Sharing LCG files across different platforms

Cheng Yaodong, Wang Lu, Liu Aigui, Cheng Gang
Institute of High Energy Physics, Beijing 100049, China
Yaodong.cheng@ihep.ac.cn

Abstract: Currently more and more heterogeneous resources are integrated into LCG. Sharing LCG files across different platforms, including different OS and grid middleware, is a basic issue. We implemented web service interface for LFC and pseudo LCG file access client by using globus Java CoG Kit. This paper describes the architecture, implementation, and performance testing, tuning and usage scenario.

1. Introduction

Over the last few years, the grid has developed rapidly in the research community and many business fields. LCG (LHC Computing Grid) led by CERN is one of largest grid project in the world, whose goal is to integrate large geographically distributed computing fabrics into one virtual computing environment for the LHC. Up to now, more than 240 sites in 50 counties are involved. Currently, it mainly runs over Linux based operation system and gLite based grid middleware. In fact, more and more heterogeneous resources are being integrated into LCG. For example, many national grid infrastructures, such as CNGrid in China which now has about 2,000 CPU, own a lot of computing and storage resources. These resources are potentially a part of LHC data storage and analysis infrastructure. Moreover, windows batch systems, such as the compute cluster server (CCS) are gradually being adopted by some sites. It is not feasible to install LCG data management tool in all of these platforms, so how to share LCG files across different platforms is a big challenge. It is mainly because:

- The main applications on LCG are data-intensive, which require and produce a large amount of data. It is not possible to transfer these data only through Resource Broker;
- LCG data transfer system is based on Globus. However, some other grid middles are not compatible with Globus, even without GSI security mechanism and GridFTP transfer tool. It is not possible to access LCG files directly from arbitrary grid platform;
- LCG File Catalogue (LFC) is a central service. LFC client tools are available only on Linux OS for the moment. If LFC server can not be accessible from other platforms, data can not be located, and transfer of LCG files is also impossible;
- LCG users would like to use the same executable shell script regardless of the platform on which the job is running. For example, “lcg-cp” is a popular LCG tool to copy a Grid file to a local destination. Usually, it is called before a LCG job runs, and if the job is scheduled to a platform without the command, it can not be executed successfully.

For above reasons, we can conclude, LCG data management system is necessary for data-intensive applications and LCG DM should be transparent on multiple platforms. In order to meet the requirements, a special across-platform tool which acts as LCG DM commands is very useful. Firstly,
we implemented web service interface for LFC, thus LCG files can be located on a variety of platforms because web services [1] provide a standard means of interoperating between platforms. In order to access LCG SE, a serial of pseudo LCG file access commands, such as lcg-cp, lcg-cr and so on, are implemented by using LFC WS interface and Java CoG Kit [2]. The Java CoG Kit jGlobus module provides the basic APIs to the Grid to allow access to gridFTP servers, the classic GRAM services, and a complete implementation of GSI. It also includes the myProxy client libraries. Meanwhile, programs developed by JAVA language can run on many platforms. So we choose Java CoG Kit jGlobus to write our GFISH client.

In the rest of this paper, we describe the details of this tool, including architecture, implementation, performance test and tuning, an example of use scenario. At last, we will give a conclusion.

2. Architecture and Implementation

The tool aims to share LCG files across different platforms, we call it GFISH (Grid File Sharing system), which mainly includes two components, a WS-based server and pseudo LCG DM client commands. The architecture is shown in figure 1.

In figure 1, the left side is the server based on SLC OS and LCG DM (GFAL, LCG_util and so on), whereas the right side is the client and pseudo LCG DM commands which can run on a variety of platforms. In fact, the client is only a fat jar package, and it can be downloaded easily to where LCG DM commands are needed. The client interacts with the server to access LCG files (shown as the lower line between the GFISH box and the pseudo LCG DM box in figure 1), or transfers files from/to LCG SE directly after the client gets the location information from GFISH server.

2.1. Server implementation

In HEP computing environment, users need to access not only grid files, but also local files, for example files in CASTOR, when they run jobs on local cluster computing system. That is, users are not always submitting their jobs to GRID. However, maybe job scripts have to be modified according to target computing system. In order to access files transparently, we implemented GFISH, which invokes different protocol and interface according to the prefix of file name, for example, “/castor”, “/grid”, “/dpm”, or “/afs”. Through using GFISH, user can submit the same job script to grid or local cluster without any modification.

