *University of Glasgow, Glasgow, G12 8QQ, Scotland*

[3] *INFN, Istituto Nazionale di Fisica Nucleare, Italy*

[4] *Weizmann Institute of Science, Rehovot 76100, Israel*

[5] *University of Helsinki, Finland*

Abstract.

Within the European DataGrid project, Work Package 2 has designed and implemented a set of integrated replica management services for use by data intensive scientific applications. These services, based on the web services model, enable movement and replication of data at high speed from one geographical site to another, management of distributed replicated data, optimization of access to data, and the provision of a metadata management tool. In this paper we describe the architecture and implementation of these services and evaluate their performance under stressful Grid conditions.

1. Introduction

The European DataGrid (EDG) project 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 scalable and robust data management services. Creating these services was the task of EDG Work Package 2.

The first prototype data management system was implemented early in the lifetime of the project in C++ and comprised the edg-replica-manager [17] based on the Globus toolkit and the Grid Data Mirroring Package (GDMP) [18]. GDMP was a service for the replication (mirroring) of file sets between Storage Elements and together with the edg-replica-manager it provided basic replication functionality.

After the experience gained from deployment of these prototypes and feedback from users, it was decided to adopt the web services

† This work was partially funded by the European Commission program IST-2000-25182 through the European DataGrid Project.

paradigm [22] and implement the data management components in Java. The second generation replica management system now includes the following services: the Replica Location Service, the Replica Metadata Catalog, and the Replica Optimization Service. The primary interface between users and these services is the Replica Manager client.

In this paper we discuss the architecture and functionality of these components and analyse their performance. The results show that they can handle user loads as expected and scale well. Work Package 2 (WP2) services have already been successfully used as production services for the LHC Computing Grid [15] in preparation for the start of the next generation of physics experiments at CERN in 2007.

The paper is organised as follows: in Section 2 we give an overview of the architecture of the WP2 services, in Section 3 we describe the replication services in detail and Section 4 evaluates their performance. Section 5 discusses directions of possible future work and related work and conclusions are in Section 6.

2. Design and Architecture

The Work Package 2 replica management services are based on web services and implemented in Java, adhering to the following principles [9, 14]:

- Modularity: Data management components were designed to be modular so that plug-ins and future extensions are easy to apply and each service can operate independently of the others.
- Evolution: The future directions and standards for Grid services are not well defined, but several groups are working to implement a framework for Service Oriented Computing ([12, 13]). The data management components were designed to be flexible with repsect to these evolving areas and the design allowed for an easy adoption of these concepts.
- Deployment: A vendor neutral approach was adopted for all components to allow for different deployment scenarios. The data management services are independent of the underlying operating system and have been tested on Tomcat and the Oracle 9i Application Server, interfacing to MySQL and Oracle database back-ends.

2.1. Web Service Design

Web service technologies [22] provide an easy and standardized way to logically connect distributed services via XML (eXtensible Markup Language) messaging. They provide a platform and language independent way of accessing the information held by the service and, as such, are highly suited to a multi-language, multi-domain environment such as a DataGrid.

All the data management services have been designed and deployed as web services and run on Apache Axis [3] inside a Java servlet engine. All services use the Java reference servlet engine, Tomcat [4], from the Apache Jakarta project [20]. The Replica Metadata Catalog and Replica Location Service have also been successfully deployed into the Oracle 9i Application Server and are being used in production mode in the LCG project [15].

The services expose a standard interface in WSDL format [23] from which client stubs can be generated automatically in any of the common programming languages. A user application can then invoke the remote service directly. Pre-built client stubs are packaged as Java JAR files and shared and static libraries for Java and C++, respectively. WP2 also provided C++ clients, which provide significant performance benefits and are built based on the gSOAP toolkit [21]. Client Command Line Interfaces are also provided.

The communication between the client and server components is via the HTTP(S) protocol and the data format of the messages is XML, with the request being wrapped using standard SOAP Remote Procedure Call (RPC). Persistent data is stored in a relational database management system. Services that make data persistent have been tested and deployed with both open source (MySQL) and commercial (Oracle 9i) database back-ends, using abstract interfaces so that other RDBMS systems can be easily slotted in as well.

3. 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. 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. 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.

