

Wide Area Data Replication for Scientific Collaborations

Ann Chervenak*, Robert Schuler, Carl Kesselman

USC Information Sciences Institute, Marina del Rey, California, USA

E-mail: annc, schuler, carl@isi.edu

*Corresponding author

Scott Koranda, Brian Moe

University of Wisconsin, Milwaukee, WI, USA

Email: skoranda, bmoe@gravity.phys.uwm.edu

Abstract: Scientific applications require sophisticated data management capabilities. We present the Data Replication Service (DRS), one of a planned set of higher-level data management services for scientific collaborations in Grid environments. The capabilities of the DRS are based on the publication capability of the Lightweight Data Replicator (LDR) system developed for the LIGO Scientific Collaboration. We describe LIGO publication requirements and LDR functionality. We also describe the design and implementation of the DRS in the Globus Toolkit Version 4.0 environment and present performance results.

Keywords: data management, data replication, publication

Biographical notes: Ann Chervenak is a Research Assistant Professor in the Computer Science Department at the University of Southern California and a Research Team Leader at USC Information Sciences Institute. She received her Ph.D. in Computer Science from the University of California at Berkeley in 1994. Her research focuses on data management in Grid environments.

Robert Schuler is a Programmer Analyst in the Center for Grid Technologies at the USC Information Sciences Institute. His current work involves development of replication management tools for the Globus Toolkit. He received a B.S. in Computer Science from the University of Southern California.

Carl Kesselman is the Director of the Center for Grid Technologies at the Information Sciences Institute of the University of Southern California and a Research Professor of Computer Science at the University of Southern California. He received a Ph.D. in Computer Science from the University of California at Los Angeles. His research interests are in all areas of Grid computing, including architecture, security, data management and applications.

Scott Koranda received his Ph.D. in Physics from the University of Wisconsin-Milwaukee in 1995. From 2001 to 2005, he was a staff scientist in the LIGO group in the Physics Department at the University of Wisconsin-Milwaukee, where he helped integrate grid computing into LIGO data analysis infrastructure. He is currently a Solutions Architect with Univa Corporation in Chicago, IL.

Brian Moe is a Programming Consultant working with the LIGO group in the Physics Department at the University of Wisconsin-Milwaukee. Brian's professional interests include grid computing, software design methodologies, and functional programming languages.

1 INTRODUCTION

Scientific application domains spend considerable effort on managing the large amounts of data produced during experimentation and simulation. Required functionality includes large scale data transfer, validation, replication, and catalog registration operations. To facilitate the demanding data publication and access requirements of scientists, application communities have developed customized, higher-level Grid data management services that are built on top of standard low-level Grid components such as data transport protocols and replica catalogs.

For example, the Laser Interferometer Gravitational Wave Observatory (LIGO) collaboration (Abramovici 1992; Abbott 2004; LIGO Project 2004) replicates data extensively and stores more than 40 million files across ten locations. Experimental data sets are produced at two LIGO instrument sites and replicated at other LIGO sites to provide scientists with local access to data. In addition, scientists analyze the data and publish their results, which may also be replicated. LIGO researchers developed the Lightweight Data Replicator (LDR) System (LDR Project 2004) for data management. LDR is built on top of standard Grid data services such as the Globus Replica Location Service (Chervenak 2002; 2004) and the GridFTP data transport protocol (Allcock 2005). LDR provides a rich set of data management functionality, including a pull-based model for replicating necessary files to a LIGO site; efficient data transfer among LIGO sites; a distributed metadata service architecture; an interface to local storage systems; and a validation component that verifies that files on a storage system are correctly registered in a local RLS catalog.

Another example of a customized, high-level data management system is Don Quijote (Branco 2004), a replica management service developed for the ATLAS (A Toroidal LHC Apparatus) high energy physics experiment (ATLAS Project 2005). Don Quijote is a proxy service that provides management of data replicas across three heterogeneous Grid environments used by ATLAS scientists: the US Grid3, the NorduGrid and the LCG-2 Grid. Each Grid uses different middleware, including different underlying replica catalogs. Don Quijote provides capabilities for replica discovery, creation, registration and renaming after data validation.

Other examples of scientific Grid projects that have developed customized, high-level data management services include high energy physics projects in Europe, such as the LHC Computing Grid (LCG) middleware (LCG Project 2005), the gLite system (EGEE Project 2005) and the DataGrid Reptor system (Kunszt 2003). Many Grid applications, such as the Earth System Grid (Bernholdt 2005), use a web portal to coordinate data publication, discovery and access using Grid middleware.

The functionality of these high-level data management services varies by application domain, but they share several requirements:

- The need to publish and replicate large scientific datasets consisting of thousands or millions of files
- The need to register data replicas in catalog(s) and discover them
- The need to perform metadata-based discovery of desired datasets
- Some applications require the ability to validate the correctness of replicas

In general, updates to datasets and replica consistency services are not required, since most scientific datasets are accessed in a read-only manner.

While the efforts described above have been quite successful in providing production data management services to individual scientific domains, each project has spent considerable effort and resources to design, implement and maintain its data management system. Often, scientists would prefer that their effort be spent on science rather than on infrastructure development. Another disadvantage of these customized data management services is that they typically cannot be re-used by other applications.

