Grid-Enabled Standards-based Data Management

Lana Abadie[1] Paolo Badino[1] Jean-Philippe Baud[1] James Casey[1] Akos Frohner[1]
Gilbert Grosdidier[2] Sophie Lemaitre[1] Gavin Mccance[1] Rémi Mollon[1]
Krzysztof Nienartowicz[1] David Smith[1] Paolo Tedesco[1]

[1]*CERN, European Organization for the Nuclear Research, Geneva, Switzerland, Inc.*

[2]*LAL/IN2P3/CNRS, Faculté des sciences, Orsay, France*

Abstract

The world's largest scientific machine – the Large Hadron Collider (LHC), situated outside Geneva, Switzerland – will generate some 15PB of data at rates up to 1.5GB/s (in the case of the heavy-ion experiment, ALICE) to tape per year of operation. The processing of this data will be performed using a world-wide Grid, the (worldwide) LHC Computing Grid built on top of the Enabled Grid for E-sciencE and Open Science Grid infrastructures. The LHC Computing Grid, which has offered a service for over two years now, is based upon a tier model comprising some 150 sites in tens of countries.

In this paper, we describe the data management middleware stack - one of the key services provided by data grids.

We give an overview of the different services implemented, a disk-based storage system which can support encryption, tools to manage the storage system and access files, the LCG File Catalogue, and the File Transfer Service. We also review the relationship between these services.

1. Introduction

1.1. The Large Hadron collider

The LHC [1] is located at CERN [2]. There are 4 main experiments, ALICE [3], ATLAS [4], CMS [5] and LHCb [6]. The four detectors are built 100 meters below the ground. Two beams of protons will be accelerated to the highest energy (14 TeV) ever achieved in a laboratory to allow the creation of new particles. The event rate from the different detectors will be very high, around 1PB/sec before a multi-level trigger system[1] and will be reduced to around 100MB/s (for proton-proton experiments). In total, some 15PB of data will have to be stored for subsequent physics analysis and to be accessible by users from hundreds of institutes. The LHC Computing Grid (LCG) [7] is responsible for developing, building and maintaining a distributed computing infrastructure for the storage and analysis of data from the four LHC experiments.

1.2. The LHC Computing Grid Model

The LCG is built on a model that consists of a number of main tiers [8], each having a specific role as follows:

- Tier-0 (CERN): all the raw data coming from the data acquisition system of the LHC experiments will be safely stored (first copy). Data from the first pass reconstruction will take place at Tier-0. A copy of the reconstructed data will be also stored.
- Tier-1: their roles depend on the experiment. They are responsible for managing the permanent data storage (raw, simulated and processed data), providing computational resources for reprocessing and for analysis processes that require access to large amounts of data. There are roughly 11 Tier-1 sites. They will also store a backup of the raw data and additional copies of the reconstructed data.
- Tier-2: they provide computational capacity and appropriate storage services for Monte Carlo event simulation and for end-user analysis. The Tier-2 sites will obtain data from the Tier-1 site they depend on and the data generated at Tier-2 centres will be sent to Tier-1 centres for permanent storage. There are more than 100 Tier-2 sites.

Tier-0, Tier-1 and Tier-2 sites have markedly different requirements in terms of data transfer, storage capacity and computing resources. A grid solution was adopted since it fits well the overall processing model and allows

[1] A multi-level trigger system filters the most interesting physics events. The design and the number of levels of the trigger system depend on the physics goals of the experiment. For instance, there are two trigger levels at LHCb.

24th IEEE Conference on Mass Storage Systems and Technologies (MSST 2007)
0-7695-3025-7/07 $25.00 © 2007 IEEE

efficient resource sharing and usage, enabling complex physics data analysis.

To fulfill these requirements, the LCG grid uses primarily the Open Science Grid (OSG) [9] and the EGEE [10] infrastructures. In this paper we will focus on the EGEE services.

1.3. Middleware Services

The Enabled Grid for E-sciencE (EGEE) project has implemented a set of grid middleware services [11] known collectively as gLite. gLite is based on a Service Oriented Architecture that allows:

- interoperability between different grids;
- flexible exploitation of the middleware services according to specific needs;
- support of *de-facto* grid standards.

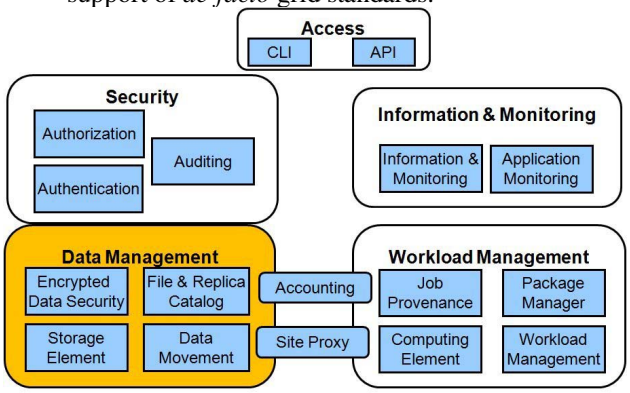


Figure 1. The gLite middleware services.

Figure 1 illustrates the gLite service architecture showing the four service categories. The *security service* is responsible for authentication, authorization and data encryption. The *information and monitoring service* collects information about the status of the different grid resources. The *Workload management service* distributes and manages jobs across the grid. The *data management services* provide reliable tools to store, replicate and retrieve files.

1.4. gLite components

Figure 2 shows the different components implemented and used by the gLite services.

In this paper, we focus on the data management components.

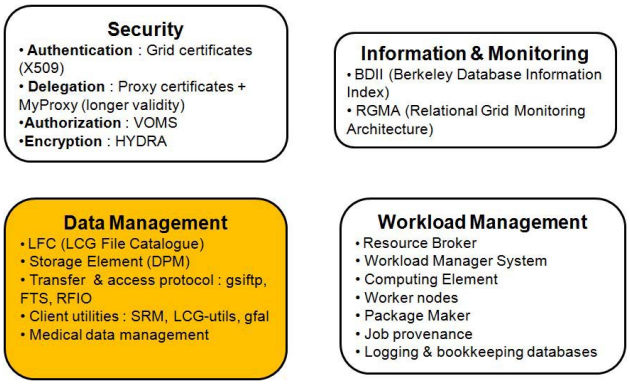


Figure 2. gLite components.