The Replica Manager is implemented as a client side tool. The Replica Metadata Catalog, Replica Location Service and the Replica Optimization Service are all stand-alone services, allowing for a multitude of deployment scenarios in a distributed environment. One advantage of such a design is that if any service is unavailable, the Replica Manager can still provide the functionality that does not make use of that particular service. Also, critical service components may have more than one instance to provide a higher level of availability and to avoid service bottlenecks.

The downside is that a lot of the coordinating logic is happening at the client side so that asynchronous interaction is not possible. Also, in case of failures on the client side, the user is left with little possibilities, there is no way to automatically re-try the operations, the user has to deal with these cases himself. Before a Replica Management Service can be provided we need a secure fine-grained delegation mechanism by the means of which the client could safely request the service to perform a well defined limited set of tasks. Full delegation is available but that gives little advantage over simple authentication which is built into the services already.

3.1. Replica Manager

The replica management system also calls upon services external to WP2. An Information Service such as MDS (Monitoring and Discovery Service) or R-GMA (Relational Grid Monitoring Architecture) needs to be present, as well as storage resources with a well-defined interface, in our case SRM (Storage Resource Manager) or the EDG-SE (EDG Storage Element). The Replica Manager also makes use of transport mechanisms such as GridFTP.

For the user, the main entry point to the Replication Services is through the Replica Manager client interface that is provided via C++ and Java APIs and a command line interface. The actual choice of the service component to be used can be specified through configuration files. Java dynamic class loading features are exploited to make them available at execution time in the Replica Manager.

3.2. Replica Location Service

In a highly geographically distributed environment, providing global access to data can be facilitated via replication, the creation of remote read-only copies of files. In addition, data replication can reduce access latencies and improve system robustness and scalability. However, the existence of multiple replicas of files in a system introduces additional issues. The replicas must be kept consistent, they must be locatable and their lifetime must be managed. The Replica Location Service (RLS) is a system that maintains and provides access to information about the physical locations of copies of files [10].

The RLS architecture defines two types of components: the Local Replica Catalog (LRC) and the Replica Location Index (RLI). The LRC maintains information about replicas at a single site or on a single storage resource, thus maintaining reliable, up to date information about the independent local state. The RLI is a (distributed) index that maintains soft collective state information obtained from any number of LRCs.

Grid Unique IDentifiers (GUIDs) are guaranteed unique identifiers for data on the Grid. In the LRC each GUID is mapped to one or more physical file names identified by Storage URLs (SURLs), which represent the physical location of each replica of the data. The RLI stores mappings between GUIDs and the LRCs that hold a mapping for that GUID. A query on a replica is a two stage process. The client first queries the RLI in order to determine which LRCs contain mappings for a given GUID. One or more of the identified LRCs is then queried to find the associated SURLs.

An LRC is configured at deployment time to subscribe to one or more RLIs. The LRCs periodically publish the list of GUIDs they maintain to the set of RLIs that index them using a soft state protocol, meaning that the information in the RLI will time out and must be refreshed periodically. The soft state information is sent to the RLIs in a compressed format using bloom filter objects [8].

An LRC is typically deployed on a per site basis, or on a per storage resource basis, depending on the site's resources, needs and configuration. A site will typically deploy 1 or more RLIs depending on usage patterns and need. The LRC can also be deployed to work in stand-alone mode instead of fully distributed mode, providing the functionality of an replica catalog operating in a fully centralized manner. In stand-alone mode, one central LRC holds the GUID to SURL mappings for all the distributed Grid files.

3.3. Replica Metadata Catalog Service

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. Many LFNs may exist for one GUID but the LFN must be unique within the RMC.

The relationship between LFNs, GUIDs and SURLs and how they are stored in the catalogs is summarised in Figure 1.

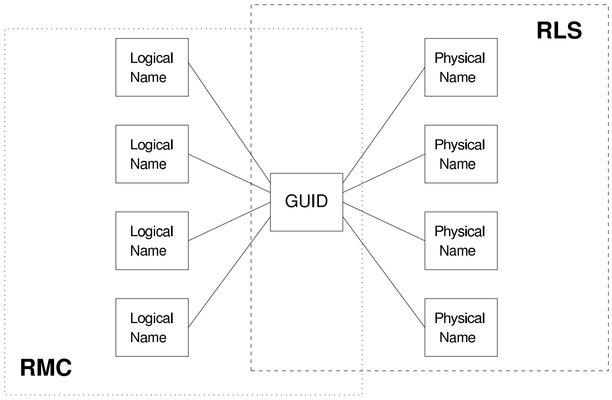

Figure 1. The Logical File Name to GUID mapping is maintained in the Replica Metadata Catalog, the GUID to physical file name (SURL) mapping in the RLS.