Our long-term goal is to generalize much of the functionality provided by these systems and make it application-independent. We plan eventually to provide a set of flexible, composable, general-purpose, higher-level data management services to support Grid applications. These services should build upon existing lower level Grid services and should be configurable by policy to meet the needs of a variety of application domains. We envision that this suite of data management services will provide capabilities such as data replication and validation that can be used individually or in combination. While application communities may still need to provide some domain-specific data management capabilities, our goal is to reduce the amount of effort required by each community to design, implement and maintain services for data management.

In this paper, we describe the design, implementation and performance of one higher-level data management service, the Globus Data Replication Service (DRS). The functionality of the DRS is based on the publication capability of the LIGO Lightweight Data Replicator (LDR) system. The DRS builds on lower-level Grid data services, including the Globus Reliable File Transfer (RFT) service (Globus Alliance 2005) and Replica Location Service (RLS) (Chervenak 2002; 2004). The function of the DRS is to ensure that a specified set of files exists on a storage site by querying the Replica Location Service to discover the locations of desired files, transferring copies of the missing files from selected locations and registering new replicas in the RLS. The DRS is implemented as a Web service that complies with the Web Services Resource Framework (WS-RF) specifications and is available as a technical preview component in the Globus Toolkit Version 4 release.

The contributions of this paper include: 1) a description of the data publication capability provided by the LIGO LDR system; 2) a generalization of this functionality to specify characteristics of an application-independent Data Replication Service (DRS); 3) a description of the design and implementation of DRS in the GT4 environment; and 4) an evaluation of the performance of DRS in a wide area Grid. The paper concludes with a discussion of related work

and our future plans for developing additional data management services.

This paper extends work published in the 6th IEEE/ACM International Workshop on Grid Computing (Grid2005) (Chervenak 2005). It includes additional details on the data publication components of LIGO, more discussion of the DRS implementation and related work, and includes updated performance measurements based on the latest versions of DRS and other software.

2 LIGO AND THE LIGHTWEIGHT DATA REPLICATOR SERVICE

The functionality included in our Data Replication Service is motivated by a careful examination of the data publication capability provided by the LIGO Lightweight Data Replicator System. In this section, we describe LIGO data publication requirements and the LDR publication functionality.

Throughout this paper, we will use the terms logical and physical file name. A *logical file name (LFN)* is a unique identifier for the contents of a file. Typically, a Virtual Organization (for example, a scientific collaboration) defines and manages the logical namespace and guarantees uniqueness of logical names within that organization. A *physical file name (PFN)* is the location of a copy of the file on a storage system. The physical namespace is managed by the file system or storage system. The LIGO environment currently contains more than six million unique logical files and more than 40 million physical files stored at ten sites.

2.1 Data Publishing Requirements in LIGO

The publishing requirements for LIGO have grown over time and are of two types. First, the two LIGO detectors at Livingston, Louisiana, and Hanford, Washington, produce data sets at a rate of slightly less than a terabyte per day during LIGO experimental runs. Each detector produces a file every 16 seconds that contains data for those 16 seconds of measurements. These files range in size from 1 to 100 megabytes. The GEO detector in Germany is also part of the LIGO scientific collaboration and produces data sets. All these data sets are copied to the main data repository at CalTech, which stores data in a tape-based mass storage system. Other sites in the LIGO collaborative can acquire copies of the data sets from CalTech as well as from one another.

Scientists also publish new or *derived* data sets as they perform analysis on existing data sets. For example, data filtering may create tens of thousands of new files. Another example of secondary data is calibration information for the interferometer data. Scientists typically want to publish these new data sets immediately. Scientists at all the LIGO sites participate in this process of analysis and data set publication. The workload for this type of publishing activity tends to be highly variable. Over time, the rate of publication of these derived data sets is growing. While currently approximately 1/3 of the data sets in the LIGO infrastructure are derived data products and 2/3 are raw data

sets from the LIGO detectors, the proportion of derived data sets is rapidly increasing.

Metadata attributes are collected during data publication and are subsequently used during replication of data sets among LIGO sites. The LDR metadata catalog associates attributes with logical files that contain measurements taken from LIGO instruments as well as derived data products. Attributes associated with LIGO files include GPS start and end times for measurements contained in each data file, the site and interferometer where the data measurements were taken, the science run or epoch during which the data were collected, the file or “frame” format, the data quality standard, and the size and MD5 checksums for the file. The role of these attributes in replication is described below.

2.2 LDR Data Publishing

Next, we describe how the LDR system supports LIGO data publishing.

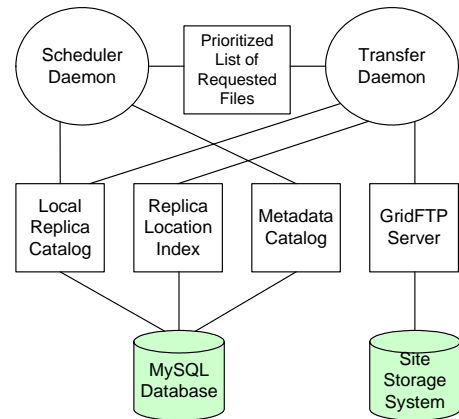


Figure 1: Illustrates the deployment of services and daemons on a typical LIGO site.

