DIRAC File Replica and Metadata Catalog

Journal of Physics: Conference Series **396** (2012) 032108 doi:10.1088/1742-6596/396/3/032108

DIRAC File Replica and Metadata Catalog

A.Tsaregorodtsev[1], **S.Poss**[2]

[1] Centre de Physique des Particules de Marseille, 163 Avenue de Luminy Case 902 13288 Marseille, France
[2] CERN CH-1211 Genève 23, Switzerland

On behalf of the DIRAC Project

E-mail: atsareg@in2p3.fr

Abstract. File replica and metadata catalogs are essential parts of any distributed data management system, which are largely determining its functionality and performance. A new File Catalog (DFC) was developed in the framework of the DIRAC Project that combines both replica and metadata catalog functionality. The DFC design is based on the practical experience with the data management system of the LHCb Collaboration. It is optimized for the most common patterns of the catalog usage in order to achieve maximum performance from the user perspective. The DFC supports bulk operations for replica queries and allows quick analysis of the storage usage globally and for each Storage Element separately. It supports flexible ACL rules with plug-ins for various policies that can be adopted by a particular community. The DFC catalog allows to store various types of metadata associated with files and directories and to perform efficient queries for the data based on complex metadata combinations. Definition of file ancestor-descendent relation chains is also possible. The DFC catalog is implemented in the general DIRAC distributed computing framework following the standard grid security architecture. In this paper we describe the design of the DFC and its implementation details. The performance measurements are compared with other grid file catalog implementations. The experience of the DFC Catalog usage in the CLIC detector project are discussed.

1. Introduction

Management of large volumes of data in a distributed computing environment represents an enormous difficulty especially for the High Energy Physics experiments. Each year, the LHC experiments are collecting multiple tens of Petabytes of data coming from the detectors but also from the extensive Monte-Carlo modelling of the experimental setups and underlying physics theories. This requires high performance storage systems with high I/O capacities deployed in multiple sites. However, keeping track of all the files and their physical replicas counting hundreds of millions of copies is a huge challenge. It requires versatile Replica Catalogs powerful enough to stand high rates of queries coming from the clients running anywhere on the computing grid.

Another important function of the Catalogs is providing descriptions of the data to help users to find the subsets suitable for their analysis. This is done with the help of Metadata Catalogs that keep tags for each piece of data in a way optimized for efficient data searches.

In the already long history of the grid middleware evolution there were several projects to provide a scalable Replica Catalog suitable for the distributed computing environments. The Replica Location Service (RLS) and FiReMan Catalog [1] were the early attempts that did not fulfil the requirements of the HEP experiments in terms of scalability and reliability. The Alice Collaboration developed their grid middleware called AliEn [2] with the File Catalog being its central component. The AliEn File Catalog is still successfully used by the experiment with the real LHC data flow. The advantage of the AliEn File Catalog is that it has all the features of the Replica and Metadata Catalog at the same time. In the gLite middleware stack the Replica and Metadata Catalogs are represented by two distinct components – LCG File Catalog (LFC) [3] and AMGA [4] services respectively.
The DIRAC Project was initially developed for the LHCb Collaboration to form the basis of its computing model [5]. DIRAC was initially focused on providing an efficient Workload Management System whereas the LHCb Data Management tasks were accomplished using third party services. After a careful study of various options with respect to the functionality and performance, the LFC was chosen as a Replica Catalog. As for the Metadata Catalog, LHCb developed a special service called BookkeepingDB, which also plays the role of the Data Provenance Database [6].
Multiple user communities other than LHCb now use the DIRAC middleware. These communities are using the resources not only coming from grid infrastructures but also from local computing clusters or clouds. Therefore, it turned out to be necessary to add a Replica and a Metadata Catalogs to the list of DIRAC components to provide a complete set. As a result, the DIRAC File Catalog (DFC) was developed providing all the required catalog functionalities in a single service. The DFC design is based on a careful study of the similar services experience taking into account advantages and drawbacks of already existing projects. Special attention was paid to the performance optimization especially for the queries returning large numbers of data records.
In this paper we discuss the main features and performance characteristics of the DFC service. The general DIRAC Catalog Framework is presented in Section 2. Replica and Metadata parts of the DFC are described in Sections 3 and 4 respectively and the service interfaces are discussed in Section 5. Sections 6 and 7 are giving details on the DFC performance evaluation and examples of its usage followed by Conclusions.