In addition, the RMC can store GUID metadata such as file size, owner and creation date. The RMC is not intended to manage all generic experimental metadata however it is possible to extend the RMC to maintain O(10) items of user definable metadata. This metadata provides a means for a user to query the file catalog based upon application-defined attributes.

The RMC is implemented using the same technology choices as the RLS, and thus supports different back-end database implementations, and can be hosted within different application server environments.

The reason to provide a separate RMC service from the RLS for the LFN mapping is the different expected usage patterns of the LFN and replica lookups. The replica lookup happens at application runtime when the user actually needs to access the file and at job scheduling time when the locations of the replicas need to be known. The LFN and GUID mapping and the corresponging metadata are used by the users for preselection of the data to be processed.

3.4. Replica Optimization Service

Optimization of the use of computing, storage and network resources is essential for application jobs to be executed efficiently. The Replica Optimization Service (ROS) [6] focuses on the selection of the best replica of a data file for a given job, taking into account the location of the computing resources and network and storage access latencies.

The Network Monitor provides the API that is used by the ROS to obtain information on network latencies between the various Grid resources. This information is used to calculate the expected transfer time of a given file with a specific size. The ROS can also be used by the Resource Broker to schedule user jobs to the site from which the data files required can be accessed in the shortest time.

The ROS is implemented as a light-weight web service that gathers information from the European DataGrid network monitoring service and performs file access optimization calculations based on this information.

4. Evaluation of Data Management Services

Grid middleware components must be designed to withstand heavy and unpredictable usage and their performance must scale well with the demands of the Grid. Therefore all the Work Package 2 services were tested for performance and scaleability under stressful conditions. Results of these tests are presented in this section and they show the services can handle the loads as expected and scale well.

Clients for the services are available in three forms: C/C++ API, Java API, and a Command Line Interface (CLI). It was envisaged that the CLI, typing a command by hand on the command line of a terminal, would be mainly used for testing an installation or individual command. The APIs on the other hand would be used directly by applications' code and would avoid the need for the user to interact directly with the middleware. Tests were carried out using all three clients for each component and as the results will show, using the API gives far better performance results than using the CLI. The reasons for this will be explained in this section.

The performance tests were run on the Work Package 2 testbed, consisting of 13 machines in 5 different sites. All the machines had similar specifications and operating systems and ran identical versions of the WP2 middleware. The application server used to deploy the services was Apache Tomcat 4 and for storing data on the server side, MySQL was used. For most of the performance tests small test applications were developed; these are packaged with the software and can therefore be re-run to check the results obtained. Note that these tests were all performed using the non-secured versions of the services (i.e. no SSL handshake).

4.1. Replica Location Service

Within the European DataGrid testbed, the RLS so far has only been used with a single LRC per Virtual Organization (group of users collaborating on the same experiment or project). Therefore results are presented showing the performance of a single LRC.

Firstly, the C++ client was tested using a test suite which inserts a number of GUID:SURL mappings, queries for one GUID and then deletes the mappings. This tests how each of these operations on the LRC 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 LRC.

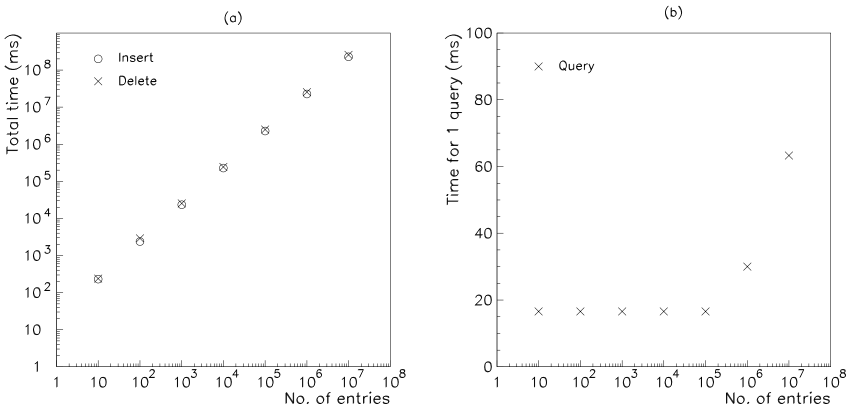

Figure 2. (a) Total time to add and delete mappings and (b) query the LRC using the C++ API.