Figure 1 illustrates the services and daemons deployed at a typical LIGO site. First, each of the ten LIGO sites includes a local storage system where data replicas are stored. Every LIGO site also includes a GridFTP server that is used for efficient transfer of files among LIGO sites. Each site also deploys a fully replicated Metadata Catalog that maintains a current copy of all associations between logical file names and attributes for the LIGO collaboration. In addition, each site deploys two Replica Location Service (RLS) (Chervenak 2002; 2004) servers: a Local Replica Catalog (LRC) that stores mappings from logical names to physical storage locations and a Replica Location Index (RLI) that collects state summaries from all ten LRCs deployed in the LIGO environment. A query to any RLI identifies all LRCs in the LIGO deployment that contain mappings for a logical file name. Each LIGO site also runs scheduling and transfer daemons that play important roles in publication.

The publication and replication component of the Lightweight Data Replicator System works as follows. Each LDR site runs a scheduling daemon that initiates data transfers to that site using a pull model. The scheduling daemon issues SQL queries to the site’s local metadata catalog to request sets of logical files with specified

metadata attributes, which make up *collections*. An example query for a collection is: “all files from the Hanford site with frame type 'RDS_R_L1' from the LIGO third science run.” Each collection has a priority level that determines the order in which different collections are transferred to the local site. For each file in a collection, the scheduling daemon checks the RLS Local Replica Catalog to determine whether the desired file already exists on the local storage system. If not, the daemon adds that file’s logical name to a priority-based scheduling queue.

An LDR site also runs a transfer daemon that periodically checks this queue of requested files and initiates data transfer operations. For each file on the queue in order from highest to lowest priority, the transfer daemon queries the RLS Replica Location Index server to find locations in the Grid where the file exists and randomly chooses among these locations. Then the transfer daemon initiates data transfer operations from the remote site to the local site using the GridFTP data transport protocol.

The transfer daemon interacts with local storage management logic to store the files correctly and registers the newly-copied files in the Local Replica Catalog. The interaction of the daemon with local storage is via a plug-in module and is usually specific to the particular details of the local storage system. The module must make available five methods that are called at various points during the replication cycle and the post-processing phase: a method that returns the destination URL for a file transfer, which can then be used in the GridFTP transfer operation; another to generate the list of PFNs to be associated with an LFN in the replica catalog; a callback method to modify permissions, user or group information for newly-created files; an optional method to verify the checksum for the transferred file; and system-specific methods to clean up local storage after failed transfers.

The LRC sends periodic summaries about its state, including the existence of newly-replicated files, to Replica Location Index servers at all LIGO locations. Thus, other sites in the LIGO deployment can discover the new replicas by querying their local RLI servers.

If a data transfer operation does not complete successfully, LDR puts the logical file names back on the prioritized list of requested files to be retried later.

3 DESIGN OF THE DATA REPLICATION SERVICE

Next, we present the design of a general data replication service suitable for use by scientific collaborations that require wide area data replication.

3.1 Generalizing the LDR Publication Scheme

Our goal in designing the Data Replication Service was to generalize LDR’s publication functionality to provide a similar capability that is independent of the LIGO infrastructure and useful for a variety of application domains.

Aspects of our design, which draw upon the features supported by the LDR model, include:

- an interface to specify which file(s) are required on the local site;
- use of Globus RLS to discover where replicas exist in the Grid;
- use of a selection algorithm for choosing among available replicas of desired data files;
- use of the Reliable File Transfer Service and GridFTP data transport protocol to copy data from selected remote location(s) to the local site; and
- registration of new files in the Globus RLS.

In contrast to LDR, we chose to make the design of DRS independent of any particular metadata service architecture, such as the fully replicated and distributed service deployed in LIGO. In LDR, the scheduling daemon initiates data transfers by performing a metadata catalog query to identify the list of desired files. By contrast, our service interface simply allows the user to specify a list of logical files that should be copied to a site without prescribing how the list is generated, which allows DRS to be compatible with other application-specific metadata catalogs.

3.2 Data Replication Service Design Overview

The function of the Data Replication Service (DRS) is to ensure that a specified set of files exist on a storage site. The operations of the DRS include *discovery*, identifying where desired data files exist on the Grid; *transfer*, copying the desired data files to the local storage system efficiently; and *registration*, adding location mappings to the Replica Location Service so that other sites may discover newly-created replicas. Throughout DRS replication operations, the service maintains state about each file, including which operations on the file have succeeded or failed.

Principles that guided our fundamental design decisions included:

- Design of a composable architecture based on reusable lower-level services;
- Use of a flexible deployment model to allow for site-specific deployment configurations; and
- Adoption of the Web Services Resource Framework (WS-RF) (Czajkowski 2004) standards to promote interoperability.

Following the WS-RF specification, the general structure of the DRS consists of a WS-Resource deployed in a WS-RF-compliant container. The container takes responsibility for many of the “plumbing” issues related to communication, messaging, logging, and security. The WS-Resource construct allows for stateful resources within a Web services context (Czajkowski 2004; Foster 2004). In the DRS, the stateful resource represents a replication request created by a client, allowing the client to access the state of the replication request and to control its behavior. We call the DRS resource a “Replicator.”

