EMI_datalib - joining the best of ARC and gLite data libraries

J K Nilsen¹, D Cameron¹, A Devresse², Zs Molnar³, Zs Nagy⁴ and M Salichos³

¹ Department of Physics, University of Oslo, P.b. 1048 Blindern, N-0316 Oslo, Norway
² University of Nancy
³ CERN
⁴ Nemzeti Informacios Infrastruktura Fejlesztesi Intezet (NIIFI) H-1132 Budapest, Victor Hugo u. 18-22., Hungary
E-mail: j.k.nilsen@fys.uio.no

Abstract. To manage data in the grid, with its jungle of protocols and enormous amount of data in different storage solutions, it is important to have a strong, versatile and reliable data management library. While there are several data management tools and libraries available, they all have different strengths and weaknesses, and it can be hard to decide which tool to use for which purpose. EMI is a collaboration between the European middleware providers aiming to take the best out of each middleware to create one consolidated, all-purpose grid middleware. When EMI started there were two main tools for managing data - gLite had lcg_util and the GFAL library, ARC had the ARC data tools and libarcdata2. While different in design and purpose, they both have the same goal; to manage data in the grid. The design of the new EMI_datalib was ready by the end of 2011, and a first prototype is now implemented and going through a thorough testing phase. This presentation will give the latest results of the consolidated library together with an overview of the design, test plan and roadmap of EMI_datalib.

1. Introduction

In a Grid environment, access to data and the ability to move data around are critical aspects of any workflow. Several technologies have emerged from Grid data management requirements: Storage Elements which allow files residing on media such as disk or tape to be accessible on the Grid, Replica Catalogs which contain information on the locations (replicas) of files and possibly file metadata, Access Protocols for reading and writing data on Storage Elements and Data Management Tools to link all these together. These tools typically consist of libraries and end user tools (e.g. services or command line interfaces) which use the libraries. Using these tools hides the underlying complexity from the Grid user, allowing him or her to manipulate and access data independently of the storage technology or access protocol used.

Over several years, the gLite [1] and ARC [2] projects have each developed data management tools, and these have gradually changed and adapted to new technologies and user requirements. The growing standardisation of Grid technologies has brought these two projects closer together to an extent that the same physical data can now be accessed and moved around by both tools. The European Middleware Initiative (EMI) [3] was started in 2010 to bring together several
different middleware providers, to harmonise and consolidate commonalities between them and to provide a single entry point to Grid software. Part of this project involves consolidation of commonality in the gLite and ARC data management libraries into a single product, with the aim of reducing maintenance costs and eliminating duplicate effort for future developments. This tool is called the EMI_datalib and this paper describes how the core libraries of gLite and ARC will be merged together to create it. The paper is structured as follows: Sections 2 and 3 present the existing gLite and ARC data management tools, Section 4 shows how they are merged into the EMI_datalib, Section 5 describes some first results and Section 6 details future plans.

2. gLite Data Library and Tools

gLite is a grid software middleware project for the LHC experiments at CERN. Its first prototypes were finalized in 2004. In 2006, gLite 3.0 became the official middleware of the Enabling Grids for E-sciencE (EGEE) project. Now, the project is supported by the European Grid Infrastructure (EGI). gLite was adopted by more than 250 computing centers, and has more than 15000 users worldwide.

The core component of gLite data management is the Grid File Access Library (GFAL) library, as shown in Figure 1. It provides a POSIX-like interface for data access and has a pluggable architecture, where different plugins provide support for different data transfer protocols. This means that files on a variety of Storage Elements accessible via different protocols can be used in exactly the same way as files on a local file system. The following transfer protocols are supported:

- GridFTP [4]: File Transfer Protocol with GSI security
- File: Regular POSIX [5] access to a local file system
- RFIO: Native random access protocol for the Disk Pool Manager [6] Storage Element
- DCAP: Native random access protocol for the dCache [7] Storage Element

