Wide Area Data Replication for Scientific Collaborations

Ann Chervenak, Robert Schuler,
Carl Kesselman
USC Information Sciences Institute
(annc, carl, schuler@isi.edu)

Scott Koranda, Brian Moe
University of Wisconsin Milwaukee
(skoranda, bmoe@gravity.phys.uwm.edu)

Abstract

Scientific applications require sophisticated data management capabilities. We present the design and implementation of a Data Replication Service (DRS), one of a planned set of higher-level data management services for Grids. 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.

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 [1, 2, 21] 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 [20] for data management. LDR is built on top of standard Grid data services such as the Globus Replica Location Service [8, 9] and the GridFTP data transport protocol [3]. 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 [7], a replica management service developed for the ATLAS (A Toroidal LHC ApparatuS) high energy physics experiment [4]. 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 the LHC Computing Grid (LCG) middleware [19], the gLite system [12] and the DataGrid Reptor system [18]. Many Grid applications, such as the Earth System Grid [6], 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 [16] and Replica Location Service (RLS) [8, 9]. The function of the DRS is to ensure that a specified set of files exists on a storage site by comparing the contents of an RLS Local Replica Catalog with a list of desired files, transferring copies of the missing files from other locations and registering them in the replica catalog. 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.

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 and Hanford 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.

2.2 LDR Data Publishing

In this section, we describe how the LDR system supports LIGO data publishing.

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. Each site also includes a GridFTP server that is used for efficient transfer of files among LIGO sites. Each site also deploys a Metadata Catalog that contains associations between logical file names and metadata attributes. In addition, each site deploys two Replica Location Service (RLS) [8, 9] 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.

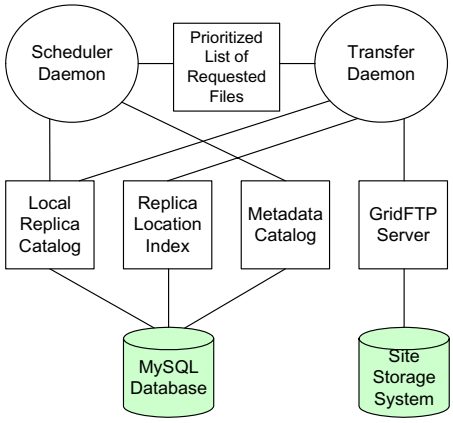

Figure 1: Illustrates the deployment of services and daemons on a typical LIGO site.

The publishing component of the Lightweight Data Replicator System works as follows. Each LDR site runs a scheduling daemon that initiates local data transfers using a pull model. The scheduling daemon queries the site's local metadata catalog to request sets of files with specified metadata attributes. These sets of files are called *collections,* and each collection has a priority level that determines the order in which files from different collections will be 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 list of requested files and initiates data transfer operations. For each file on the list 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 the available locations. Then the transfer daemon initiates data transfer operations from the remote site to the local site using the GridFTP data transport protocol. The daemon interacts with the local storage management logic to store the files correctly and registers the newly-copied files in the Local Replica Catalog.

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 higher-level 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 and whether they exist at the local site;
- 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) [10] 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 [10, 13]. 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-WS 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.

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.

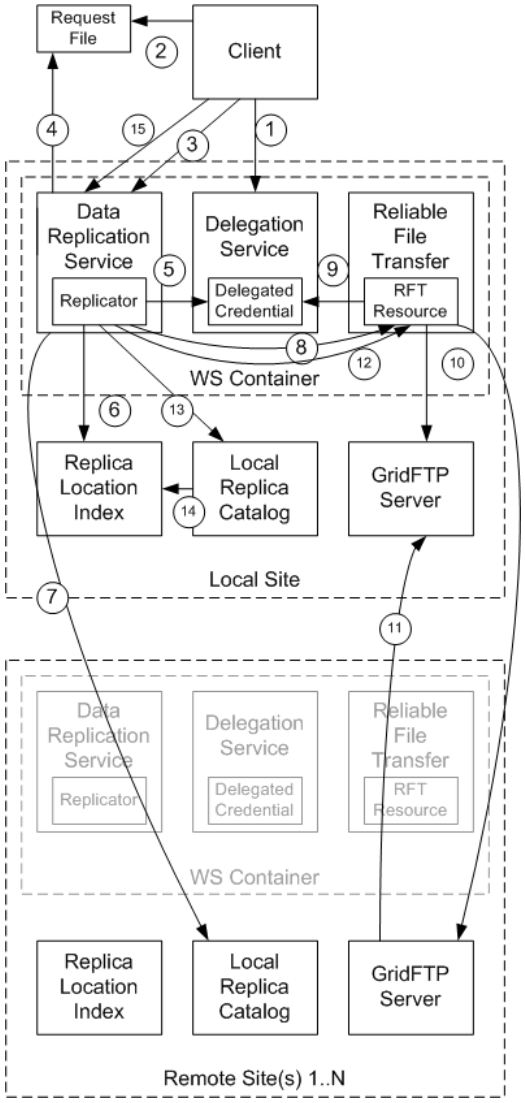

Figure 2: Illustrates deployment and operation of DRS in Web Service container along with RFT service and non-web service RLS and GridFTP.