A WS-Resource exposes state information in the form of WS-ResourceProperties. The DRS Replicator resource exposes resource properties pertaining to the replication request, including:

- Status – indicates whether the request is pending, active, suspended, terminated, or destroyed;

- Stage – the current stage in the replication, which may be discover, transfer, or register;
- Result – the final result of the request, which may be none, finished, failed, or exception; and
- Count – a count of total, finished, failed, and terminated files in the replication.

To access a Replicator’s resource properties, clients may use standard WS-RF defined operations to get a resource property, get multiple resource properties, query resource properties, and subscribe to resource properties. Subscriptions enable clients to receive asynchronous notifications when a resource property changes. The Replicator also defines an interface to find the status of specific files in the request by logical filename or by status and allows the client to specify an offset and limit in case the result set is very large.

Lifecycle is another important aspect of the design. Clients begin by passing a create message to the DRS, which creates a Replicator resource. The create message includes a URL of a file that contains the details of the replication request: a listing of the files to be replicated and the destinations for newly created replicas. The URL of the request file may be a file, http, or ftp URL; in the latter cases the DRS loads the request file from the remote location. After creating the Replicator resource, the client uses start, stop, suspend, and resume operations to control the replication activity. Finally, the client destroys the resource using the WS-RF standard destroy or set termination time operations. Once destroyed, the Replicator resource along with its resource properties may not be accessed.

To perform key operations of data replication, DRS depends on two non-web services, GridFTP and RLS, and two other WS-RF services, RFT and Delegation.

We considered design alternatives for DRS. We could have extended the existing RFT implementation to include replica registration capabilities. However, the RFT service is evolving quickly and independently, and it would be challenging to incorporate ongoing RFT changes into the DRS implementation. In the future, we may build DRS and other high-level services based on the Globus GRAM job execution service.

containing an explicit description of the replication request in terms of the desired files, identified by their logical file names, and the desired destination locations, identified by URLs. The client sends a message to the DRS (3) to create the Replicator resource and passes the request file’s URL, which the Replicator then retrieves and reads (4).

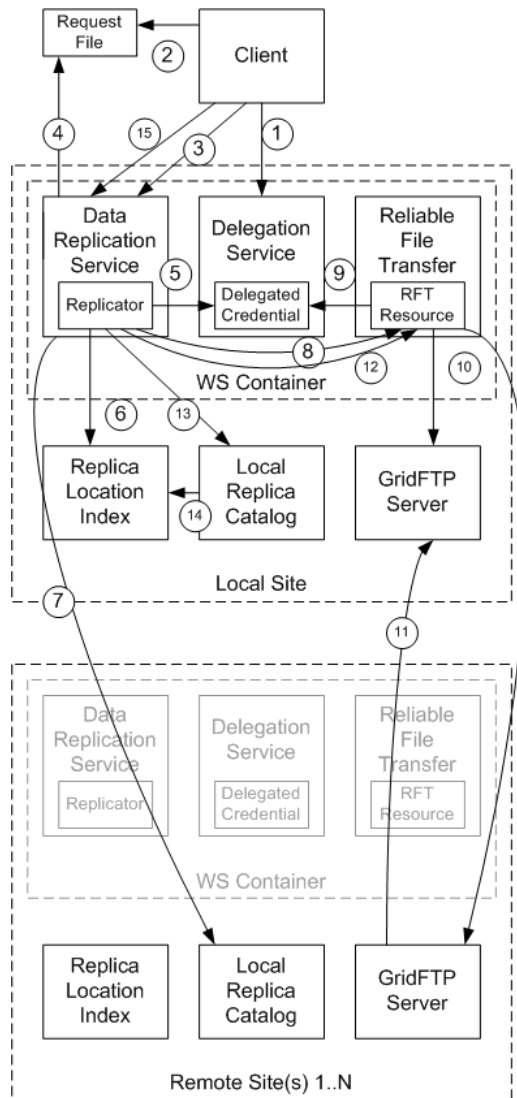


Figure 2: Illustrates deployment and operation of DRS in Web Service container along with RFT service and non-web service RLS and GridFTP.

4 DRS IMPLEMENTATION

Figure 2 depicts the implementation of the Globus Data Replication Service in a typical deployment scenario where a DRS at a local site replicates data from one or more remote sites. The GT4 Delegation service is co-located with the services that use the delegated credentials, so DRS, RFT and Delegation are deployed together. The RLS and GridFTP services may be deployed together with DRS or on separate servers. The intent of the design is to provide a flexible deployment model that allows various configurations.

The client begins by using the Delegation Service to create a delegated credential (1) that may be used by other services to act on behalf of the user according to his or her permissions. The client may create a request file (2)

The Replicator accesses the user’s delegated credential (5) and begins the *discovery* phase of replication by querying (6) the Replica Location Index to find the LRCs that contain mappings for the requested files. The Replicator invokes a catalog filter class (implemented by the user and specific to their site’s requirements) to filter out unwanted catalogs from the rest of the discovery phase. The Replicator then queries (7) each of the remaining remote Local Replica Catalogs to get the physical file names associated with the requested files. The Replicator uses a source selector class (again implemented by the user and specific to their site’s requirements) to select the desired source files for each file to be replicated.