2. Catalog framework

DIRAC is providing access to various types of resources available from grid infrastructures but also from other sources. Therefore, the DIRAC software framework is based on a modular architecture, which gives access to different third party services by means of specialized plug-ins or Clients. The Clients encapsulate specific access methods for the services while exposing a standard interface defined by the framework. In this way, DIRAC is capable to work with multiple File Catalogs of different types simultaneously, providing an abstraction layer with a specific implementation for each particular service (figure 1).

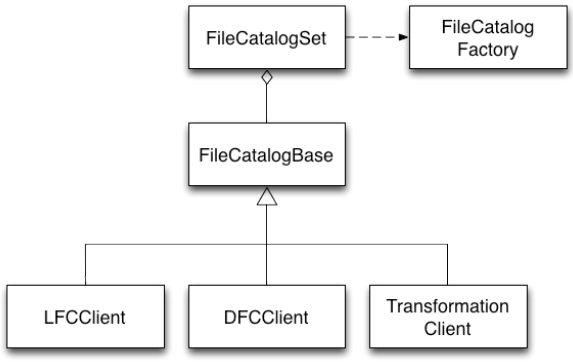

Figure 1. DIRAC framework modular architecture

This feature of the DIRAC Framework turned out to be very useful. For example, in the early stages of the LHCb Computing Project, it allowed to use simultaneously the AliEn and the LFC catalogs to make a thorough comparison of their performance. The possibility to use both LFC and DFC Catalogs in the same DIRAC installation helps users to easily migrate their tools and data from one service to another to optimize the computing models. The same principle is applied to other services, for example to various data servers. This allows in the same DIRAC installation to aggregate together storage resources coming from different infrastructures transparently to their users. Users just see

logical entities like File Catalogs or Storage Elements; the differences are completely hidden by the framework (figure 2).

An example of other Catalog-like service is the DIRAC Transformation Service. It implements the same interface that allows its use in parallel with other catalogs. This service is used in DIRAC to define various operations that are triggered by the new data registration. For example, this system is used in the LHCb Data Production Service to automatically create and submit data processing jobs as soon as the new data become available or to schedule automatic data replication.

A special care is taken to keep the contents of all the catalogs used in parallel well synchronized. This is achieved by using the FileCatalog client container object, which updates all the configured catalogs in parallel. In case of failures, special requests to accomplish the failed operations are submitted to the DIRAC Request Management System where they are executed as soon as the target service becomes available again.

The DFC is implemented using the general DIRAC framework called DISET (figure 3) [7]. The service access is done with an efficient secure protocol compliant with the grid security standards. The protocol allows also defining access rules for various DFC functionalities based on the user roles. The clients are getting the service access details from the common Configuration Service. The DFC behaviour is monitored using the standard DIRAC Monitoring service. The errors in the service functioning are reported to the SystemLogging service. A MySQL database is used to store all the DFC contents.

The reuse of the DIRAC framework components was an important advantage that allowed developing the DFC service very quickly concentrating on its specific business logic. The service deployment and maintenance is done with the standard DIRAC tools, which helps a lot the installation administrators.

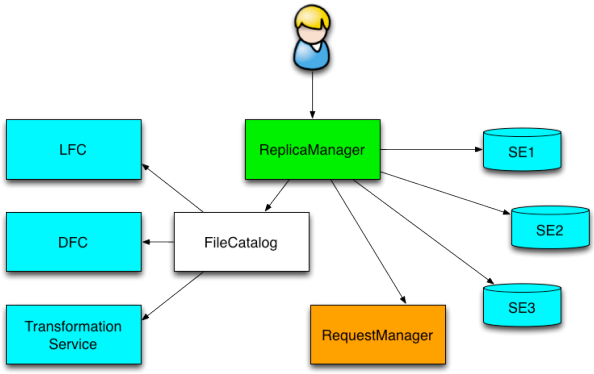

Figure 2. Aggregation of various data management services in a single DIRAC installation

