

# Large-Scale Data Replication for LIGO

LEE LIMING

The Laser Interferometer Gravitational Wave Observatory (LIGO) is a national research facility whose objective is the detection of gravitational waves. Gravitational waves are tiny distortions of space and time caused when very large masses, such as stars, move suddenly. They are predicted by Einstein's theory of general relativity, but much work must still be done to observe them and match observations with theoretical predictions.

In practice, detecting and measuring gravitational waves involve incredible amounts of data and computation. The scientists that work with the LIGO facility are distributed across many organizations and physical locations, so the system for storing and processing the data is also distributed. One of the many challenges for the LIGO activity is the need to replicate large amounts of data so that it is readily available to scientists in several parts of the world. This month's column describes this data challenge and the solution currently in use by LIGO, which relies heavily on Grid technology.

## The Challenge

LIGO consists of two detector facilities—one in Livingston, Louisiana, and one in Richland, Washington—operated jointly by the California Institute of Technology (Caltech) and the Massachusetts Institute of Technology (MIT). The detectors at these two sites

are used together and in cooperation with detectors in other countries to look for coincident variations in space-time that may be indications of gravitational waves.

The LIGO facility includes three interferometers at the two sites, each of which records thousands of channels at several sampling rates. Approximately one terabyte (1 TB = 1,024 GB = 1,048,576 MB) of data is collected every day during a detection run. During the commissioning phase of the LIGO interferometers, these runs have typically lasted 2–8 weeks. LIGO expects to begin a one-year detection run late in 2005. Because the LIGO detector sites are remote, the data was originally stored on tapes and shipped to a data center at Caltech, where it was made available online to scientists. More recently, upgraded network connections have allowed replication to other data centers directly from the detector sites.

The data generated by a LIGO run must be scientifically analyzed for it to be of any value. The analysis is computationally intensive (some classes of astrophysical searches can require hundreds of teraflops), and the data volume itself is quite large. Nine sites within the LIGO collaboration (each operated independently) currently provide computing facilities based on commodity cluster computing. The scientists who have the expertise to

perform this analysis are spread across 41 institutions on several continents, and this community is growing all the time. The key task for LIGO is to get the data from the LIGO detectors to the sites where analysis happens and to make those sites accessible to the participating scientists.

The data management challenge faced by LIGO is therefore to replicate up to 1 TB/day of data to multiple sites securely, efficiently, robustly, and automatically; to keep track of where replicas have been made for each piece of the data; and to use the data in a multitude of independent analysis runs. The nine sites each use mass storage systems, but different systems are used at different sites. Scientists and analysts need a coherent mechanism to learn which data items are currently available, where they are, and how to access them. More specific requirements include the following.

- When high-bandwidth links (10+ Gb/s) are available, they should be utilized efficiently: there should not be unused bandwidth while data transfers are taking place.
- All network links should be used efficiently: there should be no idle time on the network between transfers.
- Scientists should be able to locate data and understand data items using application-level terms (also known as “metadata”).

- Scientists should be able to locate replicas (copies) of any data item using database queries.
- Data transfer endpoints should be authenticated using “strong” security, and data transfers should preserve data integrity.

### The Solution

To meet this challenge, the LIGO Scientific Collaboration (a cooperation between physicists, computer scientists, and information technology experts) developed the Lightweight Data Replicator (LDR), an integrated solution that combines several basic Grid components with other tools to provide an end-to-end system for managing LIGO’s data. LDR’s features make it very useful for replicating data sets to multiple sites within a joint project.

- LDR is intended to be the simplest, easiest-to-maintain software that meets the LIGO requirements. It is based on existing open source Grid components, adding control logic to orchestrate the replication tasks.
- LDR uses network links efficiently because it uses parallel data streams, tunable TCP windows, and tunable write/read buffers. LDR also manages continuous data transfers between sites, ensuring that the network is always being used when there are transfers that need to happen.
- LDR tracks where copies of specific files can be found, given that it may have been replicated at several sites. This information can be queried by user applications so that local

data is used when it is available. Each site maintains a catalog of the data it contains, which can be queried directly. The catalogs update each other so that so that one can query any member site’s catalog to locate a particular file.

- LDR also stores descriptive information (metadata) in a database. Hence, site administrators can select groups of files for replication based on descriptive fields rather than having to specify the name of every file.
- LDR uses the Grid Security Infrastructure (GSI) to authenticate clients and services.

LDR combines several existing Grid components—GSI, GridFTP, Globus Replica Location Service, a metadata catalog service, and PyGlobus—with customized Python daemon code to provide a solution that meets the requirements above.

### Constructing the Lightweight Data Replicator

A Data Grid is a set of geographically distributed sites, institutions, or groups that have a common need to store and access specific data items. In order to provide reasonable performance to users, the Data Grid may need to replicate data items at several sites. LDR’s purpose is to make it easy for Data Grid administrators to manage replicas, the replication process, and access mechanisms within a Data Grid.