The Replicator then moves on to the *transfer* phase of the replication by using the RFT service to create a RFT resource (8) and start the transfer. At this point, control is passed to RFT, which also retrieves the delegated credential (9). RFT sets up the file transfers (10), while GridFTP servers perform the data transfer between sites (11). Once the file transfers are complete, the Replicator checks the status of the RFT resource and gets the status (12) of each file in the transfer request to ensure that each file transferred successfully.

The Replicator then moves to the *register* phase of the replication by adding mappings (13) for the newly created replicas to its Local Replica Catalog. In time, the Local Replica Catalog updates (14) its Replica Location Index along with RLIs located at remote sites to make the new replicas visible throughout the Grid environment.

Finally, the client may check the status of its Replicator resource (15) and query the status of each of the files in its replication request.

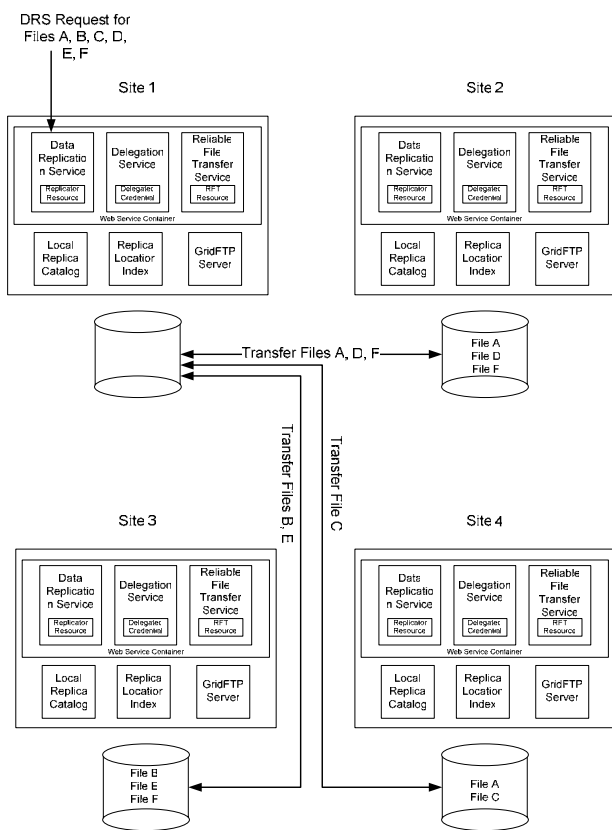


Figure 3: A multi-site DRS operation

Figure 3 shows an overview of a multi-site DRS operation, where the client submits a DRS request at site 1 to make replicas at that site of five files, labeled A through F. The DRS checks its Replica Location Index to determine where the replicas of the file exist, and finds that one or more copies of the desired files exist at three other sites. DRS selects among available replicas and transfers three files from site 2, two files from site 3, and one file from site 4. After the transfers are complete, these new files are registered in the Local Replica Catalog at site 1.

The present implementation of the DRS is based on the Java version of the Globus Toolkit Web services container and utilizes standard database interfaces. Based on this infrastructure, the DRS inherits the capabilities associated with WS-RF resources. These include standard interfaces to query resource properties synchronously and subscribe to resource property changes via asynchronous notification (Graham 2004). Also, the DRS uses the WS Authentication & Authorization APIs (Globus Alliance 2004) to communicate securely with other services in the deployment, to properly authenticate users, and to ensure that users perform operations permitted to them by administrators of the system.

5 DATA REPLICATION SERVICE PERFORMANCE

In this section, we present wide area performance measurements for our DRS implementation. We describe DRS performance for two types of requests: replication of a small number of relatively large files and for a large number of small files. These performance measurements update those published in the 6th IEEE/ACM International Workshop on Grid Computing (Chervenak 2005) using the latest versions of Globus Toolkit Version 4 software. In some cases, the measurements reported here are slower than in the previous paper, reflecting more contention on the network between Los Angeles and Chicago that affects both data transfer times and remote catalog discovery operations.

The local site, which is the destination for pull-based data transfers, is located in Los Angeles. It is a dual-processor, 1.1 GHz Pentium III workstation with 1.5 GBytes of memory and a 1 Gbit Ethernet connection. This machine runs a GT4 container and deploys services including RFT and DRS as well as GridFTP and RLS (which are not Web services). When a client on this machine makes a DRS request, the DRS Replicator Resource is created and started in the container. DRS queries the local RLI and then queries remote LRCs to find the physical locations of source files. Next, DRS initiates file transfers by invoking the RFT service, which creates an RFT Resource. The RFT service coordinates data transfers from the remote site to local storage using GridFTP.

The remote site where desired data files are stored is a machine at Argonne National Laboratory in Illinois. This machine is a dual-processor, 3 GHz Intel Xeon workstation with 2 gigabytes of memory with 1.1 terabytes of disk storage. This remote machine runs a GT4 container as well as GridFTP and RLS services. The RLS Local Replica Catalog at the remote site is queried by the DRS in Los Angeles during the discovery of file locations. During RFT operations, the GridFTP server on the remote machine acts as the source for pull-based data transfers.