GFISH includes some commands and APIs. If theses APIs are used in GFISH WS, GFISH WS server can access local and grid files transparently. Grid files, we define here, are those registered in LFC server and stored in grid storage element. Thus, GFISH WS is an interface for LFC and some other file systems. We use gSOAP Web services development toolkit [3] to implement GFISH WS.
server. For the moment, the web service interfaces of a serial file system calls, such as access, chmod, chown, stat, lstat, mkdir, rmdir, unlink, opendir, readdir, closedir, listreplicas, open, read, write, close and so on, have been implemented.

2.2. Client implementation

Client is designed for users to access remote files, including grid files registered in LFC, across a variety of platforms. We implement the client on basis of JAVA virtual machine. Axis [4] tool is used to generate the JAVA client of GFISH web service interface described in file “gfishws.wsdl”. Thus, GFISH Java client can connect the server easily, though the server is implemented in gSoap C/C++. Secondly, we use CoG jglobus tool to initiate grid environment and tune performance (described in later section). Based on the GFISH client, some pseudo LCG DM commands, such as lcg-cr, lcg-cp, lcg-rf, lcg-uf, lcg-rep, lcg-gt and so on, are implemented. These pseudo commands are possible shell scripts on Linux, or bat files on Window. The client is packaged into a fat jar file, through which a user can access grid files like in LCG environment after he/she is authenticated.

2.3. Security

GFISH WS provides two methods to guarantee the security, one is session-based, and another is GSI-based. The two methods are shown in figure 2.

![Figure 2: security in GFISH WS](image)

Figure 2-A illustrates the session-based authentication method, the steps are:

1. Client tells server the local user and password, myproxy information (server, user, and password).
2. Server authenticates the user and generates a session ID with corresponding information stored in server and transfers the ID to client. Meanwhile, server also gets user delegation from myproxy server. Then client saves the ID in local storage.
3. During the session, Client calls services with given ID. Server get corresponding information, such as local user and grid proxy, then use the information to connect third-part server, eg, LFC and CASTOR.
4. Client and server destroy session when user logout explicitly or expired.

Figure 2-B illustrates the GSI-based authentication method, the steps are:

1. Client and server do mutual authentication by through client user proxy and server certificate.
2. Server checks if the client is in the access list. If client is allowed, a new proxy of the client is created.
3. Server uses new created proxy to connect third-party server in behalf of the client.

In term of session-based method, user is only required to login once, and the following operations can be executed without spending much time on authentication. Thus, better performance can be achieved but maybe security vulnerability exists. GSI-based method has good security, but requires authentication on each operation.
3. Performance test and tuning

SOAP provides good interoperability between different platforms, but it has low performance on transferring a large amount of data [5] because (1) encoding and decoding of SOAP message need much time and (2) the encoded message is 4~8 larger than original. According to the reasons, we perform some performance tuning of GFISH WS gSoap server as follows:

- **Data encoding**: XML is based on ASCII coding mechanism, which has much redundancy information, and it can not be integrated with binary data easily. Base64Binary mechanism has good ability of encoding the binary data, and redundant information is also less. However, it is inefficient because coding and decoding will cause considerable processing overhead. DIME transmits binary data via SOAP attachment to support dynamic data processing, saving the overhead of coding and Serialization. GFISH uses XML for parameters encoding, base64Binary for a small amount of data, DIME for large amounts of data to get better scalability and overall performance.

- **Data compress**: Compression technology can reduce the volume of data transmission, but also have a lot of overhead. In cases of low-bandwidth, it helps to improve performance.

- **KEEP-ALIVE**: HTTP connection would produce a certain delay. Permanent link will potentially improve the performance, especially for high latency and low bandwidth network, as well as the frequent exchange of small capacity of SOAP messages.

- **CHUNK**: During HTTP data transmission, the size of data must be pre-determined in order to calculate the length of message. This requires to cache data firstly, which will cause low efficiency. CHUNK model which is a block-mode HTTP data transmission doesn’t need pre-determine data length. So, it can greatly improve efficiency.

- **Buffer length**: There are two types of cache including SOAP, and I/O. One cache with the length of SOAP_BUFLEN in gSOAP is used to encapsulate SOAP message. A reasonable record size is also important for GFISH read/write operations. Appropriate value of SOAP_BUFLEN and record length vary in different network circumstance.