2. The LCG File Catalog

The number of files stored in the grid is typically of the order of 10^9. These should be accessible from any grid site, without requiring the user to know their physical location. A file can have as many replicas as needed and can be stored in any types of storage systems.

To provide a full flexibility to the users, a file catalog has been designed to allow a user from any EGEE grid site to locate and retrieve files using a Logical File Name (LFN). The LCG File Catalog (LFC) has been developed by the Data Management team at CERN.

2.1. Logical path concept

In UNIX, each file and each directory have a name defined by the user. A file or a directory can have one or several soft links also called aliases. A file or a directory can be identified by its inode (defined by the operating system) or by its full path. The full path is commonly used as it is human-readable.

The LFC uses similar concepts. The full path (resp. inode) is called an LFN (resp. a Grid Unique Identifier - GUID). Likewise UNIX, the LFN is often used to interrogate the LFC and identify a file. The soft link is called an alias.

Thus an LFN is associated with a GUID. An LFN can have aliases. Each path to a replica of a file is identified with a "storage URL" or SURL. The syntax of the SURLs depends on the type of the storage system. The concept of SURLs also allows hiding the physical path of a file within a storage system. In other words, there are two levels of logical path, the first one on the grid level (LFN) and the second one on the storage system itself (SURLs).

The LFC provides mapping between an LFN or a GUID and SURLs, as shown in Figure 3.

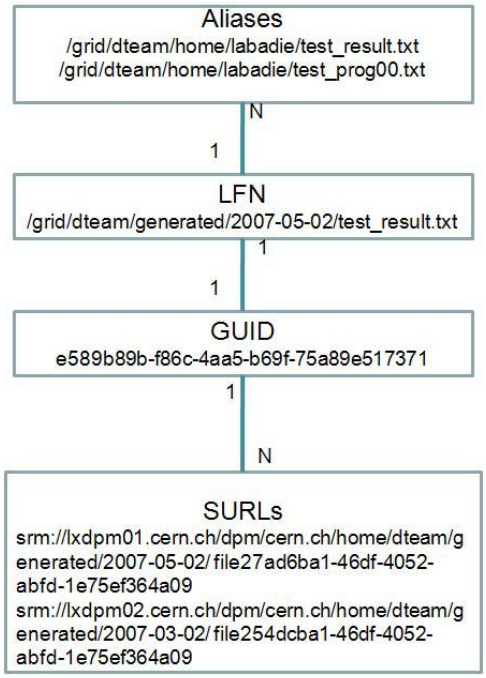


Figure 3. Relationship between LFNs, GUIDs, SURLs and aliases.

The introduction of the logical path concept permits independence of the hardware setup of the storage element. Thus, there is no need to know the name of the storage element that hosts the file. This allows the architecture of the storage system to be transparent against changes to the physical path (such as the name of the host machine or the file system).

2.2. Path representation using tables

Path and associated permissions are stored in a relational database. The LFN is represented hierarchically, as in the UNIX system and as shown in Figure 4.

2.3. Implementation outline

The LFC is built on a proven architecture, using a multi-threaded C-based application middle tier that may be scaled out horizontally for both availability and performance.

It uses a database backend to store the LFNs, GUIDs, SURLs, aliases and file permissions (both Oracle and MySQL are supported). It implements a secure catalog that uses grid certificates for authentication and Virtual Organization Membership Service (VOMS)[2] [12] for authorization in order to be compliant with the security service. The gLite security service provides all the necessary libraries to allow other services (Data Management, Workload Management) to be compliant with the EGEE security policy.

fileid	parent_fileid	name
0		/
2	0	grid
3	2	dteam
4	3	generated
8	4	2007-05-02
10	8	test_result.txt

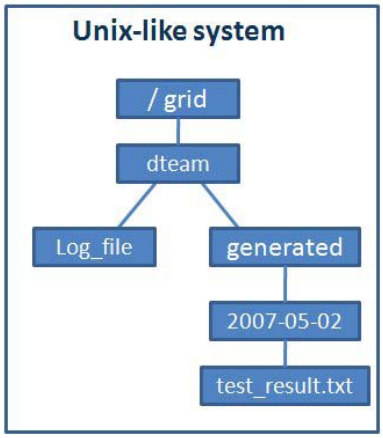


Figure 4. Representing an LFN with a table.

To satisfy administrators' and the users' requirements, the LFC can be accessed via a transactional C-library with two bindings (Python and Perl) or the LFC Command Line Interface (CLI), similar to the UNIX CLI. A web service Data Location Interface (DLI) [13] has also been implemented. Unlike the other bindings, this latter interface is unsecure (no authentication) and only allows information retrieval. It is mainly used by the Resource Broker component of the Workload Management service.

Volatile files[3] are not declared in the LFC.

3. The Disk Pool Manager

As described above, the different tiers of the LCG have markedly different storage capacity and service requirements. The Tier-0 site uses the CERN Advanced STORage manager (CASTOR) [14] storage system. It is a Hierarchical Storage Management (HSM) system developed at CERN used to store physics production files and user files. CASTOR manages disk caches and the data on tertiary storage or tapes.

dCache [15] is installed at most Tier-1 sites (CASTOR is installed on a few Tier-1 sites). dCache provides a

[2] VOMS allows classifying users that are part of a Virtual Organization (VO), i.e. they share a set of attributes that will be granted to them upon request and to include that information inside Globus-compatible proxy certificates. A user can be part of one or several VOs.

[3] Files which have a lifetime less than one month are considered as volatile files.

system for storing and retrieving huge amounts of data, distributed among a large number of heterogeneous server nodes, under a single virtual file system tree with a variety of standard access methods. dCache is a joined effort between the Deutsches Elektronen-Synchrotron (DESY) [16] in Hamburg and the Fermi National Accelerator Laboratory (FERMILAB) [17].

DPM is installed at most Tier-2 sites (there are some Tier-2 sites which use dCACHE). In this section, we focus on the DPM, implemented by the Data Management team at CERN.

