The Rucio File Catalog in DIRAC implemented for Belle II

Cédric Serfon[1,*], *John Steven* De Stefano Jr[1], *Michel* Hernández Villanueva[2], *Hironori* Ito[1], *Yuji* Kato[3], *Paul* Laycock[1], *Ruslan* Mashinistov[1], *Hideki* Miyake[4], and *Ikuo* Ueda[4]

[1]Brookhaven National Laboratory, Upton, NY, USA
[2]University of Mississippi, MS, USA
[3]KMI - Nagoya University, Nagoya, Japan
[4]High Energy Accelerator Research Organization (KEK), Japan

Abstract. DIRAC and Rucio are two standard pieces of software widely used in the HEP domain. DIRAC provides Workload and Data Management functionalities, among other things, while Rucio is a dedicated, advanced Distributed Data Management system. Many communities that already use DIRAC have expressed their interest in using DIRAC for Workload Management in combination with Rucio for Data Management. In this paper, we describe the integration of the Rucio File Catalog into DIRAC that was initially developed for the Belle II collaboration.

1 Introduction

DIRAC (Distributed Infrastructure with Remote Agent Control) [1] is a piece of software initially developed for the LHCb collaboration [2]. It is designed as "interware" as it provides a complete solution for managing distributed resources. Among the various functionalities of DIRAC, one can find a Workload Management System and a Distributed Data Management system. On the other hand, Rucio [3] is a Distributed Data Management system. Initially developed by the ATLAS collaboration [4], Rucio has quickly gained popularity outside ATLAS due to its advanced features. Both DIRAC and Rucio [5] are now used by a large community beyond their initial collaboration. In recent years, some communities have expressed interest in using DIRAC for Workload Management in combination with Rucio for Distributed Data Management. The Belle II collaboration [6] developed a Rucio File Catalog (RFC) plugin for its extension of DIRAC called BelleDirac [7]. In section 2, this article details some of the differences between the RFC plugin and the file catalog plugins already implemented in DIRAC. Section 3 details all the methods that have been implemented and presents differences with respect to the other catalogs.

2 DIRAC and File Catalogs

For experiments using distributed computing resources it is often the case that there will be multiple copies, or replicas, of files across the different computing sites. An obvious example

*e-mail: cedric.serfon@cern.ch

is that all experiments ensure that there is a copy of the precious raw detector data, but there may be multiple copies of processed outputs to allow those data to be used at multiple sites at the same time. File catalogs provide a means to provide coherent access to file replicas. Each file in the catalog has a Logical File Name (LFN) and each LFN can have a list of associated Physical File Names (PFN) that correspond to the physical copies of the files at different sites. If an application or a user wants to locate a particular LFN, then they simply query the catalog to get the list of associated file replicas.

2.1 File catalogs supported by DIRAC

DIRAC provides a File Catalog interface to the actual file catalog service, more details about that interface are described in section 3. To interact with a particular file catalog service, a plugin that implements the methods defined in the File Catalog interface is needed. Before the implementation of the RFC plugin, two file catalogs were supported by DIRAC:

- The first one is an external catalog called the LCG (LHC Computing Grid) File Catalog (LFC) [8]. It is a hierarchical catalog that allows one to organise the files into a directory structure. The LFC only contains the file replica information and very limited metadata including the checksum and the file size.
- The DIRAC File Catalog (DFC) is another catalog internal to DIRAC. In contrast to the LFC, the DIRAC File Catalog combines both replica and metadata functionality [9].

2.2 Differences between Rucio and the LFC

The Belle II collaboration previously used the LFC and wanted to maintain the same functionality and behaviour. The RFC plugin was designed to meet that goal with some slight differences listed below:

- Whereas the LFC is inherently hierarchical, Rucio uses by default a flat namespace with files contained in datasets which are a collection of files, but the datasets are not connected to one another. Rucio has another type of data structure called containers which are a collection of datasets and/or containers. Containers cannot contain files. Using containers, it is then possible to reproduce the directory structure of the LFC (see figure 1) with the constraint that directories cannot contain a mixture of files and directories. This same constraint was introduced into Belle II Software to prevent users from encountering this feature.
- Rucio has the concept of scope : The scope is a way to partition the Rucio namespace and to apply different policies, permissions, etc. Every Data IDentifier (DID) which represents a file, dataset, or a container, has a scope and the DID name must be unique within the scope. Since there is no concept of scope in the LFC, it needs to be hidden from end-users and applications. To achieve this, a deterministic function uses the LFN to associate each LFN to a unique scope in a transparent way.
- Another big difference is related to the Rucio concept of replication rules. The replication rules are a way to describe how a DID must be replicated on a list of Rucio Storage Elements (RSE). If a rule is created for a particular DID on certain RSE, Rucio will ensure that the rule is fulfilled either by locking the DID at the specified RSE if it is already there, or by transferring and then locking it to the specified RSE. This difference has important consequences for deletion as explained in subsection 3.3.
- Finally there is no concept of fine-grained Access Control List (ACL) on files or directories in Rucio, contrary to the LFC. In Rucio, permissions are managed at the scope level, and all of the files in the same scope have the same permissions.