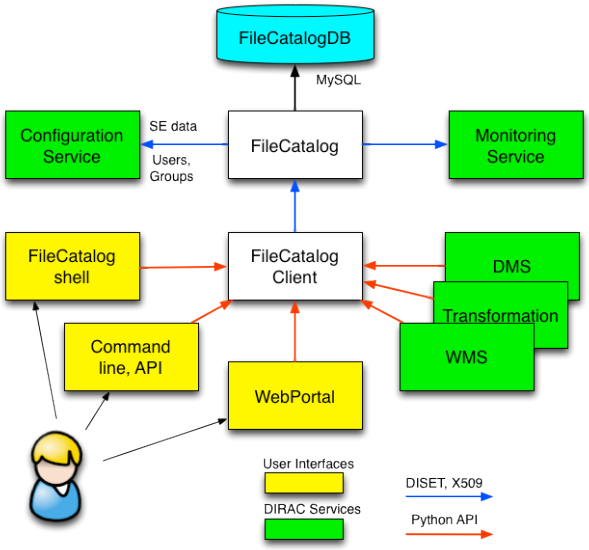

Figure 3. The DFC is built in the general DIRAC DISET software framework

3. Replica Catalog

The file data in DFC is stored in a hierarchical directory structure, which helps in the optimization of the file and replica lookups. All the standard file metadata commonly found in file systems are provided. Extra fields for file status information are provided that can be used according to specific needs of the given application.

A peculiarity of the DFC compared to the LFC, for example, is that the Logical File Name (LFN) is required to be unique for a given file. The file GUIDs are also supported for those applications that require them, the GUIDs uniqueness can be switched on and off in the configuration options.

The access control is implemented as a number of plug-ins that can be chosen according to the needs of a given user community. Plug-ins ranging from *NoSecurity* to *PosixSecurity* types are provided. If necessary other plug-ins can be implemented for communities with specific needs and included in the

DFC distribution. There is a possibility to define a *GlobalReadAccess* mode when the access control checks are skipped for any read-only operation. This allows increasing considerably the catalog performance in case the community data does not have read access limitations, which is usually the case of the HEP experiments.
The replica information can be stored in 2 different ways. In the first case, the full Physical File Names (PFN) are stored as complete access URLs. In the second case, the DFC exploits a convention that the PFNs are containing the corresponding LFN as its trailing part. This is a very common convention usually referred to as a "trivial file catalog". It allows for easy establishment of correspondence between the physical files and entries in the catalogs, which in turn helps a lot in checking the data integrity between the catalog and the storage. In this second case, there is no need to store full PFNs in the catalog as part of the replica information; it is sufficient just to store the identifier of the Storage Element (SE) containing the replica. The full PFN can be constructed on the fly at the moment of the replica look-up by the clients (figure 4).

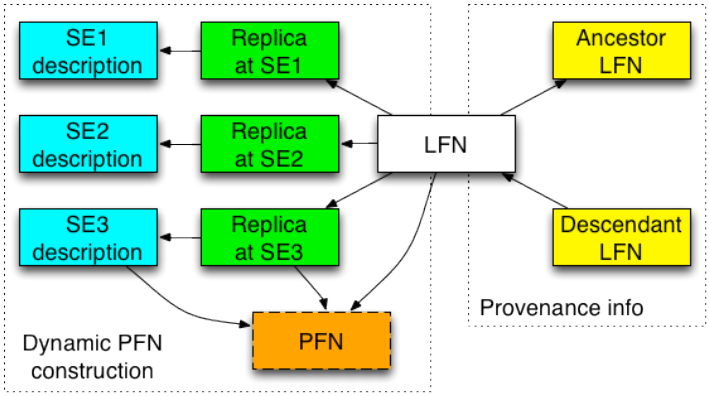

Figure 4. DFC Replica Catalog schema

There are several advantages in this approach. First, the amount of replica data to be stored in the database is considerably reduced which in turn reduces the overall footprint of the database files and increases its performance. Second, in case the SE description changes, for example, its access point, it is enough to change the SE description in the DIRAC Configuration Service to continue receiving correct PFNs.
The experience with using File Catalogs shows the importance of tools that allow preparing in real time reports on storage usage by the users, groups, files in a given directory, etc. This is necessary, for example, in order to efficiently manage user storage quotas. The information of the users exceeding their quotas must be immediately available to trigger actions according to community policies. The DFC has a special support for the storage usage reports by maintaining special internal structures updated each time when a new replica is registered in the catalog.
Another feature requested based on the experience in the HEP experiments is to store and provide information about ancestor-descendent relations between files. DFC supports this basic data provenance information. The ancestors of the given file can be specified at the moment of the new file registration or later. The ancestors or descendants can be looked up taking into account the "generation", e.g. parent files, grandparents, etc.