3.1. Motivation

The DPM primarily addresses the storage needs of the Tier-2 sites. They usually require at most a few hundreds of TB of storage capacity. The data will be stored for a few months and permanent data will be sent to the Tier-1 site they depend on.

However configuring and managing a tape-based storage system is expensive in terms of money and manpower and is typically not required at such sites and hence not currently provided by the DPM.

Based on the requirements that have been collected, the main objective of the DPM is to provide a secure, reliable and scalable storage system that offers simple configuration and management.

3.2. Architecture

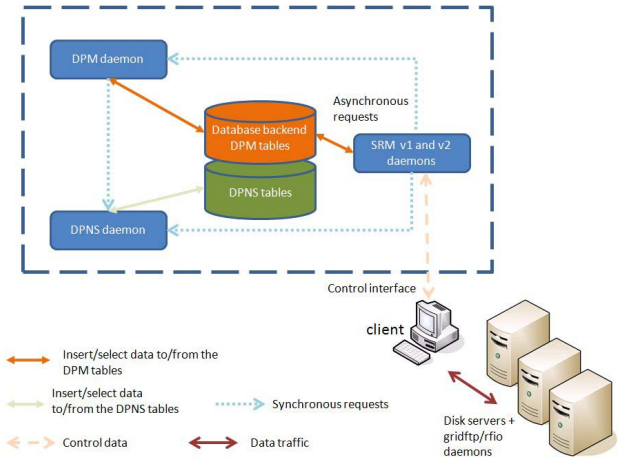


Figure 5. Overview of the DPM design.

Figure 5 shows the outline of the DPM design architecture.

To allow interoperability between different storage system types, a Storage Resource Manager (SRM) server and client have been designed. SRM is a common standard based on web services to manage storage systems and is further described in another paper submitted to this conference. The DPM supports both versions 1.1 and 2.2 of the SRM standard [18]. The requests are serviced by the respective SRM daemon. If an asynchronous request, such as a get or a put, is submitted to an SRM server, it is inserted into the DPM database and the SRM client polls for its status. Then, the DPM daemon picks up the request from the DPM database and processes it. The DPM daemon contacts the DPNS (Disk Pool Naming Service) daemon regarding authorization and mapping between Site File Name (SFNs) and Physical File Names (PFNs) (see next sub section for details).

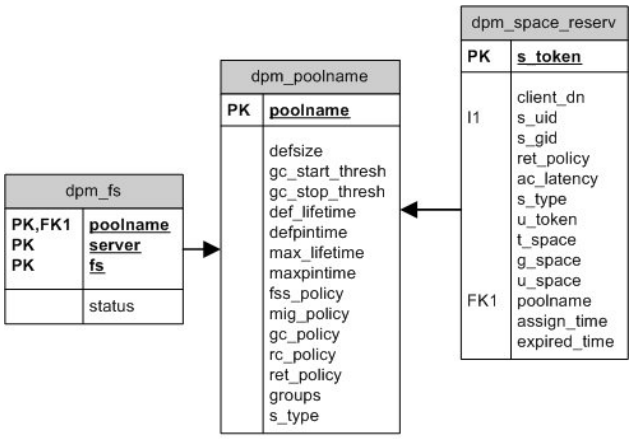
If the request is synchronous, such as obtaining the current DPM configuration, reserving space, it is immediately processed by the DPM daemon.

Physical files are then accessed and stored via any of the supported protocols, such as Remote File Input/Output (RFIO), GSIFTP [19], http, xrootd [20].

3.3. The Database Backend

As shown in Figure 5, the DPM uses a database that is split in two parts:

- The DPM part contains information related to the DPM configuration and status. Figure 6 shows an extract of the DPM table schema. The current setup of the DPM is retrieved using the **dpm_fs** and **dpm_pool** tables. All the coming asynchronous requests (get and put) are stored in the **dpm_pending_req** table. A request is identified by the *r_token* column in the **dpm_pending_req** table. The user or an application can request to put or get several files at once. Each file involved in a given put/get request is associated with the pair (*r_token, f_ordinal*) which are two columns of the **dpm_put_filereq** and **dpm_get_filereq** tables. The DPM daemon selects one of the pending requests (i.e. one given *r_token*) and processes it. Distinct tables for each type of request enable to gain in performance especially in case of concurrent accesses.



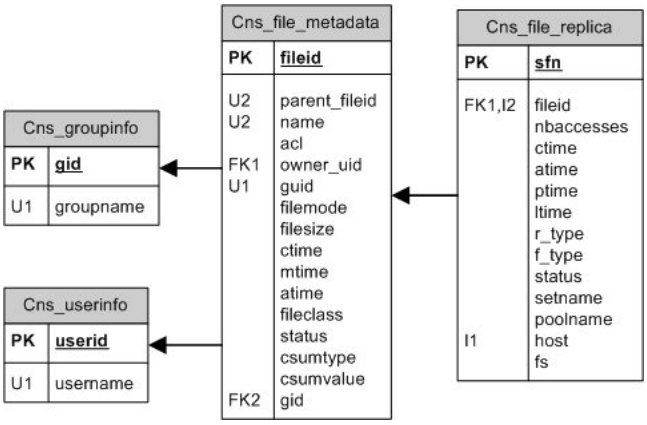


Figure 7. Extract of the DPNS table schema which is the same as for the LFC.

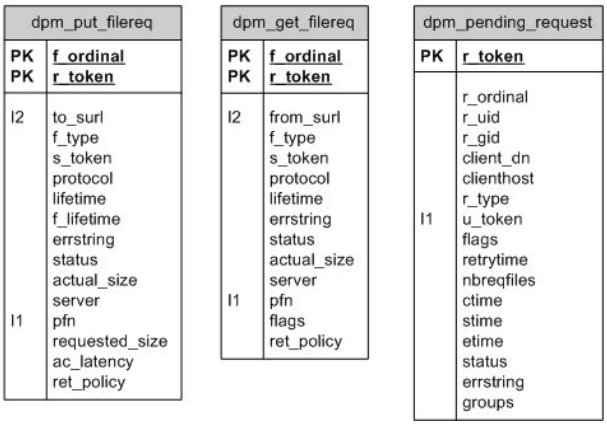


Figure 6. Extract of the DPM table schema.