It also has a plugin for interaction with the LCG File Catalog (LFC) [8], which provides a hierarchical logical namespace and a mapping from logical filenames to physical replicas. As well as the POSIX-like interface, GFAL has a plugin for the Storage Resource Manager (SRM) [9], which provides a management interface to Storage Elements for operations such as reserving space and staging data between different media types. Also part of the gLite data library is the lcg_util library, which provides higher-level file operations and uses GFAL for the actual file access. The lcg_util command line tools are thin wrappers around the lcg_util library.
3. ARC Data Library and Tools

The Advanced Resource Connector (ARC) [2] is a Grid middleware originally conceived to connect distributed resources in Nordic countries into NorduGrid. It now forms the basis of national Grid infrastructures in those countries and in many other countries. The middleware consists of a Computing Element to connect computing resources to the Grid, an Information System to provide information on resources and jobs and Client Tools to enable access to those resources.

![Figure 2. ARC data management library and tools.](image)

The ARC data library, libarcdata2, is the core data management part of the client tool set (see Figure 2). It has a pluggable architecture, with different plugins providing support for different data transfer protocols. It has a file-based (non-POSIX) interface providing higher-level operations to read and write files and plugins for the following protocols:

- GridFTP: File transfer protocol with GSI security
- File: Regular POSIX access to a local file system
- LFC: LCG File Catalog
- SRM: Storage Resource Manager
- HTTP(s/g): HTTP transfer protocol with SSL or GSI security
- xrootd [10]: Protocol for high-performance data access

libarcdata2 supports copying files between any two protocols, however does not support 3rd party transfer where data is transferred directly between two remote endpoints. Since all data flows through the point where the transfer was initiated, the library supports automatic buffer handling and checksum calculation.

The ARC command line interface provides Unix-like tools (arccp, arcls, arcrm) to manage Grid data much like the corresponding commands for local files. The ARC Computing Element (A-REX) uses libarcdata2 to upload and download files for jobs that are sent to it. External clients also exist, some of which use the Python API.

4. EMI_datalib

From the preceding two sections it can be seen that a lot of functionality is duplicated in both gLite and ARC libraries. Therefore it seems natural that one of the goals of EMI can be fulfilled by consolidating the two libraries into one, EMI_datalib. Concurrently to EMI_datalib, a new version of GFAL, GFAL2, is being implemented which is a complete reimplementation of GFAL with new plugins for the different protocols. GFAL2 forms the basis of the consolidated EMI_datalib, which is illustrated in Figure 3.
The proposed architecture will keep both the POSIX-like interface provided by GFAL2 and the file-based interface provided by libarcdata2. In addition, a transfer interface will be made available in GFAL2 to handle initiation and monitoring of 3rd party transfers.

In the proposed architecture, libarcdata2 will use the POSIX-like interface of GFAL2 through a GFAL-plugin, and thus no other plugins will be needed for libarcdata2. While third-party clients requiring POSIX-based byte-level data access can use the GFAL2 library directly, file based, data moving clients such as the lcg_util and ARC CLIs and A-REX will use the libarcdata2 file-based interface. Additionally, FTS3 and parts of the lcg_utils and ARC CLIs will use the 3rd party transfer interface of GFAL2. Hence, the lcg_util library can be removed. A Python library will be created to replace the needed functionality of the lcg_util Python API.

Note that the LFC and SRM plugins are put under GFAL2, even though they are (strictly speaking) file based. Having these plugins under GFAL2 allows POSIX access to files without needing to know physical file locations. For the same reasons, even though HTTP and xrootd plugins already exist in libarcdata2, HTTP and xrootd TURLs may be returned from the SRM plugin and the corresponding plugins should therefore be contained within GFAL2.

For the special case of 3rd party transfer, needed by FTS3, neither libarcdata2 nor GFAL2 currently have support for this. As this feature is crucial for FTS3 and will require some work to be fully supported in the libarcdata2 GFAL2 chain, an intermediate 3rd party transfer library will be made available. To take advantage of the existing GFAL2 transfer plugins, this library will be closely related to the GFAL2 plugin. The intention is that FTS3 will first use this new library directly, while later libarcdata2 will add support for 3rd party transfer using this library so that also FTS3 can use the high-level C++ interface of libarcdata2.