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) 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).

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, 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, 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.

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 [17]. Also, the DRS uses the WS Authentication & Authorization APIs [15] 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.

The local site, which is the destination for the pull-based 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 this container. DRS queries the local RLI and then queries the remote LRC 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 transfer operations, the GridFTP server on this machine acts as the source for pull-based transfers to the local site.

For all our tests, we report the following measurements.

- *Create Replicator* is the time for the DRS service to create a persistent Replicator resource, including storing resource state in a database.
- *Start* is the time required for the DRS service 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 1,000,000. These parameter values are based on guidance in the GridFTP documentation [14]. We ran this transfer five times and obtained the average performance shown in Table 1.

This corresponds to a data rate during the transfer portion of the request of approximately 49 Mbits/sec. The table also shows standard deviation for each measured value. We note that the variance is high on operations such as *discover*, which varies from 307 to 5371 milliseconds during the five trials, and *register*, which varies from 295 to 4305 milliseconds. The *discover* operation accesses the remote LRC over the wide area, which may account for some of the variability in these numbers, while *register* accesses a database on another machine in the local area network. 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
Create Replicator	310.8	163.6
Start	11.4	20.5
Discover	1355.2	2245.4
Create RFT	780.8	250.12
Transfer	163207.8	88744.2
Register	2584.4	2111.7

Next, we run 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 two for RFT transfers (i.e., two RFT worker threads performing these transfers). Individual GridFTP transfers use parallelism of four TCP streams and a TCP buffer size of 1,000,000. The performance numbers are shown in Table 2.

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

Component of Operation	Time (milliseconds)	Standard Deviation
Create Replicator	317	156.2
Start	12.4	19.0
Discover	449.0	184.0
Create RFT	808.6	256.9
Transfer	1186796.0	31596.9
Register	3720.8	2121.3

The time to create the Replicator and RFT resources is similar to the previous example. The average time to discover replicas using the RLS is less than in the previous example and shows less variance. The data transfer rate for this experiment was approximately 67.4 Mbits/sec. The average RLS registration time was

slightly higher than in the previous experiment and exhibited large variance.

Finally, we performed two tests that transfer large numbers of smaller files. We initiate a transfer of 1000 files of size 1 megabyte and of size 10 megabytes. 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 100,000. The performance numbers in Table 3 and Table 4 show values averaged over five experiments.

Table 3: DRS performance for transfer of 1000 files of size 1 megabyte

Component of Operation	Time (msec) File size: 1 MB	Standard Deviation
Create Replicator	324.2	172.3
Start	10.0	17.3
Discover	2226.6	2288.0
Create RFT	2540.0	548.3
Transfer	606296.0	1107.1
Register	5537.4	464.2

Table 4: DRS performance for transfer of 1000 files of size 10 megabytes

Component of Operation	Time (msec) File size: 10 MB	Standard Deviation
Create Replicator	1561.0	1643.1
Start	9.8	17.4
Discover	1286.6	196.5
Create RFT	3300.2	1090.1
Transfer	963456.0	30163.9
Register	11278.2	6602.9

As might be expected, the times to create the Replicator and especially the RFT resources are substantially longer in these cases than for the pervious tests, since the services have to save state for 1000 outstanding transfer operations. The RLS discover and register operations continue to show high variance. The average transfer rate achieved is 13.2 Mbits/sec for the 1 megabyte file size, which reflects the lower efficiency of transferring relatively small files using RFT. For the 10 megabyte file size, the transfer rate achieved is on average 83 Mbits/sec.

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

The LIGO project and its participation in the Grid Physics Network (GriPhyN) project are described in the following references [1, 2, 11, 21]. Marka et al. [22] describe the Network Data Analysis Server, an early data distribution system for gravitational wave data that provides some functionality similar to LDR.

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 [5, 23] and Grid DataFarm [25] projects.