- The DPNS part allows mapping between the SFN (Site File Name) and the PFN (Physical File Name) and retrieving the permission associated to a SFN. An extract of the table schema is presented in Figure 7. The **Cns_userinfo** and **Cns_groupinfo** tables contain information respectively about the user Distinguished Name (DN) and VOMS attributes. The physical location of files is stored in the **Cns_file_replica** table. The *sfn*, *host* and *fs* columns represent respectively the full physical path (PFN), the name of the machine and the file system where the the file is stored. The **Cns_file_metadata** table contains information about the SFN and file permission. The SFN corresponds to the SURL of the LFC as illustrated in Figure 8. The TURL is the association of a PFN and an access protocol such as RFIO, GSIFTP.

The *acl* column fixes the permission of a file or a directory. It follows the same scheme as the UNIX file system.

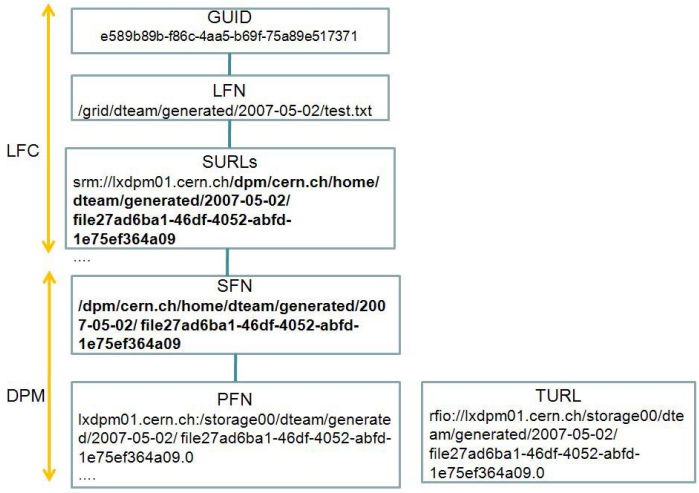


Figure 8. Relationships between the SURLs, SFNs and PFNs.

All persistent states are maintained in the database and are required in case of system recovery.

3.4. Key aspects in authentication and authorization

Figure 9 illustrates the authentication and authorization mechanisms that allow a client to access a given DPM site. This mechanism follows the rules of the gLite Security service. The authentication is based on grid certificates. This step is performed by the first daemon contacted by the client request. For example, if it is a SRM request, the SRM daemon will authenticate the user. Both the DPM host and the client must have valid grid certificates; otherwise the user request is cancelled.

The next step is to check that the user is entitled to access the DPM site. There are two scenarios:

1) A user comes with a valid voms-proxy, i.e. the user is already associated with a VOMS. Then s/he has the right to access the DPM.

2) A user comes with a valid grid-proxy. Its user DN has to be declared in the lcgdm-mapfile and the grid-mapfile. If the user DN is not in one of the mapfiles, the request fails. Otherwise the Virtual Organization (VO) associated with the user DN is retrieved[4]. This information user DN and VO are mapped to virtual IDs.

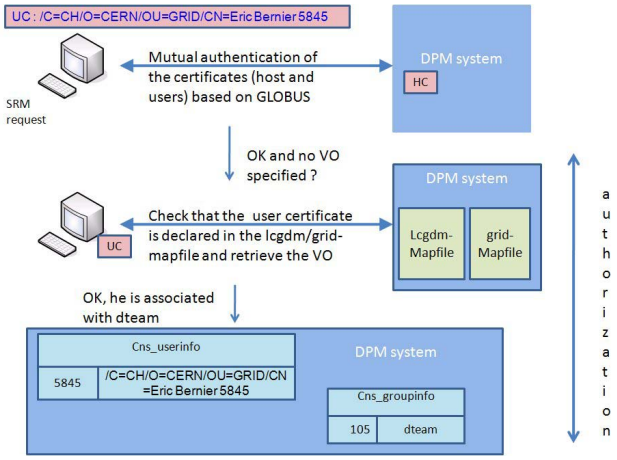


Figure 9. Overview of the authentication and authorization mechanisms.

They are inserted respectively in the **Cns_userinfo** and **Cns_groupinfo** tables if they do not exist. Virtual IDs are very useful as they permit to remove the concept of pool account and to support Access Control Lists (ACLs). Thus the site administrator does not have to create accounts or to modify the password file. They also allow a fast check of the ACLs and ownership of a file.

3.5. Pool concept

A pool is an ensemble of (disk server, file system). Referring to Figure 6, the **dpm_pool** table describes the characteristics of a pool.

The quality of a pool is defined in the *ret_policy* column. It can be custodial (very good quality), replica (to be used for replicas) or output (medium quality). This characteristic depends on the hardware and is defined during the setup of the DPM by the DPM administrator. However it can be updated if a hardware change occurs.

A file is associated with a quality type (*ret_policy* column of **dpm_put_request** and **dpm_get_request** tables). It is set by the user and it determines the selection of the pool in which this file will be stored. The *f_type* column of **dpm_put_request** and **dpm_get_request** tables corresponds to the type of file. The *f_lifetime* column of **dpm_put_request** table corresponds to the lifetime of a file in the storage system. *The f_type* column can be set to permanent for files which are never to be deleted (and *f_lifetime* is equal to infinite), volatile for files to be deleted (when *f_lifetime* is expired) or durable (warn the owner of the file by mail that *f_lifetime* has been expired). However the durable space type is not implemented yet.

A pool can be associated to one or several VOs (*groups* column). In other words, it is like attributing ACLs on pools. Thus only the users which are in the specified VOs will be authorized to store files in this pool. It allows preventing greedy VOs from storing their files in any pool and saturating the DPM quickly.

A pool has a default reserved space (*defsize* column). It is used to reserve space for a given file by default if no size is specified in the client request.

3.6. Replica and file system selection

A file can be stored in different locations. The selection of a file system is based on the round robin mechanism. The main reasons are the following ones:

- It is fast to compute;
- It reduces the number of concurrent accesses for a given file system.

A file can have several replicas. The question is which one to select. The algorithm is quite intuitive and fast to compute:

1. Get the list of replicas with the physical location;
2. Select the ones which are in the same domain as the client;
3. Select randomly one of them if still several or if no host is in the same domain.