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 with a single client thread 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, when it tends to increase. This is due to the underlying database, which takes longer to query the more entries it contains.

Taking advantage of the multiple threading capabilities of Java, it was possible to simulate many concurrent users of the catalog and monitor the performance of the Java API.

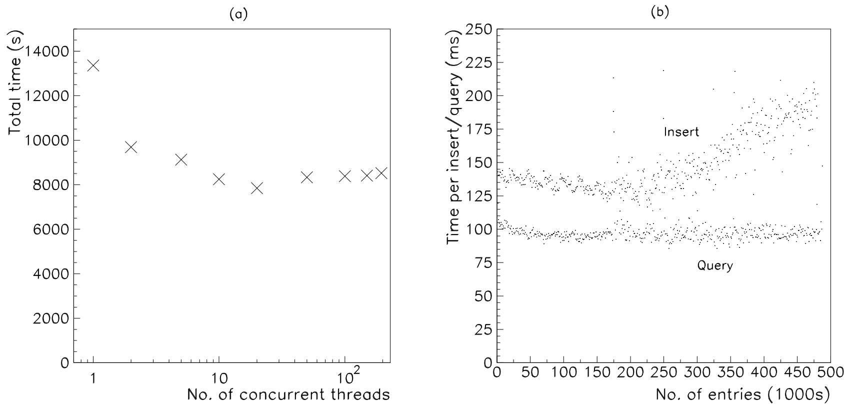

Figure 3. (a) Total time to add 500,000 mappings to the LRC using concurrent threads and (b) time to insert mappings and query one GUID for different numbers of entries in the LRC, using 5 concurrent inserting clients and 5 concurrent querying clients.

To measure the effective throughput of the LRC, i.e. the time to complete the insert of a certain number of entries, the total time to insert 500,000 mappings was measured for different numbers of concurrent threads. Figure 3 shows that the time falls rapidly with increasing numbers of threads, bottoming out after 10 or 20 threads. For 20 threads the total time taken is about 40% less than using one thread. Although the time for an individual operation is slower the more concurrent operations are taking place, the overall throughput actually increases, showing the ability of the LRC to handle multiply threaded operations.

Figure 3(b) compares insert time and query time for the LRC with 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.

4.2. Replica Metadata Catalog

The Replica Metadata Catalog can be regarded as an add-on to the RLS system and is used by the Replica Manager to provide a complete view on LFN:GUID:SURL (Figure 1) mapping. In fact the way the RMC and LRC are used is exactly the same, only the data stored is different and thus one would expect similar performance from both components.

In the European DataGrid model, there can be many user defined LFNs to a single GUID. Therefore the query behaviour was analysed in the case where there were multiple LFNs mapped to a single GUID. Figure 4(a) shows the time to insert and delete 10 GUIDs with different numbers of LFNs mapped to them and Figure 4(b) shows the time to query for 1 LFN with varying numbers of LFNs per GUID. These tests used the C++ API.

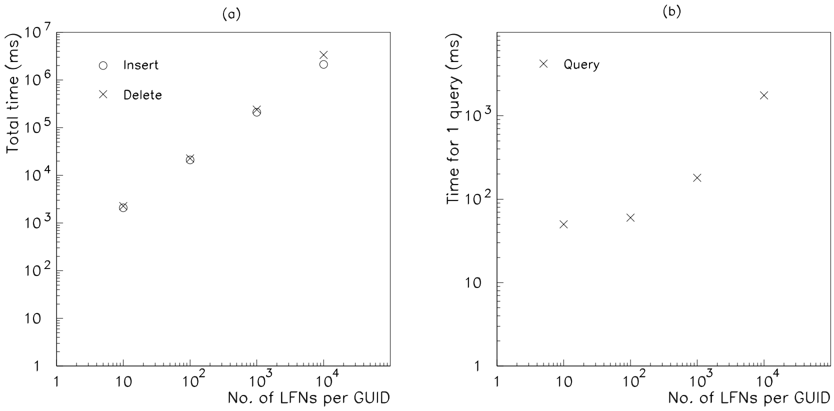

Figure 4. Total time to (a) insert and delete 10 GUIDs with varying number of LFNs, and (b) query for one LFN.