4. Metadata Catalog

As was already mentioned above, DFC is a Replica and Metadata Catalog at the same time. It allows defining arbitrary user metadata associated with its files and directories as arbitrary key-value pairs. The catalog administrators can declare certain keys, or tags, as indexes. The indexed tags can be used in the file lookup operations. The following rules apply (figure 5):

- Subdirectories are inheriting the metadata of their parent directories. The inherited tag values can not be overwritten;
- Files are inheriting the metadata of their directories

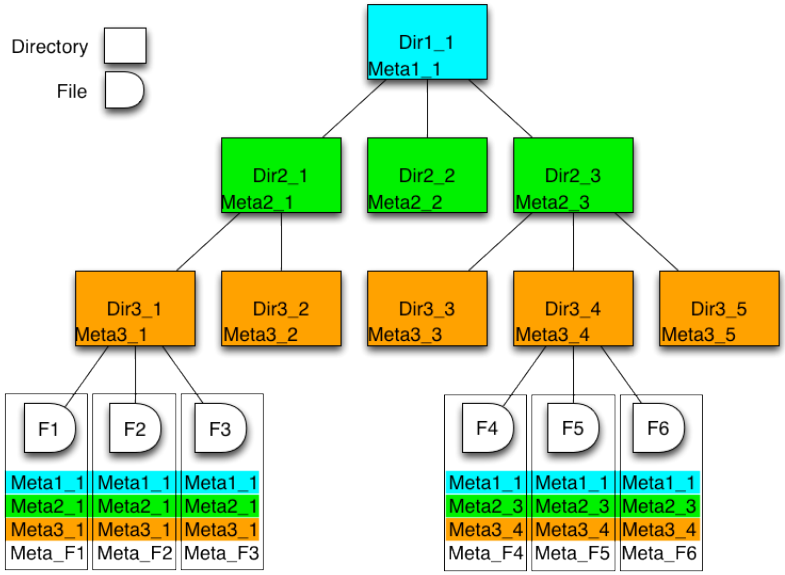

Figure 5. DFC Metadata inheritance rules

After the metadata is stored, the files can be looked up with queries like the following (example, using the DFC console):

```
FC> find /lhcb/mc Year>2010 EventType=mbias Version=v20r1,v21r2
```

The value types include numbers, strings or time stamps. The search can be limited to a given subdirectory.

Storing the metadata in the same directory hierarchy as the file logical name space allows keeping the same schema for both replica and metadata information which is much less confusing for the users compared to the use, for example, of LFC and AMGA as complementary services. The hierarchical metadata organization allows for query efficiency optimizations. For files that share the same metadata there is no need to duplicate it.

5. DFC interfaces

The convenience of the File Catalog usage largely depends on its interfaces. The DFC is providing several user interfaces suitable for different usage patterns.

The DFC console gives access to the DFC functionality in a similar way as the normal unix shell. Usual file system commands can be used, e.g. *ls, cd, chmod*, etc. Several specific DFC commands are also available to manage the file metadata, make storage usage reports, maintain the DFC service, etc. The console is suitable for occasional users to browse the file name space.

The DFC Python API is a programming interface that allows creating any custom commands or even small applications working on the DFC data. This is the main interface for the developers and power users.

Finally, the emphasis is made on the DFC Web interface (figure 6). It provides the DFC file browser with a look and feel similar to the desktop applications. All the file information can be looked up by this interface. In addition, certain operations, e.g. file download or replication can be also performed.

The DFC Web interface is built in the DIRAC general Web Portal framework and provides a secure access based on the user certificates loaded into the web browser.