3.7. Outline of the DPM server implementation

The core of the DPM architecture is the distinct daemons and the database. The communication between daemons and the database is based on sockets. As with the LFC, the main application layer is via multi-threaded daemons implemented in C and the database backend can use either Oracle or MySQL databases. ProC [21] (resp. C MySQLclient library) is used as a DB interface if it is an Oracle DB (resp. MySQL).

An automated garbage collection is implemented to remove expired volatile files.

A clean-up mechanism has been implemented to delete incomplete files based on the timeout which corresponds to the *lifetime* column referring to Figure 6. It prevents from having internal consistency in the DPM.

Each daemon produces a daily log file used for offline debugging and analysis. The different log files are also used for auditing purposes. Each user is tracked by its DN and the different operations s/he performs.

To improve the robustness of the system, the different DPM sites can report their problems and comments to the support staff via different grid services [22].

[4] If a user issues a voms-proxy command and s/he belongs to several VOs, then there is a virtual ID for each pair (user DN, VO he is part of).

3.8. Tools to manage the DPM

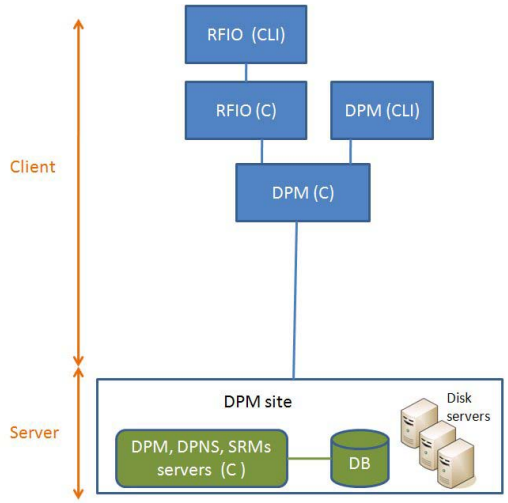


Figure 10. Software architecture of the client tools specific to DPM.

Figure 10 shows the different tools specific to the management of the DPM. There are two utilities:

- The RFIO CLI and C-library permit to manipulate files and get information about files via the RFIO protocol. The RFIO API is POSIX compliant.
- The DPM C-library and CLI are the lowest level. It consists of functions which can contact both the DPNS and DPM daemons. It allows the expert user to define its own parameters instead of using the default ones.

The RFIO and DPM libraries are thread-safe.

The RFIO and DPNS commands are very similar to UNIX commands, thus users can use them intuitively. In addition, no database knowledge is required.

There is no checksum to verify whether a file is corrupted or not. The main reason is that this operation is time-consuming and there is no standard method provided by the file access protocol (RFIO, GSIFTP, etc.). However the user can implement its own checksum and inserts it in the LFC. There are two columns (*csumtype* and *csumvalue*) in **Cns_file_metadata** referring to Figure 7.

It is up to the user or site administrator to guarantee data consistency when writing applications based directly on the DPM library. The DPM and RFIO libraries (or CLI) do not communicate with the LFC.

The choice of this implementation is justified by the following user's requirements:

- it should be possible to have different permissions according to the location of a storage resource. We have a use case, where ACLs should be more restrictive on storage resources which are not local.
- it should be possible to store files in a storage system locally. In other words, these files should not be visible in the EGEE grid. This is achieved if these files are not declared in the LFC.

4. Generic client utilities

Regardless of whether a grid user is a physicist, physician or an engineer, they should all be able to use the client utilities to access the gLite services and in particular the storage system. Moreover there are different types of storage systems. Generic client utilities must be implemented to make the interoperability between different storage types possible.

The gLite Data Management team has contributed in the implementation of several types of tools to allow the user to interact with different types of storage system.

4.1. Software architecture

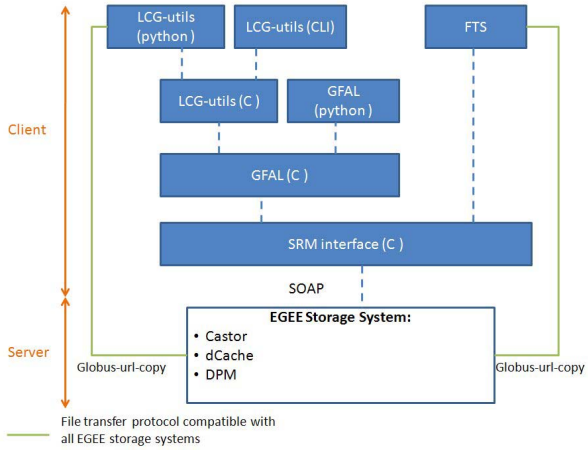


Figure 11. Software architecture to interact with the different types of storage systems.

Figure 11 shows the software architecture. The aim of this architecture is to provide access to any type of EGEE storage systems. There are different tools:

- The LCG-utils tools (Python and Perl module, CLI and C-library) [23] are top-level tools. In other words, they require little knowledge of the storage system implementation. They allow a transparent access to different types of grid catalogues and storage elements. The user can store, replicate, delete and copy a file using LCG-utils tools.
- The Grid File Access Library (GFAL) [24] is a C library which provides a POSIX interface to local and remote Storage Elements on the Grid.
- The File Transfer Service (FTS) allows file transfer between different storage systems. See section 6 for more details.

- The SRM interface is used by GFAL. This interface allows interoperability between the different types of storage system. It is a web service based on SOAP [25]. The API has been defined and agreed by the SRM consortium [18].

4.2. Data consistency

All these tools allow modular and flexible file management in the storage system. For the moment, there is no mechanism to check the authenticity, i.e. the data provenance is correctly tracked.

LCG-utils has been implemented to reduce human interventions and try to automate the different operations. For instance, if a replica is inserted or deleted, both the LFC and the storage system databases are up-to-date.

However if a disk server at a given storage system crashes, there is no notification to the LFC to delete the files. It is foreseen to provide such a script by the end of the year.

4.3. Integration with the grid middleware

The typical use case is a user who submits a job which needs to open a specific file stored in the EGEE grid. Where is the file stored? Which storage site is up and can handle the request? What protocol can be used to access the file?