For all our tests, we report the following measurements.

- *Create Replicator* is the time for the DRS to create a persistent Replicator resource, including storing resource state in a database.
- *Start* is the time required for the DRS to 'start' the Replicator resource.

- *Discover* is the time to perform lookups in the local RLI and remote LRC(s) for all logical files that will be replicated in this request.
- *Create RFT* is the time required to create the RFT Resource associated with the DRS request.
- *Transfer* is the total time taken by the RFT Service to transfer all files in the request.
- *Register* is the time required to register the LFN-PFN mappings for new replicas in the LRC.

For a transfer of a single 1 gigabyte-file, we use four parallel TCP streams for the GridFTP transfer and a TCP buffer size of 16000 bytes. These parameter values are based on guidance in the GridFTP documentation (Globus Alliance 2005), and transfer performance can be tuned by varying these values. We ran this transfer five times and obtained the average performance shown in Table 1. The transfer time for the RFT operation corresponds to a data rate during the transfer portion of the request of approximately 11.7 Mb/s. The table also shows standard deviation for each measured value. We note that the variance is high on operations such as *create replicator*, which varies from 223 to 724 milliseconds during the five trials, and *create RFT*, which varies from 718 to 1614 milliseconds. The *register* operation accesses a database on another machine in the local area network, and its high standard deviation is due to a lengthy registration operation for one experiment. There is also substantial variance in the *transfer* times, which is likely due to variations in the wide area transfer bandwidth between Los Angeles and Chicago.

Table 1: Performance for DRS replicating one file of size 1 gigabyte

Component of Operation	Time (milliseconds)	Standard Deviation
<i>Create Replicator</i>	403.60	203.95
<i>Start</i>	19.80	39.25
<i>Discover</i>	347.40	41.20
<i>Create RFT</i>	2030.60	2441.93
<i>Transfer</i>	699110.40	58000.49
<i>Register</i>	566.40	783.77

Table 2: Performance for DRS replicating ten files of size 1 gigabyte

Component of Operation	Time (milliseconds)	Standard Deviation
<i>Create Replicator</i>	415.40	256.89
<i>Start</i>	7.20	11.08
<i>Discover</i>	413.60	215.68
<i>Create RFT</i>	1155.20	346.78
<i>Transfer</i>	1042988.40	17026.96
<i>Register</i>	307.40	181.38

Next, we ran a test that copies ten files of size 1 gigabyte to the local site. Again, we ran the tests five times and report average measurements. We set the concurrency to ten for RFT transfers (i.e., ten RFT worker threads performing these transfers). Individual GridFTP transfers use parallelism of four TCP streams and a TCP buffer size of 16000 bytes. The performance is shown in Table 2.

The time to create the Replicator and RFT resources is similar to the previous example. The data transfer rate for this experiment was approximately 78.5 Mb/s. This reflects the efficiency of transferring large files using the GridFTP protocol.

Finally, we performed tests that transfer large numbers of smaller files. We initiate transfers of 1000 files of size 10 megabytes and of 10000 files of size 1 megabyte. For these experiments, we set the concurrency to ten for RFT transfers. Individual GridFTP transfers use parallelism of four TCP streams and a TCP buffer size of 16000 bytes. The performance numbers in Table 3 and Table 4 show values averaged over five experiments.

As might be expected, the times to create the Replicator and especially the RFT resources are substantially longer in these cases than for the previous tests, since the services have to save state for 1000 and 10000 outstanding transfer operations, respectively. The average transfer rates achieved are 43.3 Mb/s for the 10 megabyte file size and 36.4 Mb/s for the 1 megabyte file size, which reflect the lower efficiency of transferring relatively small files.

Table 3: Performance for transfer of 1000 files of size 10 megabytes

Component of Operation	Time (msec) File size: 10 MB	Standard Deviation
<i>Create Replicator</i>	381.60	127.67
<i>Start</i>	12.60	13.63
<i>Discover</i>	1260.60	374.97
<i>Create RFT</i>	28662.40	2589.40
<i>Transfer</i>	1847194.40	78331.41
<i>Register</i>	6210.00	756.31

Table 4: Performance for transfer of 10000 files of size 1 megabytes

Component of Operation	Time (msec) File size: 10 MB	Standard Deviation
<i>Create Replicator</i>	602.60	363.47
<i>Start</i>	8.60	12.54
<i>Discover</i>	5454.00	1294.82
<i>Create RFT</i>	147505.20	13037.91
<i>Transfer</i>	2192842.80	12865.47
<i>Register</i>	28347.80	2639.13

These results provide insights into the performance of DRS operations for small numbers of large files and for larger numbers of smaller files. In future testing, we plan to increase the scale of the number of files per request and the size of the files transferred.

