Sharing LCG files across different platforms

Cheng Yaodong, Wang Lu, Liu Aigui, Chen 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, performance test and tuning and use 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 the largest grid projects 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 are involved. Currently, it mainly runs over Linux based operation system and gLite based grid middleware. In fact, more and more heterogeneous resources are 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 system, the compute cluster server (CCS) is gradually adopted by some sites. It isn't 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 or 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 middle wares 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), which is a centric service, is accessible only on Linux OS for the moment. If it is not accessible, data can't 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 usually called before a LCG job runs, and if the job is scheduled to a platform without the command, it can't 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 globus Java CoG Kit [2].

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.

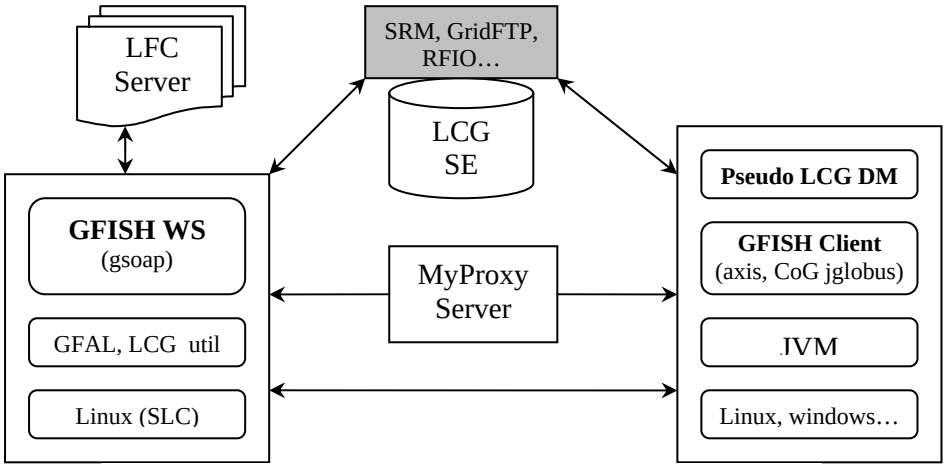

Figure 1: The architecture of GFISH

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.

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, 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 to 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 to 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 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.

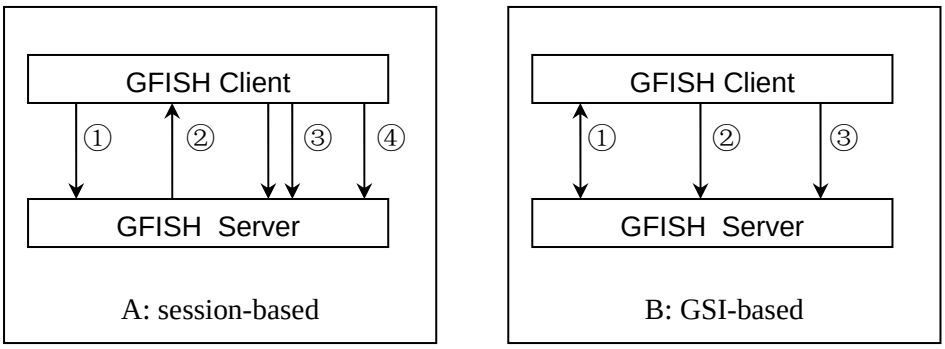

Figure 2: security in GFSIH WS

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, the 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 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 for RPC arguments, base64Binary for small data, DIME for large data;
- Data compress: Compress data before transferring it;

- KEEP-ALIVE: It helps reduce time for HTTP connection;
- CHUNK: transferring data by block without need to compute the length the data;
- Buffer length: The micro SOAP_BUFFLEN of gSoap and the record size of GFISH read/write affects the performance, and they should be set appropriate value.

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 memory). 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 show in figure 3.