A complete integration in the grid middleware stack and interoperability between the existent storage systems are means to ensure a long lifetime.

The Information & Monitoring service via the Berkeley Database Information Index (BDII) [26] provides two types of information:

- Static information such as the different end-points (storage elements for instance)
- Dynamic information such as the free space of a given storage element.

Figure 12 illustrates the interaction between the four gLite services. Authentication and authorization are performed for each grid component (LFC, DPM server and DPM disk servers). For the BDII, there is no authentication as anybody can have a read-only access.

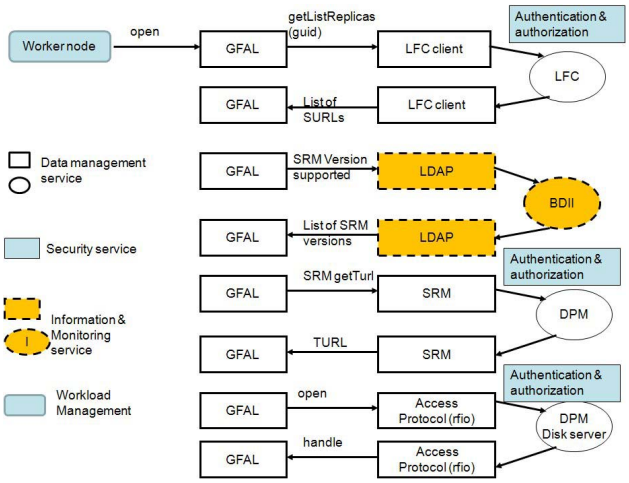


Figure 12. Example of a data flow of a call to gfal_open.

Referring to Figure 12, the worker node needs to open a file to execute the job. It issues a gfal_open. The LFC is called to return the list of replicas (SURLs) given a GUID. Then there is a call to the BDII to get the version of the SRM interface to use. The next step is to get a TURL from the DPM using the SRM interface. The file can be opened using an access protocol such as RFIO. The protocol is specified in the TURL. The selection of the protocol is the result of a negotiation between the SRM client and the SRM server.

5. Encrypted Data Storage

5.1. Objectives

The EGEE grid is used mainly by High Energy Physics (HEP) applications. However, main other areas are also actively represented in the project. In the specific case of medical institutes, additional requirements are involved, such as data encryption during the file transfer and in the storage system. To fulfill these requirements, the gLite Data Management Service has designed the Encrypted Data Storage (EDS) as shown Figure 13.

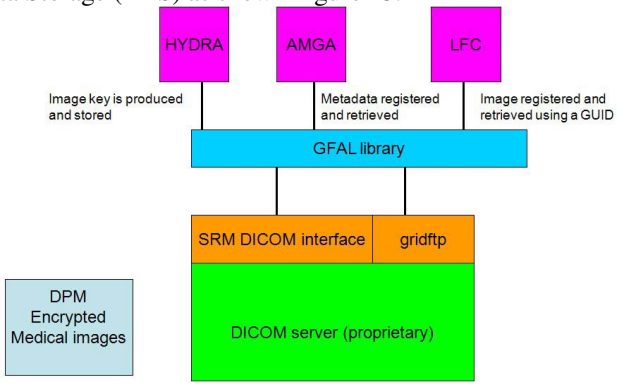


Figure 13. The EDS architecture.

5.2. Design

Digital Image and Communication in Medicine (DICOM) [27] is a medical image standard developed to satisfy clinical practice. A DICOM image integrates the image itself and metadata information (related to the patient for instance). DICOM servers allow storing and retrieving DICOM images both on disk and tape based back-ends. However the DICOM security model is rather weak. DICOM images are accessed only internally within the hospitals. By using the different EGEE services, DICOM images can be accessed from outside. An extended version of SRM (SRM DICOM) has been implemented to be both compatible with the grid and the DICOM protocol. A DICOM image is identified with a device triplet (3 UIDs) [27].

Each DICOM image has an entry in the LFC and is identified with a GUID so that the physical location of the image can be found.

ARDA Metadata Catalogue Project (AMGA) [28] is used to store relational information on medical images stored on the grid, plus information on patients and doctors.

The HYDRA [29] library allows medical images to be encrypted and decrypted using the file specific keys stored in the Hydra keystore.

GFAL is used to retrieve the medical images. The medical image stored in the DICOM storage is encrypted on the fly. The client decrypts it locally using the specific key (HYDRA). A replicated medical image is always stored encrypted in grid storage elements such as DPM.

There is an access control based on the ACL to check that a given user has the permissions to get a given replica when contacting the LFC.

6. File Transfer Service

6.1. Motivation

Given the data volumes and rates involved, a robust and reliable file transfer service, automatically implementing retry to cater with all common errors, is required. The LHC uses in average a file size of 1GB. Based on expected data rates, there will be 10^5-10^6 files transferred per day. Manual intervention, i.e. due to file transfers failures should not occur more than 1 per day at a given site as this problem is time-consuming. The service has to be reliable to 1 in $10^{5/6}$.To meet these requirements, the gLite File Transfer Service (FTS) has been implemented [30].

6.2. Architecture

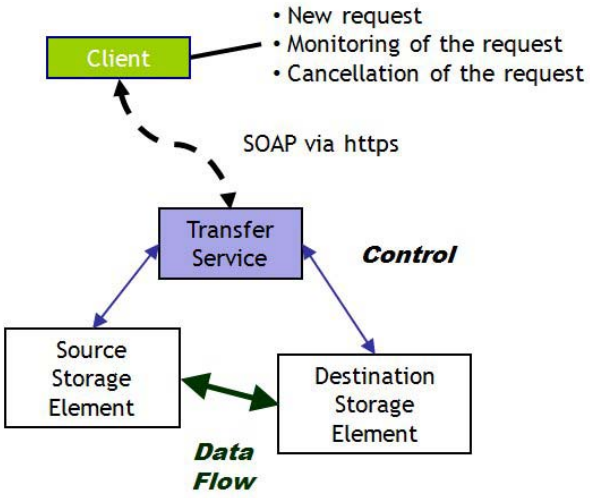


Figure 14. Outline of the architecture.

Figure 14 describes the architecture of this service.