6 RELATED WORK

As already discussed, we envision developing a suite of general-purpose, configurable and composable higher-level data management services, of which the Data Replication Service is the first. Our goal is to let application communities deploy those services that provide desirable

functionality. By contrast, several other Grid systems take a different architectural approach and provide higher-level data management capabilities using highly integrated functionality, including replica registration, metadata management, data discovery and maintenance of consistency among replicas. These systems include the Storage Resource Broker (Baru 1998; Rajasekar 2003) and Grid DataFarm (Tatebe 2003) projects.

In the introduction section, we discussed several application communities that have developed customized data management services, including the Don Quijote system for the ATLAS physics application (Branco 2004), several high energy physics projects in Europe, including the LHC Computing Grid (LCG) middleware (LCG Project 2005) and the gLite system (EGEE Project 2005), and a number of applications that coordinate Grid middleware using portals, such as the Earth System Grid (Bernholdt 2005) and the Geosciences Network (GEON) (GEON Project 2004). All these systems include customized functionality. By contrast, the goal of the DRS is to provide an application-independent data replication capability that could be used in a variety of application domains.

Several data transfer tools offer functionality similar to that provided by the Globus Reliable File Transfer (RFT) Service (Globus Alliance 2005). The EGEE File Transfer Service (FTS) (EGEE Middleware Activity JRA1 2006) provides asynchronous, reliable transfers of one or more files using GridFTP data transport protocol or the Storage Resource Manager (Shoshani 2002). State related to ongoing transfers is maintained in a database to allow restart or retries of failed operations. Like RFT, FTS operations specify the physical names of source and destination files, and the service does not interact with replica catalogs.

The Storage Resource Manager (SRM) (Shoshani 2002) from Lawrence Berkeley Laboratory provides efficient management of storage systems and coordinates large data transfers. SRM features include space management for storage systems, including hierarchical mass storage systems; reservation of space for file transfers; lifetime management for storage allocations; management of multiple-file transfer requests; and invocation of data transport protocols such as GridFTP. SRM does not provide integrated replica management functionality.

The LIGO project and its participation in the Grid Physics Network (GriPhyN) project are described in the following references (Abramovici 1992; Deelman 2002; Abbott 2004; LIGO Project 2004). Marka et al. (Marka 2002) describe the Network Data Analysis Server, an early data distribution system for gravitational wave data that provides some functionality similar to LDR.

7 FUTURE WORK

We will do more extensive performance testing of the Data Replication Service, and its functionality will evolve as we gain additional experience with the service.

As already described, we plan to design and implement a suite of generally-useful and configurable high-level data management services for Grids. The experience gained with DRS in the GT4 release and earlier experience with LDR in the LIGO environment will drive many of our design decisions for new data management services. For example, we plan to develop services that imitate the data validation capability currently being provided by LDR. The goal of the validation is to keep storage servers and catalogs synchronized. LDR scripts periodically verify that files registered in Local Replica Catalogs at each site actually exist on the storage system. Less frequently, these scripts also calculate checksums for stored files and verify that these checksums match the expected checksums for files registered in LRCs. The eventual goal for a local validation service is to populate LRCs automatically to reflect the current contents of a storage system. We also plan to develop wide-area validation services that verify the correctness of replicas registered in the RLS.

8 CONCLUSIONS

We have described the needs of applications for sophisticated data management services as well our goal of providing general, configurable, composable, higher-level data management services for Grids. We presented the design and implementation of the Data Replication Service, whose functionality is based on the publication capabilities of the successful LDR system. We also presented wide area performance results for this service in the Globus Toolkit Version 4 environment.

One goal for higher-level data management services like DRS is that eventually they will replace the customized functionality of the LIGO Lightweight Data Replicator system. LIGO science runs began in 2002 and are expected to run until at least September 2007. Throughout that time, LDR will continue to be a production resource that provides data management capabilities to LIGO scientists. Concurrently, LIGO researchers will work with Grid data service designers to enhance the functionality of the Data Replication Service and to design future data management services.

ACKNOWLEDGEMENT

We are grateful to Mats Rynge for his help in setting up the DRS service in the GT4.0 testbed; to Ravi Madduri and Bill Allcock for help with RFT testing and configuration and for providing access to resources at ANL; and to Mike Link and Shishir Bharathi for their assistance with security issues during our testing.

This research was supported in part by DOE Cooperative Agreements DE-FC02-01ER25449 & DE-FC02-01ER25453. Scott Koranda and Brian Moe are supported in part by NSF grant 0326281.