The insert/delete times increase linearly as one might expect, since each new LFN mapping to the GUID is treated similarly to inserting a new mapping, thus the effect is to give similar results to the insert times for the LRC seen in Figure 2 in terms of number of operations performed. Query operations take longer the more LFNs exist for a single GUID, however the query time per LFN mapped to the GUID actually decreases the more mappings there are, hence the RMC performance scales well with the number of mappings.

The command line interface for all the services is implemented in Java using the Java API. Table I shows some timing statistics giving the time to execute different parts of the command addAlias used to insert a GUID:LFN mapping into the RMC.

The total time to execute the command was 3.0s and this time is broken down into the following areas: The start-up script sets various options such as logging parameters and the class-path for the Java executable and this, along with the time to start the Java Virtual

Table I. Timing statistics for adding a GUID:LFN mapping in the RMC using the CLI.

Time (s)	Operation
0 - 1.0	Start-up script and JVM start-up time
1.0 - 1.1	Parse command and options
1.1 - 2.1	Get RMC service locator
2.1 - 2.3	Get RMC object
2.3 - 3.0	Call to the rmc.addMapping() method
3.0	End

Machine, took 1.0s. After parsing the command line it took a further 1.0s to get the LRC service locator - during this time many external classes had to be loaded in.

The call to the addAlias() method within the Java API took around 0.7s, due to the effect of dynamic class loading the first time a method is called. Compared to the average over many calls of around 25 ms observed above in the API tests, this is very large, and because every time the CLI is used a new JVM is started up, the time to execute the command is the same every time.

In short, the time taken to insert a GUID:LFN mapping using the command line interface is about 2 orders of magnitude longer than the average time taken using the Java or C++ API. Therefore the command line tool is only recommended for simple testing and not for large scale operations on the catalog.

5. Open Issues and Future Work

Most of the services provided by Work Package 2 have satisfied the basic user requirements and thus the software system can be used efficiently in the DataGrid environment. However, several areas still need work.

5.1. User Feedback

There are a number of capabilities that have been requested by the users of ou services or that we have described and planned in the overall architecture but did not implement within the project.

There is currently no proper *transaction support* in the Replica Management services. This means that if a seemingly atomic operation is composite, like copying a file and registering it in a catalog, there is no transactional safety mechanism if only half of the operation is successful. This may leave the content of the catalogs inconsistent with respect to the actual files in storage. A *consistency service* scanning the catalog content and checking its validity also would add to the quality of service.

The other extreme is the grouping of several operations into a single transaction. Use cases from the HEP community have shown that the granularity of interaction is not on a single file or even of a collection of files. Instead, they would like to see several operations managed as a single operative entity. These are operations on sets of files, spawned across several jobs, involving operations like replication, registration, unregistration, deletion, etc. This can be managed in a straightforward manner if data management jobs are assigned to a session. The *Session Manager* would hand out session IDs and finalize sessions when they are closed, i.e. only at that time would all changes to the catalogs be visible to all other sessions. In this context sessions are not to be misinterpreted as transactions, as transactions may not span different client processes; sessions are also managed in a much more lazy fashion.

We have also received requests for the support of *file-system semantics* in the logical file namespace, including the support for directories and *fine-grained access control* mechanisms in the catalog layer. The directory support in the logical namespace would satisfy most requests for proper *file collections* if metadata can also be associated with directory entries. For more complex collection semantics there might be the need for additional metadata.

In the original WP2 design, *pre- and post-processing hooks* were foreseen to be available through the Replica Manager. The reason to have such hooks is that many Virtual Organizations have use-cases concerning replication where they have to add some processing before the data can be replicated. These can be checksums, specialized validation after copy, encryption and decryption of data, additional entries to be made in application catalogs, actual data generation from templates, etc.

5.2. Future Services

There are several other services that need to be addressed in future work. In the first prototype of the EU DataGrid testbed WP2 provided a *replica subscription facility*, GDMP. The hope was to replace GDMP with a more robust and versatile facility fully integrated with the rest of the replication system, but this was not done due time pressures. The functionality to automatically distribute files based on some subscription mechanism is still much-needed.

