The Globus® Toolkit Replica Location Service

ANN L. CHERVENAK

The Replica Location Service (RLS) is a tool that provides the ability to keep track of one or more copies, or replicas, of files in a Grid environment. This tool, which is included in the Globus Toolkit, is especially helpful for users or applications that need to find where existing files are located in the Grid.

RLS Overview

The Replica Location Service is one component of data management services for Grid environments. It is a simple registry that keeps track of where replicas exist on physical storage systems. Users or services register files in the RLS when the files are created. Later, users query RLS servers to find these replicas.

The RLS is a *distributed* registry, meaning that it may consist of multiple servers at different sites. By distributing the RLS registry, we are able to increase the overall scale of the system and store more mappings than would be possible in a single, centralized catalog. We also avoid creating a single point of failure in the Grid data management system. If desired, the RLS can also be deployed as a single, centralized server.

Some Terminology

Before explaining the RLS in detail, we need to define a few terms. A *logical file name* is a unique identifier for the contents of a file. A *physical file name* is the location of a copy of the file on a storage system. The job of the RLS is to maintain associations, or *mappings*, between logical file names and one or more physical file names of replicas. A user can provide a logical file name to an RLS server and ask for all the registered physical file names of replicas. The user can also query an RLS server to find the logical file name associated with a particular physical file location. In addition, the RLS allows users to associate *attributes* or descriptive information (such as size or checksum) with logical or physical file names that are registered in the catalog. Users can also query the RLS based on these attributes.

Using the RLS: An Example

One example of a system that uses the RLS as part of its data management infrastructure is the Laser Interferometer Gravitational Wave Observatory (LIGO) project. LIGO scientists have instruments at two sites that are designed to detect the existence of gravitational waves. During a run of scientific experiments, each LIGO instrument site produces millions of data files. Scientists at eight other sites want to copy these large data sets to their local storage systems so that they can run scientific analysis on the data. Therefore, each LIGO data file may be replicated at up to ten physical locations in the Grid. LIGO deploys RLS servers at each site to register local mappings and to collect information about mappings at other LIGO sites. To find a copy of a data file, a scientist requests the file from LIGO's data management system, called the Lightweight Data Replicator (LDR). LDR queries the Replica Location Service to find out whether there is a local copy of the file; if not, the RLS tells the data management system where the file exists in the Grid. Then the LDR system generates a request to copy the file to the local storage system and registers the new copy in the local RLS server.

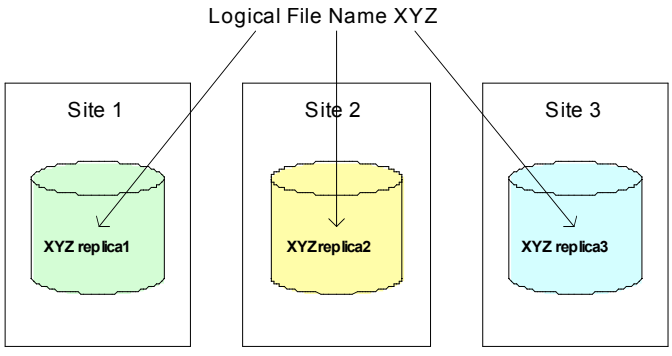

FIGURE 1 Associations between a logical file name and three replicas on different storage sites

LIGO currently uses the Replica Location Service in its production data management environment. The system registers mappings between more than 3 million logical file names and 30 million physical file locations.

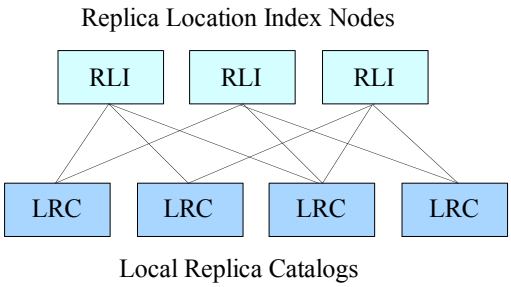

FIGURE 2 Sample deployment of a Replica Location Service

Components of the Replica Location Service

The RLS design consists of two types of servers: the Local Replica Catalog and the Replica Location Index.

The *Local Replica Catalog (LRC)* stores mappings between logical names for data items and the physical locations of replicas of those items. Clients query the LRC to discover replicas associated with a logical name. The simplest RLS deployment consists of a single LRC that acts as a registry of mappings for one or more storage systems. Typically, when an RLS is deployed on a site, an administrator populates it to reflect the contents of a local file or storage system. If new data files are produced by a workflow manager or a data publishing service, these services typically register newly created files with the RLS as part of their publication process. Our performance studies for an LRC deployed on a moderately powerful workstation with a MySQL relational database back end show that the catalog can sustain rates of approximately 600 updates and 2,000 queries per second.

For a distributed RLS deployment, we also provide a higher-level *Replica Location Index (RLI)* server. Each RLI server collects information about the logical name mappings stored in one or more LRCs. An RLI also answers queries about those mappings. When a client wants to discover replicas that may exist at multiple locations, the client will pose that query to an RLI server rather than to an individual Local Replica Catalog. In response to a query, the RLI will return a list of all the LRCs it is aware of that contain mappings for the logical name contained in the query. The client then queries these LRCs to find the physical locations of replicas.

Information is sent from the LRCs to the RLIs using *soft-state update protocols*. Each LRC periodically sends information about its logical name mappings to a set of RLIs. The RLIs collect this mapping information and respond to queries regarding the mappings. Information in RLIs times out and gets periodically refreshed by subsequent updates. An advantage of using such soft-state update protocols is that if an RLI fails and later resumes operation, its contents can be reconstructed using these updates.