REFERENCES

- Abbott, B., et al., (LIGO Scientific Collaboration) (2004). 'Detector Description and Performance for the First Coincidence Observations Between LIGO and GEO.' *Nucl. Instrum. Meth.*, Vol. A517, pp. 154-179.
- Abramovici, A., W. Althouse, et al. (1992) 'LIGO: The Laser Interferometer Gravitational-Wave Observatory', *Science*, Vol. 256, pp. 325-333.
- Allcock, W., J. Bresnahan, R. Kettimuthu, M. Link, C. Dumitrescu, I. Raicu, I. Foster (2005) 'The Globus Striped GridFTP Framework and Server', *Proc. of SC05 Conference*, Seattle, WA, November.
- Baru, C., R. Moore, et al. (1998) 'The SDSC Storage Resource Broker', *Proc. of CASCON'98 Conference*.
- Bernholdt, D., S. Bharathi, D. Brown, K. Chancio, M. Chen, A. Chervenak, L. Cinquini, B. Drach, I. Foster, P. Fox, J. Garcia, C. Kesselman, R. Markel, D. Middleton, V. Nefedova, L. Pouchard, A. Shoshani, A. Sim, G. Strand, D. Williams (2005) 'The Earth System Grid: Supporting the Next Generation of Climate Modeling Research', *Proc. of the IEEE*, Vol. 93, No. 3, pp. 485-495.
- Branco, M. (2004) 'Don Quijote - Data Management for the ATLAS Automatic Production System', *Proc. of Computing in High Energy and Nuclear Physics (CHEP) 2004*, Interlaken, Switzerland.
- Chervenak, A., E. Deelman, I. Foster, L. Guy, W. Hoschek, A. Iamnitchi, C. Kesselman, P. Kunst, M. Ripeanu, B. Schwartzkopf, H. Stockinger, K. Stockinger, B. Tierney (2002) 'Giggle: A Framework for Constructing Scalable Replica Location Services', *Proc. of SC2002 Conference*, Baltimore, MD, November.
- Chervenak, A., R. Schuler, C. Kesselman, S. Koranda, B. Moe (2005) 'Wide Area Data Replication for Scientific Collaborations', *Proc. of 6th IEEE/ACM Int'l Workshop on Grid Computing (Grid2005)*, Seattle, WA, November.
- Chervenak, A. L., N. Palavalli, S. Bharathi, C. Kesselman, R. Schwartzkopf (2004) 'Performance and Scalability of a Replica Location Service', *Proc. of Thirteenth IEEE Int'l Symposium High Performance Distributed Computing (HPDC-13)*, Honolulu, HI, June.
- Deelman, E., et al. (2002) 'GriPhyN and LIGO, Building a Virtual Data Grid for Gravitational Wave Scientists', *Proc. of 11th Intl. Symposium on High Performance Distributed Computing (HPDC-11)*, Edinburgh, Scotland, June.
- Kunzt, P., Erwin Laure, Heinz Stockinger, Kurt Stockinger (2003) 'Advanced Replica Management with Reptor', *Proc. of 5th International Conference on Parallel Processing and Applied Mathematics*, Czestochowa, Poland.
- Marka, S., Benoit Mours, Roy Williams (2002) 'Network Data Analysis Server (NDAS) Prototype Development', *Classical and Quantum Gravity*, Vol. 19, No. 7, pp. 1537-1540.
- Rajasekar, A., et al. (2003) 'Storage Resource Broker - Managing Distributed Data in a Grid', *Computer Society of India Journal (Special Issue on SAN)*.
- Shoshani, A., Alex Sim, Junmin Gu (2002) 'Storage Resource Managers: Middleware Components for Grid Storage', *Proc. of Nineteenth IEEE Symposium on Mass Storage Systems (MSS '02)*.
- Tatebe, O., et al. (2003) 'Worldwide Fast File Replication on Grid Datafarm', *Proc. of 2003 Computing in High Energy and Nuclear Physics (CHEP03)*.

WEBSITES

- ATLAS Project (2005) ATLAS: A Toroidal LHC Apparatus, <http://atlas.web.cern.ch/Atlas/>.
- Czajkowski, K., et al. (2004) The WS-Resource Framework Version 1.0, <http://www-106.ibm.com/developerworks/library/ws-resource/ws-wsrf.pdf>.
- EGEE Middleware Activity JRA1 (2006) DJRA1.4 - EGEE Middleware Architecture and Planning (Release 2), <https://edms.cern.ch/document/594698/1.0/>, EGEE Project. 2006.
- EGEE Project (2005) gLite Lightweight Middleware for Grid Computing, <http://glite.web.cern.ch/glite/>.
- Foster, I., et al. (2004) Modeling Stateful Resources with Web Services, <http://www-106.ibm.com/developerworks/library/ws-resource/ws-modelingresources.html>
- GEON Project (2004) GEON: The Geosciences Network, www.geongrid.org.
- Globus Alliance (2004) GT4.0 Pre-WS Authentication & Authorization Developer Guide, <http://www-unix.globus.org/toolkit/docs/development/4.0-drafts/security/prewsaa/developer/>.
- Globus Alliance (2005) globus-url-copy Documentation, <http://www.globus.org/toolkit/docs/4.0/data/gridftp/rn01re01.html>.
- Globus Alliance (2005) Reliable File Transfer Service, <http://www.globus.org/toolkit/docs/4.0/data/rft/>.
- Graham, S., et al. (2004) Web Services Resource Properties (WS-ResourceProperties) Version 1.1, <http://www-106.ibm.com/developerworks/library/ws-resource/ws-resourceproperties.pdf>.
- LCG Project (2005) LHC Computing Grid: Distributed Production Environment for Physics Data Processing, <http://lcg.web.cern.ch/LCG/>.
- LDR Project (2004) Lightweight Data Replicator, <http://www.lsc-group.phys.uwm.edu/LDR/>.
- LIGO Project (2004) LIGO - Laser Interferometer Gravitational Wave Observatory, <http://www.ligo.caltech.edu/>.