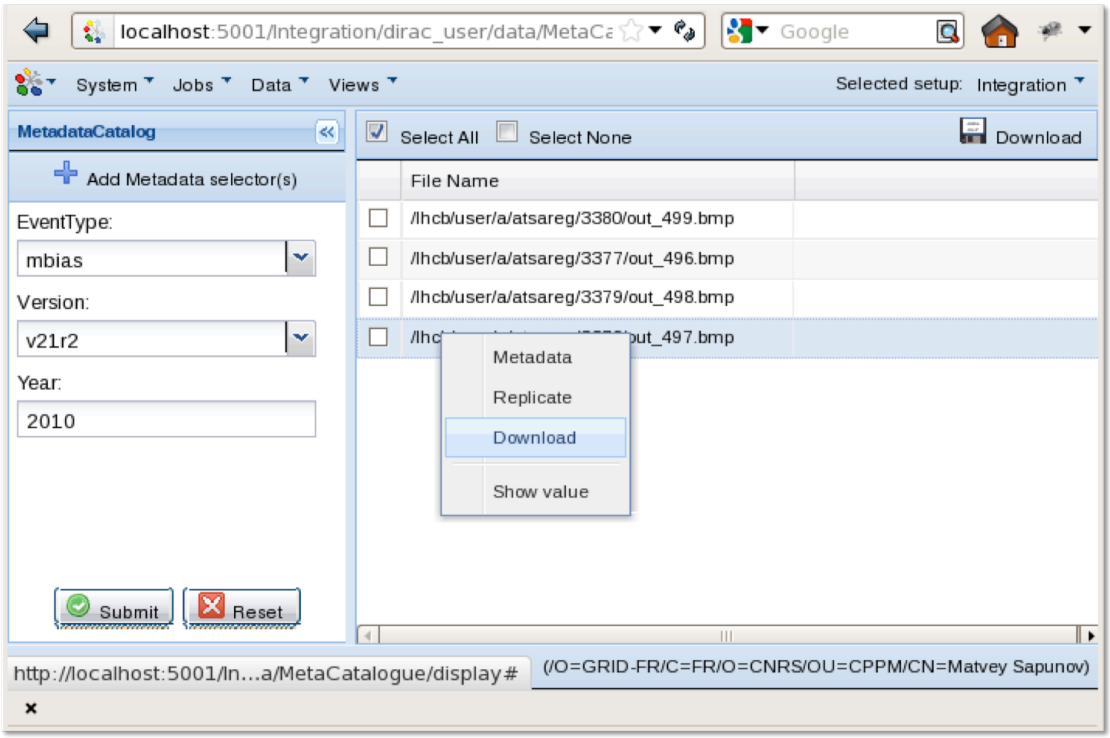

Figure 6. DFC Web Interface example

6. Performance tests

Extensive DFC performance tests were carried out to study the behaviour of the service and of the underlying database. The final goal is to increase the catalog efficiency especially for the heavy bulk operations often needed in the HEP experiments handling vast data volumes. Some results are present in the following.

For the tests of the Replica Catalog part, DFC was populated with the whole set of the LHCb data with the following volume: ~9M files with 12M replicas in ~1.2M directories. The DFC service together with its MySQL database was installed on a mid-range server with 8 CPU cores and 16GB of RAM. The service was configured to use 2 CPU cores only.

The tests consisted in querying the replica information for a random list of LFNs of a given size. The list size varied from 10 to 100K files. Each query has its own list to exclude the effect of the database caching the results of the previous queries. The catalog was configured with the *GlobalReadFlag*, so that no access control checks were performed. As was mentioned earlier, this is the main operational mode for the HEP experiments. The results of the tests are presented in figures 7 and 8.

Tests were performed with a varying number of clients simultaneously querying the catalog. Figure 7 presents the timing of the queries with 10 to 10K files in one query and with 1 to 20 simultaneous clients stressing the service. As can be seen in the figure, the catalog stands well multiple parallel clients up to 20 without any significant performance degradation. The response time dependency on the number of files in one query is linear which confirms excellent scalability properties of the catalog. Figure 8 shows the results of similar queries with one single client but with the number of files up one order of magnitude. Querying replica data for 100K files at once takes just 10 sec. With this respect the performance is largely superior to the one of LFC [4]. It is important to note that a precise direct comparison of the performance of the two catalogs is difficult to interpret due to considerably different

server settings of the two catalog services. For example, the LFC database backend was provided by a dedicated ORACLE service running in a separate host, whereas the DFC MySQL database backend was running in the same host as the catalog service itself.