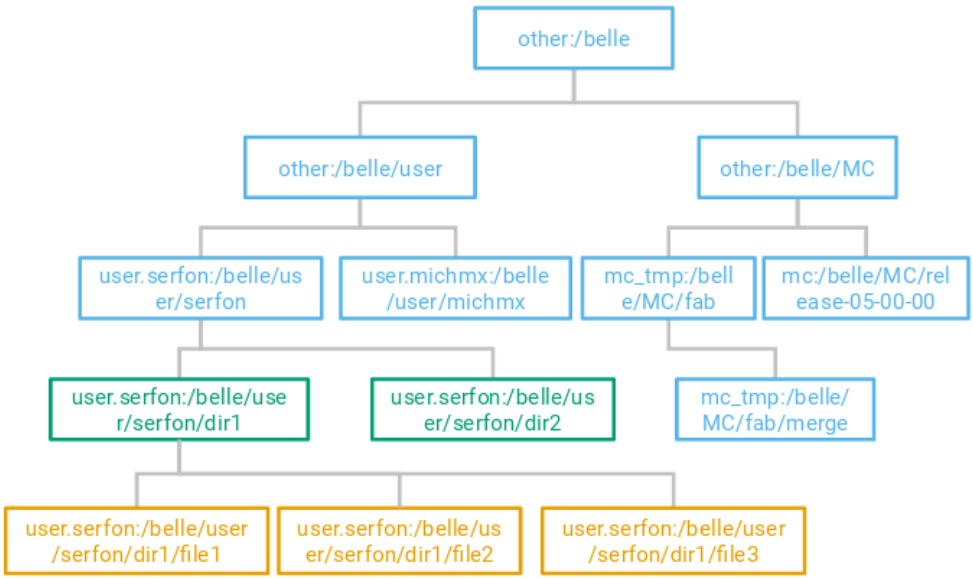

Figure 1. Schema showing how the data are structured in Rucio to reproduce the LFC hierarchy. The orange boxes represent files, bluish green boxes represent datasets that can only contain files, and sky blue boxes are containers that can only contain datasets and other containers. The first part of the name before the colon represents the scope and is associated uniquely to the LFN which follows the colon.

2.3 Differences between Rucio and the DFC

All the differences listed in section 2.2 also apply to the RFC compared to the DFC. In addition, similarly to the DFC, Rucio supports generic metadata. In the DFC, subdirectories inherit the metadata of their parent directories and files inherit the metadata of their directories. In Rucio it is possible to set any metadata to any DID and children do not inherit metadata from their parents.

3 Rucio file catalog plugin functionality

DIRAC provides a File Catalog interface that allows users or any other (Belle)DIRAC component to interact with the file catalog service. It provides several different methods which can be categorised as:

- Read methods: To list the content of a directory, to get stats about files or directories, to get the list of PFN (replicas) associated with one LFN, and many other methods.
- Write methods: To create new files, and to create new replicas.
- Delete methods: To delete file replicas (remove a PFN associated to an LFN), or delete a file completely, i.e. remove an LFN from the catalog.

To implement a new catalog plugin, all these methods need to be implemented. One potential obstacle to implementing a Rucio file catalog plugin is due to the important concept of scope. As mentioned in the previous section, the scope is an unknown concept both for the LFC and DFC. Therefore the scope cannot be passed to the catalog method and needs to be extracted

directly from the LFN. This is done with the help of a deterministic function that maps each LFN to one and only one scope.

The current implementation of the RFC plugin only contains the replica methods that are available in both the LFC and the DFC, but not the metadata methods that are unique to the DFC. In order to use the RFC plugin, the Rucio clients need to be installed on the DIRAC server. This can be done using the Python package manager pip. In the future, it is foreseen to include the Rucio clients into DIRACOS [10]. To setup the RFC plugin, a few environment variables need to be defined in a RucioFileCatalog section of the DIRAC Configuration system as shown in figure 2.