In terms of *Metadata Management*, currently, the metadata support in the RMC is limited to of O(10) basic typed attributes, which can be used to select sets of LFNs. The RMC cannot support many more metadata attributes or more complex metadata structures. There is ongoing work in the context of the GGF DAIS working group to define proper interfaces for data access and integration, much of their findings can be used to refine and re-define the metadata structures of the RMC.

5.3. Future Optimisation Strategies

The current ROS component is conservative in terms of functionality, and there has been much recent research in this area that should be leveraged to improve the capabilities and the performance of the optimisation systems.

5.4. Adhering to Standards

The Open Grid Service Architecture (OGSA) is a GGF effort to describe Grid service architectures which define in generic terms the structure and mechanisms that have to be made available by Grid services in order to be 'OGSA compliant'. The current versions of these Grid standards are extensions of the standard web service frameworks already used for Work Package 2 software, consequently the migraton to any of these newly emerging Grid standards should be straightforward.

6. Related Work

As mentioned, one of the first Grid data management prototypes has been GDMP [18]. In its first toolkit the Globus project provided in its toolkit an LDAP-based replica catalog service and a simple replica manager that could manage the copy and registration as a single step. In the early stages of WP2 these services have wrapped these tools into the EDG Replica Manager's initial implementation, providing a more user-friendly API and mass storage bindings. The most recent overview of Globus's data management services can be found in [1]. Later, we have developed the concept of the Replica Location Service (RLS) together with Globus [10]. Both projects have their own implementation of the RLS.

An integrated approach for data and meta-data management is provided in the Storage Resource Broker (SRB) [5]. Related work with respect to replica access optimization has been done in the Earth Since Grid (ESG) [2] project, which makes use of the Network Weather Service (NWS). Within the High-Energy Physics community one of the most closely related projects is SAM [19] (Sequential data Access via Metadata) that was initially designed to handle data management issues of the D0 experiment at Fermilab. In terms of storage management, we have also participated actively in the definition of the Storage Resource Management (SRM) [7] interface specification. In terms of data management services there is very relevant work being done on a Reliable FTP service [16] in the Globus Alliance, which may be exploited by future high-level data management services for reliable data movement. Another data management system as part of the Condor project is Kangaroo [11], which provides a reliable data movement service. It also makes use of all available replicas in its system such that this is transparent to the application.

7. Conclusion

In this paper we have 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 accessed via a single entry point, the Replica Manager. The adoption of the web services model enables a platform and vendor independent means of accessing and managing the data and associated metadata of the user applications. Performance analysis has shown that when the services are used as intended, they can cope under stressful conditions and scale well with increasing user load.

It remains to be seen what the final standard will be for a Grid services framework. But the data management services we have developed should be adaptable with minimal effort to the emergent standards and can provide a solid base for any future efforts in this area.

References

1. Allcock, B., J. Bester, J. Bresnahan, A. L. Chervenak, I. Foster, C. Kesselman, S. Meder, V. Nefedova, D. Quesnal, and S. Tuecke: 2002, 'Data Management and Transfer in High Performance Computational Grid Environments'. *Parallel Computing Journal* **28**(5), 749–771.
2. Allcock, B., I. Foster, V. Nefedov, A. Chervenak, E. Deelman, C. Kesselman, J. Lee, A. Sim, A. Shoshani, B. Drach, and D. Williams: 2001, 'High-Performance Remote Access to Climate Simulation Data: A Challenge Problem for Data Grid Technologies.'. In: *Supercomputing 2001*. Denver, Texas, USA.
3. Apache Axis. `http://ws.apache.org/axis/`.
4. Apache Tomcat. `http://jakarta.apache.org/tomcat`.