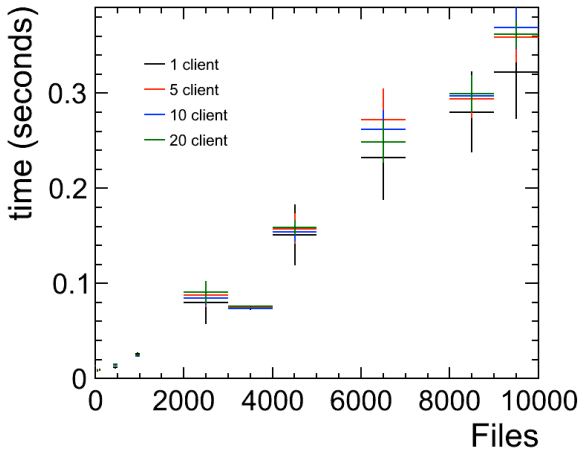

Figure 7. Time to get replica information for different data set sizes and for different number of concurrent client

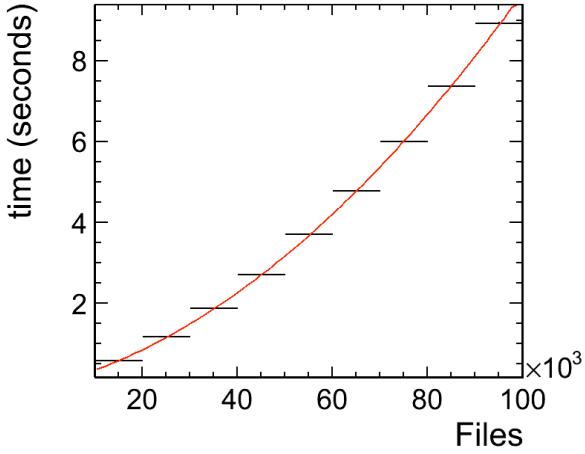

Figure 8. Time to get replica information for large data set sizes with only one client at a time

Comprehensive performance tests of the Metadata Catalog part of the DFC services were carried out by the BES experiment as part of their evaluation of the DIRAC solution compared to the AMGA based service [12]. The conclusion is that the performance of both services is comparable taking into account considerably different schemas used to store the metadata in both cases.

7. Usage examples

7.1. ILC, CLIC

In 2011, DIRAC was used to produce on the grid the Monte Carlo data required for the CLIC physics and detectors Conceptual Design Report [9,10]. Among all the available solutions, it was the one that proved to be easy to install and maintain, and that provided the necessary utilities to compensate the lack of manpower. Among the required features was an easy to use File Catalog. The DFC functionalities used are Replica and Metadata catalogs, ancestor-daughter relationships, Storage Element space usage, users' quota monitoring.

The Metadata Catalog aspect was the most crucial, as it allowed us to setup a data driven production framework using the Transformation System, making the whole mass production fairly automatic, and reducing the amount of manual bookkeeping. The ancestor-daughter functionality allowed us to store the relevant information for files and from their daughters look those up. For example, the luminosity of a given generator file was stored, and was read back for the reconstructed file analysis. It also made cleanup easy: the generation of some files was done with an error; therefore all files produced with that error had to be removed. Without the ancestor-daughter functionality, this operation would have been extremely difficult, if not impossible. As for the Storage Element space monitoring, it gave us fast control of space usage, giving us the tools to prepare resource requests for future needs.

The CLIC detector project successfully managed to produce and store more than 3 million files in 1.5 years with DIRAC and the DFC, for production and users' activities. Those files are used daily for analysis on the GRID, using both the CLI and the python API to access them. This success led to the

adoption of (ILC)DIRAC and the DFC by the SID detector community [11] for the Detailed Baseline Detector document that will be produced during 2012.