This design brings several benefits:

- One set of file transfer plugins instead of two,
- One set of control plugins instead of two,
• One file-based library and one POSIX-like library,
• GFAL gets support for HTTP and xrootd transfer,
• ARC gets support for 3rd party transfer,
• ARC gets support for BDII (an information system not currently used by ARC),
• No change for existing users,
• A clear division of components, with one file-based layer and one underlying POSIX layer with the transfer plugins should simplify support.

5. First Results
While the complete list of features for EMI_datalib is not planned to be implemented before the fall of 2012, the main building blocks, the GFAL2 library and the arcdatalib2 GFAL2 plugin, are already at a prototype level. The most important features, the third-party transfer and the use of the GFAL2 plugin in ARC have both been demonstrated:

• FTS 3 demonstrated successful third-party transfers using GFAL2 in April. We used dCache servers as gridftp endpoints in the EMI test-bed, and executed normal FTS transfer jobs. In FTS 2 terminology, it was an SRM-less transfer in a urlcopy "channel". GFAL2 was able to execute the transfers and report results back to FTS after which FTS changed and stored the job states accordingly. Report back transfer markers were no in place yet for the demonstration, but this will be supported in the first official version of GFAL2.
• Using the ARC CLI, we successfully demonstrated the arcdatalib2 GFAL2 plugin at CHEP2012. In a live demonstration, files where copied, listed and removed from a DPM instance in the EMI test-bed at CERN.

6. Future Plans
The design of EMI_datalib is complete and a prototype implementation exists with limited functionality. While a fully functional implementation of EMI_datalib is planned to be finished during the fall of 2012, a first product preview with the core functionality, i.e., GFAL2 with a POSIX-like interface and a 3rd party transfer interface and libarcdata2 with a GFAL2 plugin, is already implemented and will be released in July 2012 as part of the first update of the EMI 2 Matterhorn release[11]. External clients using GFAL may need some changes to switch to GFAL2, and those clients using lcg_util lib will have to switch to libarcdata2, but for the end users there should be no visible difference to using EMI_datalib. By the end of the EMI project the consolidation will be complete, with a unified library in use by all gLite and ARC data management tools, simplified support and lower maintenance costs.

Acknowledgements
This work is co-funded by the European Commission as part of the EMI project under Grant Agreement INFSO-RI-261611.

References
[1] gLite Web site. URL http://glite.cern.ch/
[2] Ellert M et al. 2007 Future Gener. Comput. Syst. 23 219-240 ISSN 0167-739X
[3] European Middleware Initiative project Web site. URL http://www.eu-emi.eu/
[4] Alcock W 2003 GridFTP: Protocol Extensions to FTP for the Grid Tech. Rep. GFD-R-P. 020 Global Grid Forum
[5] IEEE 2008 Portable Operating System Interface (POSIX.1-2008) Tech. rep. The Open Group
[6] DPM Web site. URL https://svnweb.cern.ch/trac/legdm/wiki/Dpm
[7] Fuhrmann P and Glzow V 2006 Euro-Par 2006 Parallel Processing (Lecture Notes in Computer Science vol 4128) ed Nagel W, Walter W and Lehner W (Springer Berlin / Heidelberg) pp 1106–1113
[8] Baud J P, Casey J, Lemaitre S and Nicholson C 2005 14th IEEE International Symposium on High Performance Distributed Computing, HPDC-14. pp 91 – 99
[9] Sim A et al. 2008 The Storage Resource Manager Interface Specification Version 2.2 Tech. Rep. GFD.129 Open Grid Forum available at: http://www.ogf.org/documents/GFD.129.pdf
[10] Dorigo A, Elmer P, Furano F and Hanushevsky A 2005 WSEAS Transactions on Computers vol 4 pp 348–353
[11] EMI 2 Matterhorn release Web site. URL http://www.eu-emi.eu/emi-2-matterhorn