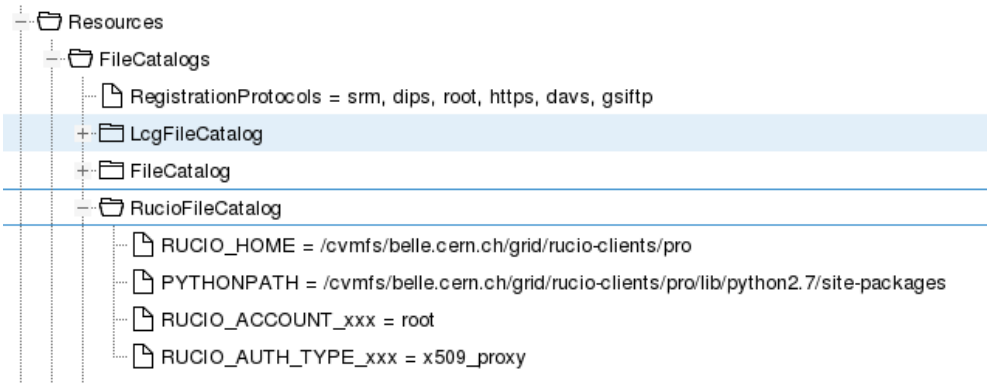

Figure 2. Snapshot of the DIRAC Configuration Service used by Belle II. The RucioFileCatalog section is a subsection of Resources/FileCatalogs that contains a few environment variables that are used to setup the Rucio client.

Before starting to use the RFC plugin, the Storage Elements registered in the Resources section of the DIRAC Configuration, as well as the users registered in the Registry section, need to be created on the Rucio server. In the Belle II case this is done automatically by a new DIRAC agent called the `RucioSynchronizer` that creates the RSE and their associated protocols, as well as the user accounts.

3.1 Read methods

The following read methods have been implemented: `getReplicas`, `listDirectory`, `getFileMetadata`, `getFileSize`, `isDirectory`, `isFile`, `getDirectorySize`. Using the mapping between datasets/containers and directories described in 2, all these methods have the same behaviour as the ones in the LFC plugin. They use bulk queries to the Rucio server which allows for faster response times in case more than one file is specified. This is particularly important considering that the DIRAC server and the Rucio server can be relatively far away, e.g. for Belle II there is a distance of more than 10000 kilometers between the DIRAC servers and the Rucio servers which represents about 180 ms of Round trip Time.

3.2 Write methods

There are only two write methods: `addFile` and `addReplica`. For `addFile` a new atomic bulk method was added to Rucio. Adding new files involves many operations such as creating all of the parent directories if they do not exist, attaching files to the dataset, creating the file

replicas, and creating a replication rule for the dataset. The creation of the replication rule ensures that Rucio will be responsible for managing the dataset replica. The whole workflow is described in figure 3. Regarding `addReplica`, the method simply adds a file replica and a replication rule for this replica.