7.2. BES

The BES Collaboration at BEPC, Beijing, China is now preparing for the phase III of the experiment. This is the occasion to review the BES Computing Model to cope with the higher data acquisition rates foreseen for this stage. Therefore, the Data Management system is revisited and evaluation of various solutions is being carried out [12]. The decision to base the BES III Computing Model on the use of the DIRAC middleware to manage the distributed computing resources is already taken. In particular the DFC was chosen as both Replica and Metadata Catalog after a thorough comparison with the LFC and AMGA solution respectively. It was shown that with comparable or superior performance, DFC offers the functionality more suitable for the BES III Computing Model.
It is worth to mention that during this extensive DFC evaluation work by the BES team, a lot of extremely useful feedback was communicated to the DIRAC developers. This helped a lot in the software optimization as well as in adding extra functions on request of the BES users.

8. Conclusions and outlook

The necessity to develop a full-featured File Catalog service for DIRAC became clear when the goal of providing a complete middleware stack was adopted by the project. Indeed, having a comprehensive Data Management system with all the components natively communicating with each other is a clear way to increase the system performance together with least possible maintenance efforts. The validity of this approach can be justified by another example of a successful system integrating all the Data Management tasks - the iRods project [13]. It offers a complex solution for data cataloguing, storing and replication where users can perform all the necessary operations staying in the same environment. The project also features a modular architecture allowing users to provide custom plugins for specific functionalities. This, in particular, opens opportunities for interoperability of iRods and DIRAC services.
New user communities like HEP experiments are looking for a solution for managing even higher data volumes than the ones of the LHC experiments. Optimization of the performance but also reducing the operational and maintenance costs is the issue. Therefore, comprehensive systems offering solutions for all the computing tasks within the same software framework is a clear advantage.
The DIRAC File Catalog was relatively easy to develop as it reuses heavily all the framework components of the DIRAC middleware. Now that it is used by running experiments, the emphasis of further developments is put on the performance and reliability of the service. Working towards precise requirements of the DFC users allows concentrating on the most actual problems and feature requests.

References

[1] G.Munro, B.Koblitz, Performance comparison of the LCG2 and gLite file catalogues, *Nucl. Instrum. Methods Phys. Res., A* 559 (2006) 48-52;
http://cern.ch/egee
[2] P. Saiz, L. Aphecetche, P. Buncic, R. Piskac, J.-E.Revsbech and V. Sego, AliEn – ALICE environment on the GRID, *Nucl. Instrum. and Methods Phys. Res.,* A 502 (2003) 437-440;
http://alien2.cern.ch
[3] J.-P. Baud, J. Casey, S. Lemaitre, C. Nicholson, "Performance analysis of a file catalog for the LHC computing grid", in Proceedings of the 14th IEEE International Symposium on High Performance Distributed Computing, 2005, pp. 91-99
[4] N.Santos, B.Koblitz, Distributed Metadata with the AMGA Metadata Catalog, in Workshop on Next-Generation Distributed Data Management, HPDC-15, Paris, France, June 2006
[5] A Tsaregorodtsev et al, DIRAC3 : The New Generation of the LHCb Grid Software, 2010 *J. Phys.: Conf. Ser.*, **219** 062029;
http://diracgrid.org

[6] Ph.Charpentier, The LHCb Data Management System, 2012 Int. Conf. on Computing in High Energy and Nuclear Physics (New York)
[7] A.Casajus, R.Graciani, DIRAC Distributed Secure Framework, 2010 *J. Phys.: Conf. Ser.* **219** 042033
[8] A.Casajus, M.Sapunov, DIRAC: Secure web user interface, 2010 *J. Phys.: Conf. Ser.* **219** 082004
[9] L. Linssen et al, Physics and Detectors at CLIC: CLIC Conceptual Design Report, arXiv:1202.5940. CERN-2012-003. ANL-HEP-TR-12-01. DESY-12-008. KEK-Report-2011-7
[10] S. Poss, DIRAC File Catalog Structure and Content in preparation of the CLIC CDR, LCD-OPEN-2012-009, https://edms.cern.ch/document/1217804
[11] https://silicondetector.org/display/SiD/home
[12] C.Nicholson et al, File and metadata management for BESIII distributed computing, 2012 Int. Conf. on Computing in High Energy and Nuclear Physics (New York)
[13] http://www.irods.org