The Storage Resource Manager project [24] and the DataMover client from Lawrence Berkeley Laboratory provide efficient management of large numbers of data transfers. However, these components do not provide integrated replica management functionality.

7. Future Work

We plan to do more extensive performance testing of the Data Replication Service and to scale up the size of the files being transferred and the number of files per DRS request. We will gain additional experience as others use the service as part of the Globus Toolkit 4.0.

As already described, we plan to design and implement a suite of generally-useful and configurable high-level data management services for Grid environments. The experience gained with the DRS in the GT4 release and earlier experience with LDR in the LIGO environment will drive many of our design decisions for these new data management services.

For example, we plan to develop services that imitate the data validation capability currently being implemented for 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 of the

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.

The 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. The goal of LIGO researchers is that eventually these services will replace the LDR system.

9. Acknowledgements

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.

10. References

1. Abbott, B., et al., Detector Description and Performance for the First Coincidence Observations Between LIGO and GEO. Nucl. Instrum. Meth., A517. 154-179.

2. Abramovici, A., W. Althouse, et al. LIGO: The Laser Interferometer Gravitational-Wave Observatory. Science, 256. 325-333.

3. Allcock, W., et al., The Globus Striped GridFTP Framework and Server. in SC05 Conference, 2005.

4. ATLAS Project. ATLAS: A Toroidal LHC ApparatuS, http://atlas.web.cern.ch/Atlas/, 2005.

5. Baru, C., R. Moore, et al., The SDSC Storage Resource Broker. in CASCON'98 Conference, (1998).

6. Bernholdt, D., et al., The Earth System Grid: Supporting the Next Generation of Climate Modeling Research. Proceedings of the IEEE, 93 (3). 485- 495.

7. Branco, M., Don Quijote - Data Management for the ATLAS Automatic Production System. in Computing in High Energy and Nuclear Physics (CHEP) 2004.

8. Chervenak, A., et al., Giggle: A Framework for Constructing Sclable Replica Location Services. in SC2002 Conference, (Baltimore, MD, 2002).

9. Chervenak, A.L., et al., Performance and Scalability of a Replica Location Service. in Thirteenth IEEE Int'l Symposium High Performance Distributed Computing (HPDC-13), (Honolulu, HI, 2004).

10. Czajkowski, K., et al. The WS-Resource Framework Version 1.0, 2004.

11. Deelman, E., et al., GriPhyN and LIGO, Building a Virtual Data Grid for Gravitational Wave Scientists. in 11th Intl. Symposium on High Performance Distributed Computing (HPDC-11), (Edinburgh, Scotland, 2002).

12. EGEE Project. gLite Lightweight Middleware for Grid Computing, http://glite.web.cern.ch/glite/, 2005.

13. Foster, I., et al. Modeling Stateful Resources with Web Services version 1.0, 2004.

14. Globus Alliance. globus-url-copy Command Documentation, http://www.globus.org/toolkit/docs/4.0/data-/gridftp/rn01re01.html, 2005.

15. Globus Alliance. Authentication & Authorization: Developer's Guide, http://www-unix.globus.org/toolkit/docs-/development/4.0-drafts/security/prewsaa/developer/, 2004.

16. Globus Alliance. Reliable File Transfer Service, http://www.globus.org/toolkit/docs/4.0/data/rft/, 2005.

17. Graham, S., et al. Web Services Resource Properties (WS-ResourceProperties) Version 1.1, 2004.

18. Kunszt, P., et al., Advanced Replica Management with Reptor. in 5th International Conference on Parallel Processing and Applied Mathematics, 2003.

19. LCG Project. LHC Computing Grid: Distributed Production Environment for Physics Data Processing, http://lcg.web.cern.ch/LCG/, 2005.

20. LIGO Project. Lightweight Data Replicator, http://www.lsc-group.phys.uwm.edu/LDR/, 2004.

21. LIGO Project. Laser Interferometer Gravitational Wave Observatory, http://www.ligo.caltech.edu/, 2004.

22. Marka, S., et al., Network Data Analysis Server (NDAS) Prototype Development. Classical and Quantum Gravity, 19 (7). 1537-1540.

23. Rajasekar, A., et al. Storage Resource Broker - Managing Distributed Data in a Grid. Computer Society of India Journal (Special Issue on SAN).

24. Shoshani, A., et al., Storage Resource Managers: Middleware Components for Grid Storage. in Nineteenth IEEE Symp. on Mass Storage Systems (MSS '02), (2002).

25. Tatebe, O., et al., Worldwide Fast File Replication on Grid Datafarm. in Computing in High Energy and Nuclear Physics (CHEP03), (2003).