In fact, the performance in GFISH WS includes two aspects, one is accessing metadata server, for example, LFC server, and another is transferring data file. The two aspects are both measured. The backend database of LFC is MySQL, GFISH WS server and LFC are running on the same machine at IHEP (two dual-core Xeon CPU 3.0 /4 GB memories). The machine of testing “mkdir” and “rmdir” is executed on lxplus.cern.ch which has about 250ms delay to IHEP. Data transfer testing is performed in local area network. The results are shown in figure 3.
The operations “mkdir” and “rmdir” are selected to represent performance of metadata in left side of figure 3. The delay of GSI-based method is very close to that of original operation lfc-*, while session-based is very fast. lfc-* commands, including lfc-mkdir, lfc-rmdir, lfc-ls, and so on, interact with LFC server to operate metadata in file catalogue. GSOAP tuning (DIME, Chunk, keeplive combination) is very useful in LAN environment (right side of figure 3), about 42.3% increase than non-tuning base64Binary method, but it doesn’t help over high-latency network (it is even lower than “scp”, and result is not listed here). To improve the performance of transferring large file, gridftp API in CoG jglobus tool is used in GFISH client, which can perform transfer from LCG SE directly in multiple-stream mode and achieve high performance. The performance of GridFTP is reviewed by many other papers. The paper [6] summarizes some recent developments in Globus GridFTP.

4. An use scenario

Here we take an example of interoperability between grid middleware. For the moment, the typical method is to introduce a gateway between two grid middleware as shown in figure 4. When a LCG user submits a job though UI, the job may be scheduled into a virtual CE (that is gateway), then runs at other grid platform (we call it “unknown grid”, and suppose it is not compatible with globus). The job has to get or put files through gateway because “unknown-grid” can’t access LCG SE directly. In this case, our GFISH tool will be very useful. As shown in Figure 4, GFISH server acts as file information “gateway”. The use scenario is as follows (dashed lines describe the case after GFISH is introduced into grid interoperability):

1) Before job script runs, it checks if LCG DM commands exist (here we take “lcg-cp” as an example). If “lcg-cp” doesn’t exist, the script download GFISH client jar package from GFISH web site. If it exists, go to step 3.
2) Retrieves the credential from the myproxy-server that was previously stored using myproxy-init.
3) Execute LCG DM commands, such as “lcg-cp” to get file from LCG SE directly. These commands are possible from GFISH client, not really from LCG.

5. Conclusion

More and more heterogeneous resources integrated into LCG bring great challenge to data transferring across different platforms. This work aims to let user can access LCG files transparently on a variety of platforms, mainly building a pseudo LCG DM environment, including GSI, LFC,
GFAL, lcg_util, gridftp and so on. The advantage of this tool is that the size of the across-platform LCG DM module is very small, about 6MB. Thus, the tool can be downloaded on demand.

However, it is not always the case. Suppose the executable in a user job script is compiled and linked with GFAL or Globus libraries dynamically. Then the job is scheduled into another grid which is not compatible with Globus. When the executable in the job is running, it will fail to load GFAL or Globus libraries because the target machine doesn’t provide them. That is, GFISH tool only simulates LCG DM commands, and doesn’t provide complete LCG run-time environment. So, the user application should be independent of grid middleware. In fact, it is a basic requirement of grid interoperability. In summary, GFISH is a useful tool for LCG files sharing across different platforms.

![Diagram of use scenario](image)

**Figure 4: An example of use scenario**

**References**

[1] Web Service website: http://www.w3.org/2002/ws/

[2] CoG jglobus, http://dev.globus.org/wiki/CoG_jglobus

[3] gSOAP, The gSOAP Project, http://gsoap2.sourceforge.net/

[4] Axis: http://ws.apache.org/axis/

[5] R. van Engelen. Pushing the SOAP envelope with Web services for scientific computing. In proceedings of the International Conference on Web Services (ICWS), pages 346–352, Las Vegas, 2003.

[6] Globus GridFTP: What's New in 2007 (Invited Paper). John Bresnahan, Michael Link, Gaurav Khanna, Zulfikar Imani, Rajkumar Kettimuthu and Ian Foster. Proceedings of the First International Conference on Networks for Grid Applications (GridNets 2007), Oct, 2007.