5. Baru, C., R. Moore, A. Rajasekar, and M. Wan: 1998, 'The SDSC Storage Research Broker'. In: *CASCON'98*. Toronto, Canada.
6. Bell, W. H., D. G. Cameron, L. Capozza, A. P. Millar, K. Stockinger, and F. Zini: 2002, 'Design of a Replica Optimisation Framework'. Technical Report DataGrid-02-TED-021215, CERN, Geneva, Switzerland.
7. Bird, I., B. Hess, A. Kowalski, D. Petravick, R.Wellner, J. Gu, E. Otoo, A. Romosan, A. Sim, A. Shoshani, W. Hoschek, P. Kunszt, H. Stockinger, K. Stockinger, B. Tierney, and J.-P. Baud: 2002, 'SRM joint functional design'. In: *Global Grid Forum 4*. Toronto, Canada.
8. Bloom, B.: 1970, 'Space/Time Trade-offs in Hash Coding with Allowable Errors.'. *Communications of ACM* **13**(7), 422–426.
9. Bosio, D., J. Casey, A. Frohner, L. Guy, P. Kunszt, E. Laure, S. Lemaitre, L. Lucio, H. Stockinger, K. Stockinger, W. Bell, D. Cameron, G. McCance, P. Millar, J. Hahkala, N. Karlsson, V. Nenonen, M. Silander, O. Mulmo, G.-L. Volpato, G. Andronico, F. DiCarlo, L. Salconi, A. Domenici, R. Carvajal-Schiaffino, and F. Zini: 2003, 'Next-Generation EU DataGrid Data Management Services'. In: *Computing in High Energy Physics (CHEP 2003)*. La Jolla, California, USA.
10. Chervenak, A., E. Deelman, I. Foster, L. Guy, W. Hoschek, A. Iamnitchi, C. Kesselman, P. Kunszt, M. Ripeanu, B. Schwartzkopf, H. Stockinger, K. Stockinger, and B. Tierney: 2002, 'Giggle: A Framework for Constructing Scalable Replica Location Services'. In: *Proc. of the International IEEE Supercomputing conference (SC 2002)*. Baltimore, USA.
11. D. Thain, J. Basney, S. S. and M. Livny: 2001, 'The Kangaroo Approach to Data Movement on the Grid'. In: *10thIEEE Symposium on High Performance and Distributed Computing (HPDC-10)*. San Francisco, California, USA.
12. Foster, I., J. Frey, S. Graham, S. Tuecke, K. Czajkowski, D. Ferguson, F. Leymann, M. Nally, T. Storey, W. Vambenepe, and S. Weerawarana: 2004, 'Modeling Stateful Resources with Web Services'. In: *GlobusWorld 2004*. San Fransisco, California, USA.
13. Foster, I., C. Kesselman, J. M. Nick, and S. Tuecke: 2002, 'The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration'. Technical report, Global Grid Forum.
14. Kunszt, P., E. Laure, H. Stockinger, and K. Stockinger: 2003, 'Replica Management with Reptor.'. In: *5th International Conference on Parallel Processing and Applied Mathematics*. Czestochowa, Poland.
15. LCG: The LHC Computing Grid. `http://cern.ch/LCG/`.
16. Reliable File Transfer. `http://www-unix.globus.org/toolkit/reliable_transfer.html`.
17. Stockinger, H., F. Donno, E. Laure, S. Muzaffar, P. Kunszt, G. Andronico, and P. Millar: 2003, 'Grid Data Management in Action: Experience in Running and Supporting Data Management Services in the EU DataGrid Project'. In: *Computing in High Energy Physics (CHEP 2003)*. La Jolla, California, USA.
18. Stockinger, H., A. Samar, S. Muzaffar, and F. Donno: 2002, 'Grid Data Mirroring Package (GDMP)'. *Scientific Programming Journal - Special Issue: Grid Computing* **10**(2), 121–134.
19. Terekhov, I., R. Pordes, V.White, L. Lueking, L. Carpenter, H. Schellman, J. Trumbo, S. Veseli, and M. Vranicar: 2001, 'Distributed Data Access and Resource Management in the D0 SAM System'. In: *10thIEEE Symposium on High Performance and Distributed Computing (HPDC-10)*. San Francisco, California, USA.
20. The Jakarta Project. `http://jakarta.apache.org/`.

21. van Engelen, R. A. and K. A. Gallivan: 2002, 'The gSOAP Toolkit for Web Services and Peer-To-Peer Computing Networks'. In: *Proc. of the IEEE CCGrid Conference 2002*. Berlin, Germany.
22. W3C. "Web Services Activity". `http://www.w3c.org/2002/ws/`.
23. Web Service Definition Language. `http://www.w3.org/TR/wsdl`.