Each site in an LDR-based Data Grid takes on a combination of three different roles: publisher,

provider, and subscriber. A site acting as a publisher provides information about available files and where they are located (via URLs), as well as information about the files themselves (metadata such as the creation time, size, and contents). A site acting as a provider provides files that can be accessed by users and replicated at other sites. A site acting as a subscriber maintains replicas of data obtained from a provider. A Data Grid must have at least one site acting as a publisher, at least one acting as a provider, and at least one acting as a subscriber. Again, an individual site can serve as any combination of publisher, provider, or subscriber.

LDR uses a metadata database to store information about what files exist and information about the files such as size or creation time. Metadata is replicated across sites so that all sites in the Data Grid are aware of all available data. Using metadata queries, Data Grid administrators define collections of files to be replicated.

LDR assumes a very general model for “storage.” The details of any particular storage system can be coded and made available to LDR via a very simple API. The “storage system” must be able to accept a file for ingestion and return a URL for the ingested file. This URL is then published in the local LRC. LDR does not need to know the details of how the file is ingested or how the URL is generated.

LDR uses the Globus Replica Location Service (RLS) to store information about what files are located where. RLS is itself a distributed

system: each site runs a Local Replica Catalog (LRC) to store information about the data maintained at the local site and a Replica Location Index (RLI) that aggregates information from all of the sites. Hence, each site has both a local and a global view of the data.

LDR uses a simple priority queue for scheduling data transfers. The local RLI is used to identify a site that has a source file, and the LRC at that site is then queried to obtain the URL of the file. LDR regularly refreshes the queue based on need lists and the current state of the LRCs and RLIs.

LDR is designed to maximize the quantity of data replicated at the expense of the reliability of any single file being transferred. Put another way, LDR tries to replicate as much data as it can without regard to the success of any single file transfer. LDR is designed for bulk replication of data, as opposed to replicating smaller sets of files for just-in-time computing.

LDR uses the GridFTP protocol to transfer data between sites. LDR attempts to transfer as many files as possible between two sites and can simultaneously transfer data between multiple sites at the same time. Failed transfers are not instantly retried. Instead, LDR moves on to the next transfer. Files that do not transfer successfully are simply rescheduled until the transfer completes successfully.

The components described above are all reusable “middleware” available in open source form for use in many systems and applications. The following components, which form the coordination subsystem for LDR, are specific to LDR. They are written in Python and are implemented as server “daemons” that run at LDR sites. Different daemons will run at different sites; the daemons that run at a particular site are determined by the roles that site plays in the Data Grid (publisher, provider,

subscriber).

An LDRMaster daemon is responsible for launching other necessary LDR daemons and watching over them. The rest of the daemons are as follows.

- LDRMetadataServer, a GSI-SOAP server that makes metadata published at a site available to other LDR instances.
- LDRMetadataUpdate, a GSI-SOAP client that updates a LDR instance with new metadata as it is published at a remote LDR site.
- LDRSchedule, which schedules (queues) transfers.
- LDRTransfer, which spawns agents for each source site to replicate or transfer files from source locations.
- LDRdataFindServer, which allows clients to query the metadata and replica location catalogs to discover data or files

Each daemon is independent of the others, and each can be stopped and restarted without needing to coordinate with other daemons.

## Results

LDR was developed by Scott Koranda, Brian Moe, and Kevin Flasch at the University of Wisconsin-Milwaukee in cooperation with the NSF-funded GriPhyN and iVDGL projects. Many members of the LIGO Scientific Collaboration have contributed to deploying and testing LDR.

The LIGO detectors went online and began producing data in 2002. By April 2005, the LIGO team had used LDR to replicate well over 50 terabytes of data to sites including Caltech, MIT, Penn State University, and the

## Resources

The Lightweight Data Replicator (LDR) combines Globus Toolkit components with Python services written specifically for LDR. If you intend to install and use LDR, we recommend that you use the “integrated installation” method mentioned below. (It will install the 3.2 versions of Globus Toolkit components.)

### Globus Toolkit 4.0

Includes the GSI, GridFTP, RLS, MCS, and PyGlobus components

- <http://www.globus.org/toolkit/>

### LDR

All of the LDR components (including those listed separately above) can be installed together as an “integrated installation” using PACMAN caches maintained by the LDR development team.

- <http://www.lsc-group.phys.uwm.edu/LDR/>

University of Wisconsin  
Milwaukee, along with sites  
in Europe including the  
Albert Einstein Institute in  
Golm, Germany, Cardiff  
University, and the  
University of Birmingham.

*Thanks to Scott Koranda for  
reviewing and correcting this  
month's column. Globus is a  
registered trademark held by  
the University of Chicago.  
This work was supported in  
part by the Mathematical,  
Information, and  
Computational Sciences  
Division subprogram of the  
Office of Advanced Scientific  
Computing Research, Office  
of Science, U.S. Department  
of Energy, under Contract W-  
31-109-ENG-38 and under  
Contract DE-AC03-76SF0098  
with the University of  
California; by the National  
Science Foundation; and by  
IBM.*

*Lee Liming is manager of the  
Distributed Systems  
Laboratory (DSL) at Argonne  
National Laboratory and the  
University of Chicago.*