Because each LRC may hold millions of logical file name mappings, updates from LRCs to RLIs can become large. Sending them around the network may be slow, especially in the wide area; when updates arrive at an RLI, they may consume considerable storage space there. One option for making these updates more efficient is to compress their contents. Various compression strategies are available. The one that we chose to implement for the RLS is based on Bloom filter compression. Each Local Replica Catalog periodically creates a bit map that summarizes its contents by applying a series of hash functions to each logical name registered in the LRC and setting the corresponding bits in the bit map.

Configuration Options for RLS Deployment

We can determine the performance and reliability of a distributed RLS by the number of Local Replica Catalog and Replica Location Index servers we deploy and the way we configure updates among LRCs and RLIs. For example, we can improve reliability by configuring the system so that every LRC updates multiple RLI indexes.

In practice, current deployments of RLS systems are relatively small. For

example, the LIGO physics collaboration deploys RLS servers at ten sites. For deployments of this scale, the Replica Location Service is often deployed in a fully connected configuration, with a Local Replica Catalog and a Replica Location Index server deployed at each site, and all LRCs sending updates to all RLIs. Such a configuration has the advantage that every site has a complete picture of the replicas in the distributed system.

For larger deployments, however, this configuration is unlikely to scale well because it requires a large number of updates to be sent among servers and stored at each RLI. For example, in a system with hundreds of RLS sites, a fully connected deployment would require every site to send and receive hundreds of updates. In such large deployments, it is likely that the system would be partitioned so that each RLI would receive updates from only a subset of the LRCs.

Ideally, the management of a distributed RLS should be automated and self-configuring, so that LRCs and RLIs discover one another and so that updates among them are redistributed automatically to balance the load when servers join or leave the system. While we are studying approaches for automated membership management, including peer-to-peer self-organization of RLS servers, the current RLS implementation uses a simple static configuration of LRC and RLI servers. An administrator must manually change the pattern of updates among servers.

The RLS and Replica Consistency Services

RLS is a fairly simple tool that provides registration and discovery of files. It is intended to be just one part of an overall system for managing data in Grids. Those considering whether to use an RLS should understand what functionality the system does and does not provide.

One of the most common assumptions new users make about RLS is that, when a new replica is registered in the RLS, the system checks to make sure that the registered entry is a valid replica of an existing file. Actually, the RLS does not perform such correctness or consistency checks on new entries. The RLS allows users to register mappings between logical names and physical locations without any verification that the physical files that are registered as replicas are actually copies of one another. Likewise, if registered replicas are modified so that they are no longer valid copies of one another, the RLS will not detect these changes or take action. Instead, the RLS relies on higher-level data management or consistency services to perform these functions.

There are several reasons for this design choice. One is the simplicity and efficiency of providing a registry that is not required to perform consistency checks on the registered files, which can be time-consuming. For example, a service that guarantees consistency might calculate checksums for all replicas and verify that they match. Providing consistency guarantees therefore creates additional overheads for the system that can limit its performance.

In practice, providing a high level of consistency checking during replica registration is not required for many applications because often only a small set of highly trusted users is allowed to publish data. These privileged users do not need to verify that registered files are actually replicas because the users control the entire publishing process. They typically need to publish a large number of files quickly and cannot tolerate extra overheads associated with performing consistency operations such as checksum calculations on every file registration.

Another reason we do not enforce replica consistency in the RLS is that users may have different definitions of what constitutes a “valid” replica. For some users, replicas are exact byte-for-byte copies of one another. For others, two files may be considered replicas if they are different versions of the same file or if the files contain the same content but in different formats, for example, compressed and uncompressed versions of the same data. We choose not to prescribe a particular definition for replicas in the RLS.

For these reasons, we leave consistency-checking operations to higher-level services, which may perform these checks before or after replicas are registered in the RLS. If these services find that registered replicas are invalid, they will remove the replica mappings from the RLS.

Resources

The Replica Location Service

- www.globus.org/rls/

Performance and Scalability of a Replica Location Service. Ann L. Chervenak, Naveen Palavalli, Shishir Bharathi, Carl Kesselman, and Robert Schwartzkopf. *High Performance Distributed Computing (HPDC-13) Conference*, June 2004.

Web Services and the RLS

The version of the RLS included in the current Globus Toolkit release is not implemented as a Web service. The reason is that our users running data-intensive applications require the efficiency of the current implementation. Because of overheads in current Web service implementations, a Web service RLS implementation could not provide the high registration and query rates these applications require.

We do plan to implement a Web service version of the RLS in 2005. This implementation will be based on a Web Services Resource Framework (WS-RF) standard for the RLS being developed through the Global Grid Forum.

In addition, we are developing a higher-level WS-RF Data Replication Service that uses the Globus Reliable File Transfer Service to copy files and then registers the new replicas in the RLS. This service will be available in the Globus Toolkit Version 4.0 release as a Technical Preview component.

Globus Toolkit is a registered trademark held by the University of Chicago. This research was supported in part by U.S. Department of Energy's Scientific Discovery Through Advanced Computing (SciDAC) program under Cooperative Agreements DE-FC02-01ER25449 and DE-FC02-01ER25453. This work was also 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; by the NASA Information Power Grid program; and by IBM.

Ann Chervenak is a research assistant professor at the University of Southern California and a research team leader at the USC Information Sciences Institute. She leads research and development projects on data management issues, including replica management, metadata services, and the integration of peer-to-peer and Grid systems.