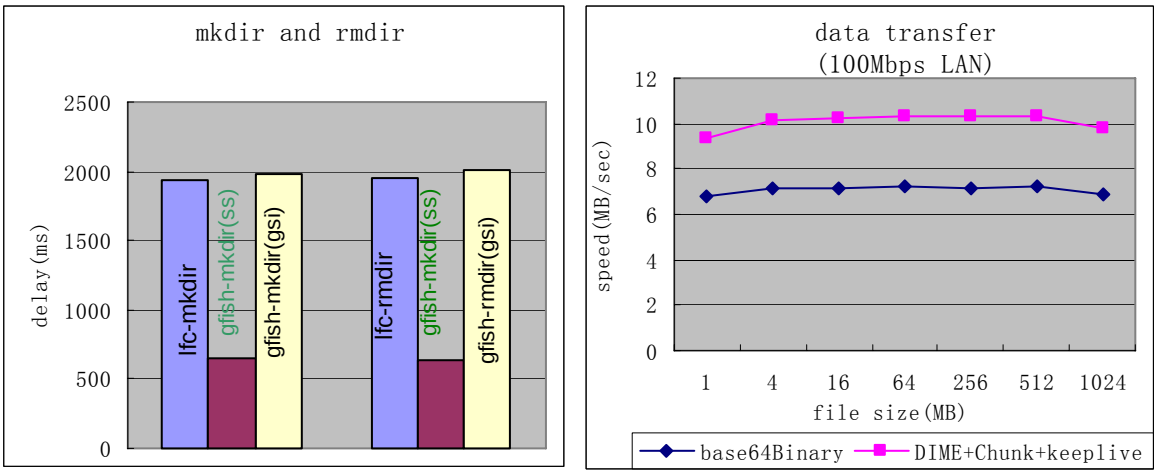

Figure 3: Performance test result of GFISH WS

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. GSOAP tuning is very useful in LAN environment (right side of figure 3), about 42.3% increase than non-tuning, but it doesn't help over high-latency network (it's even lower than "scp", and result isn't listed here). To improve the performance of transferring large files, gridftp API in CoG jglobus tool is introduced in GFISH client, which can perform transfer from LCG SE directly in multiple-stream mode and achieve high performance.

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 middlewares 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. If our GFISH tool is used, the 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-cp" command exists. 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 by using myproxy-init.
3) Executes 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 is very small, about 6MB. Thus, the tool can be downloaded on demand. However, it doesn't help when the executable is compiled and linked with GFAL or globus libraries together.

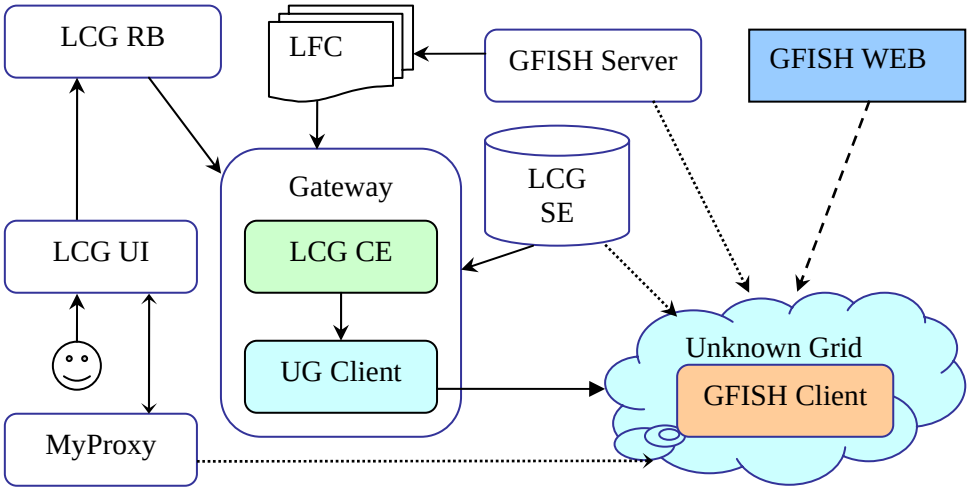

Figure 4: An example of use scenario

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.