The client securely connects to the service to submit a transfer request. The transfer service refreshes periodically the status of the transfer (polling mechanism). The client can reconnect to the service to check the transfer status or to abort its request. It is an asynchronous file transfer. The transfer service is centralized so it has a global view of the different requests. It allows load balancing and transfer scheduling.

6.3. The Channel concept

The FTS uses the concept of *channels*. A channel is an abstract unidirectional link between the source and destination storage elements. Each channel is independent which can be managed and configured distinctly. This concept has been introduced to optimize the bandwidth and to allow the VOs to manage their own channels.

6.4. Overview of the implementation

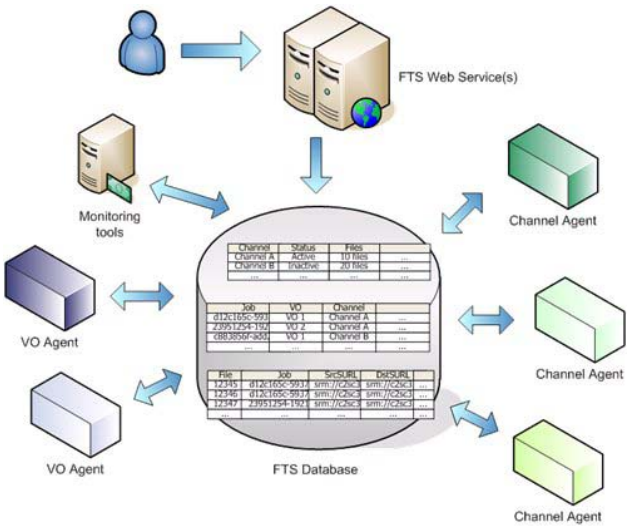


Figure 15. The architecture of the FTS.

The FTS architecture as represented in Figure 15 consists of the following components:

- A VO agent is the first element to be contacted when a request from a given VO is submitted. It authorizes the request and assigns a channel to the request. If the request is accepted, then its status moves to PENDING. It also handles the scheduling.
- A channel agent is responsible for monitoring of the transfer. It also balances requests on the channel between different VOs.
- A FTS database (Oracle database) contains all the information about the requests and the channels. It also allows collecting statistics and monitoring information.
- A FTS web service receives the requests and inserts the information in the database.

7. Test & Performance

Performance is a real issue for the FTS and for the LFC. Data rates from Tier-0 to Tier-1 vary from 50 to 400MB/s. These transfers occur 24 hours a day over a period of around 100 days. Data transfers from the Tier-2 sites to Tier1 are at lower rate, roughly tens of MB/s.

Independent test suites have been conducted to enhance the performance and the robustness of the data management components.

7.1. LFC tests

Queries against the LFC have been speed up with the use of bulk queries as it reduces the number of round trips between the client and server.

Several other tests [31] are being conducted to measure things like the impact of the size of the communication buffer between the client and a local/remote LFC server on the response time.

A test suite has been developed for stress tests.

7.2. DPM tests

Many sites have volunteered to carry out stressing tests. Their output was very fruitful as the hardware and network setups were different.

One example is the reading of files using the RFIO protocol by many simultaneous clients.

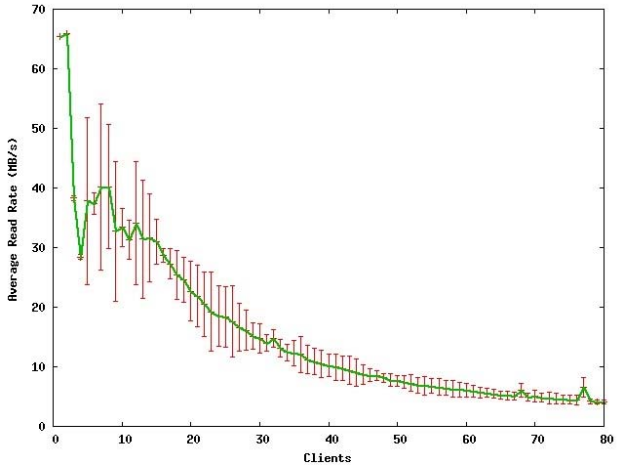


Figure 16. Read rate using the RFIO protocol in function of the number of clients.

Figure 16 shows the read rate in function of the number of clients. Each client tries to read a distinct file of 1 GB. The performance has been improved with the new version of DPM by limiting the number of round trips between the client and the server. For instance, with DPM version 1.5.10 the open time increased from ~2s with 1 client to ~12s with 80 clients. With 1.6.3 the open time increases from ~1.8s to ~2.7s with 80 clients. The authentication operation takes around 0.7sec. The one second left is due to the poll mechanism (the first poll to check the status of the request occurs after one second). However no users have complained about the DPM performance.

7.3. Client utilities

The SRM interface is tested by a dedicated group (the SRM testers) as there are different types of storage elements to be verified. They have developed test suites and all the results are presented in web pages.

This group has also started to set up stress tests. It allows verifying the robustness of the end-points – DPM in our case- and their behavior when heavily loaded.

All the results are collected on a web page [32].

The different functionalities of LCG-utils, GFAL, RFIO and DPM utilities have also been tested. It allows verifying the behavior of these functions in case of human errors and also the robustness of the components (LFC, DPM, etc.)

7.4. FTS tests

Concurrent request submission has been tested. It turned out that the FTS could accept up to 50 simultaneous connections without any failures. The robustness and reliability of the FTS have been evaluated via the LCG challenges. During these periods, the FTS is heavily loaded as hundreds of TB was transferred between CERN and Tier-1 sites such as (CNAF in Italy, DESY in Germany).

8. Conclusions

In this paper, we have reviewed the different components of the LCG Data Management service. The LFC is a secure file catalog that allows a user to retrieve replicas without knowing their physical locations. The DPM provides a lightweight, manageable and scalable storage system. The RFIO and DPM libraries allow file management and DPM configuration at low level. These tools are specific to the DPM.

The DPM as a storage system can be accessed by generic, user-friendly and reliable tools (LCG-utils, GFAL) allowing integration in the EGEE grid environment. The SRM interface provides interoperability between different types and implementations of storage systems.

The FTS offers a reliable transfer service for the LHC physics data between sites. Further studies on the scalability of the DPM and robustness of the FTS will be performed.

Medical institutes have also benefitted from this grid middleware, thanks to the EDS. The integration of the EDS with GFAL and DPM is under implementation and will have to be tested.

9. Acknowledgements

We would like to thank Greig Alan Cowan and Graeme Stewart for their fruitful cooperation with stressing tests and performance.

Also we would like to thank Jamie Shiers for his careful reading of the paper. His numerous comments, suggestions and corrections have substantially improved the quality of this text.

This work makes use of results produced by the Enabling Grids for E-sciencE project, a project co-funded by the European Commission (under contract number INFSO-RI-031688) through the Sixth Framework Programme. EGEE brings together 91 partners in 32 countries to provide a seamless Grid infrastructure available to the European research community 24 hours a day. Full information is available at http://www.eu-egee.org

10. Glossary

ACL: Access Control List
AMGA: ARDA Metadata Catalogue Project
BDII: Berkeley Database Information Index
CASTOR: CERN Advanced STORage manager
CLI: Command Line Interface
DESY: Deutsches Elektronen-Synchrotron
DICOM: Digital Image and Communication in Medicine
DLI : Data Location Interface
DN: Distinguished Name
DPM: Disk Pool Manager
DPNS: Disk Pool Naming Service
EDS: Encrypted Data Storage
EGEE: Enabled Grid for E-sciencE
FERMILAB: Fermi National Accelerator Laboratory
FTS: File Transfer Service
GFAL: Grid File Access Library
GUID: Grid Unique Identifier
HEP: High Energy Physics
HSM: Hierarchical Storage Management
LCG: LHC Computing Grid
LFC: LCG File Catalogue
LFN: Logical File Name
LHC: Large Hadron Collider
OSG: Open Science Grid
PFN: Physical File Name
RFIO: Remote File Input/Output
SFN: Site File Name
SRM: Storage Resource Manager
SURL: Storage URL
VO: Virtual Organizations
VOMS: Virtual Organization Membership Service

References

[1] LHC website [online].
Available: http://lhc.Ib.cern.ch/lhc/
[2] CERN website [online].
Available: http://public.Ib.cern.ch/public.
[3] ALICE website [online].
Available: http://aliceinfo.cern.ch/
[4] ATLAS website [online].
Available: http://atlas.web.cern.ch/Atlas/index.html
[5] CMS website [online].
Available: http://cms.cern.ch/
[6] LHCb website [online].
Available: http://lhcb.web.cern.ch/lhcb/

[7] LHC Computing Grid. Technical report. Editor Jurgen Knobloch. LCG-TDR-001. CERN-LHCC-2005-024. [online].
Available: http://lcg.web.cern.ch/LCG/tdr
[8]Gordon Bell, Jim Gray, Alex Szalay, "Petascale Computational Systems: Balanced Cyberinfrastructure in a Data-Centric World".[online]
Available:
http://research.microsoft.com/~gray/papers/Petascale%20computational%20systems.pdf
[9] OSG website [online].
Available: http://www.opensciencegrid.org/
[10] EGEE website [online]. Available:
http://www.eu-egee.org/
[11] Antonio Delgado Peris, Patricia Méndez Lorenzo, Flavia Donno, Andrea Sciabà, Simone Campana, Roberto Santinelli, gLite-3 User Guide. CERN-LCG-GDEIS-722398, 17 January 2007. Available:
https://edms.cern.ch/file/722398//gLite-3-UserGuide.pdf
[12] The Virtual Organization Membership Service website [online]. Available: http://hep-project-grid-scg.web.cern.ch/hep-project-grid-scg/voms.html
[13] Heinz Stockinger, Flavia Donno, "Data Location Interface for the Workload Management System", November, 12 2004 [online]. Available:
http://cmsdoc.cern.ch/cms/grid/docs/DataLocationInterface.pdf
[14] Castor website [online]. Available:
http://castor.web.cern.ch/castor/
[15] dCache website [online]. Available:
http://www.dcache.org/
[16] DESY website [online]. Available:
http://www.desy.de/html/home/index.html
[17] FERMILAB website [online]. Available:
http://www.fnal.gov/
[18] SRM website [online]. Available:
http://sdm.lbl.gov/srm-wg/
[19] Globus Project, GridFTP Universal Data Transfer for the Grid, white paper September 5, 2000 [online].
Available:
http://www.globus.org/toolkit/docs/2.4/datagrid/deliverables/C2WPdraft3.pdf
[20] Xrootd website [online]. Available:
http://xrootd.slac.stanford.edu/

[21] ProC documentation [online]. Available:
http://www.dbasupport.com/oracle/ora10g/proC101.shtml
[22] OSG Consortium presentation, the gLite Middleware distribution, 21-23 August 2006. Slide 13.
[23] LCG-util API [online]. Available: http://grid-deployment.web.cern.ch/grid-deployment/documentation/LFC_DPM/lcg_util/html/
[24] GFAL API [online]. Available:
http://grid-deployment.web.cern.ch/grid-deployment/documentation/LFC_DPM/gfal/html/
[25] SOAP webiste [online]. Available :
http://www.w3.org/TR/soap/
[26] The information system [online]. Available:
Antonio Delgado Peris, Patricia Méndez Lorenzo, Flavia Donno, Andrea Sciabà, Simone Campana, Roberto Santinelli, gLite-3 User Guide. CERN-LCG-GDEIS-722398, 17 January 2007 Page(s): 51-73
Available: https://edms.cern.ch/file/722398//gLite-3-UserGuide.pdf
[27] DICOM website [online]. Available:
http://medical.nema.org/
[28] AMGA website [online]. Available:
http://amga.web.cern.ch/amga/
[29] HYDRA website [online]. Available:
https://twiki.cern.ch/twiki/bin/view/EGEE/DMEDS
[30] FTS website [online]. Available:
https://twiki.cern.ch/twiki/bin/view/EGEE/FTS
[31] J-P, Baud, J. Casey, S. Lemaitre; C. Nicholson, "Performance analysis of a file catalog for the LHC computing". 14th IEEE International Symposium on Volume, Issue, 24-27 July 2005 Page(s): 91 – 99
[32] SRM tests website [online]. Available:
http://datagrid.lbl.gov/