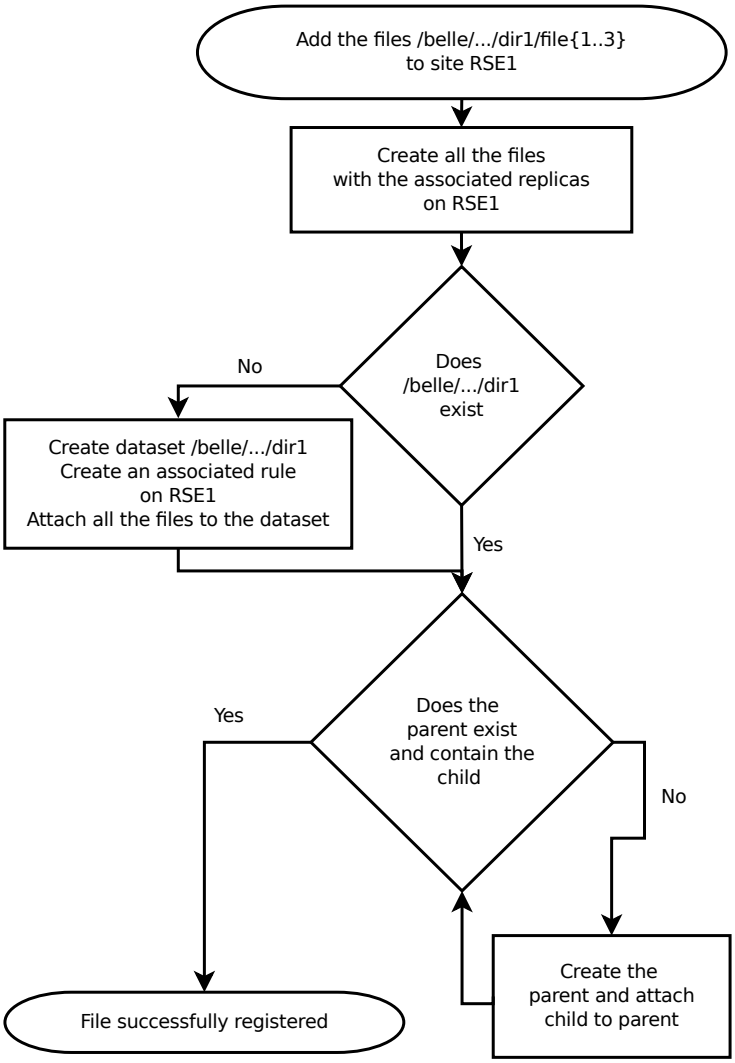

Figure 3. Diagram describing the workflow to add new files at a storage element. The entire procedure is handled in a new atomic bulk operation on the Rucio server side.

3.3 Delete methods

There are two delete methods: `removeReplica` and `removeFile`. Due to the very different concepts between Rucio and the LFC regarding deletion, their behaviour is different in the

RFC compared to the LFC methods. As explained in section 2, the DID in Rucio are locked at a specific site using replication rules. These rules prevent the deletion of a file replica, whereas this operation is a valid one for the LFC. Therefore if a file that belongs to a dataset has a replica on site A and this dataset has a replication rule on site A, it is impossible to remove the replica. Therefore the `removeReplica` method simply doesn't do anything in the RFC plugin. The `removeFile` method that removes a file from the namespace also has a different behaviour : in the LFC plugin, the command only succeeds if this file has no replicas, whereas in the RFC plugin, the file is removed even if replicas exist. Additionally, whereas in the LFC case the file deletion from storage is done synchronously, in the RFC the file is only logically removed from its parent directory synchronously, while the logical and physical deletion of the file itself and from its replicas is done asynchronously by a separate Rucio daemon.

3.4 Future work

The current implementation only supports the replica functionality and not the metadata functionality supported by the DFC. The RFC plugin could be extended to also support some metadata methods similarly to the DFC as Rucio provides the possibility to store generic metadata (key/value pairs).

In addition the plugin is currently only part of BelleDirac, but it has been made as experiment agnostic as possible. Only a few parts are Belle II specific: the function that extracts the scope from the LFN, the permission schema and one part of the `RucioSynchronizer` agent that sets some specific RSE attributes. Since many communities are interested to use Dirac and Rucio, work is currently ongoing to integrate the RFC implementation developed for Belle II to the generic DIRAC itself. For this integration, the Rucio clients have already been included in DIRACOS. One thing that is being focused on is to remove the need to install a Rucio configuration file and to rely only on the Dirac Configuration Service. The discussions and feedback from the DIRAC developers during this integration process helped to identify some possible improvements in the initial code.

4 Conclusion

A new Rucio File Catalog plugin has been developed to interface DIRAC with Rucio. The plugin is being used successfully by the Belle II collaboration in production. This paper summarized some of the differences between this new catalog and the other catalogs supported by DIRAC: the LCG File Catalog and the DIRAC File Catalog. The plugin allows to exploit fully the Rucio capabilities in particular the replication rules. Once included into the generic DIRAC, this new interface should help with the adoption of Rucio with DIRAC by more communities.

Acknowledgements

We would like to thanks the whole DIRAC developers team for all the fruitful discussions and feedback.
The work at Brookhaven National Laboratory is funded by the U.S. Department of Energy, Office of Science, High Energy Physics contract No. DE-SC0012704.

References

[1] Federico Stagni, Andrei Tsaregorodtsev, André Sailer and Christophe Haen, "The DIRAC interware: current, upcoming and planned capabilities and technologies", EPJ Web Conf. **245** 03035 (2020). doi: 10.1051/epjconf/202024503035

[2] A. A. Alves, Jr. *et al.* [LHCb], "The LHCb Detector at the LHC" JINST **3**, S08005 (2008) doi: 10.1088/1748-0221/3/08/S08005

[3] Martin Barisits *et al.*, "Rucio - Scientific data management," Comput. Softw. Big Sci. **3** (2019) no.1, 11 doi: 10.1007/s41781-019-0026-3

[4] G. Aad *et al.* [ATLAS Collaboration], "The ATLAS Experiment at the CERN Large Hadron Collider" JINST **3** S08003 (2008) doi: 10.1088/1748-0221/3/08/S08003

[5] Martin Barisits *et al.*, "Rucio beyond ATLAS: experiences from Belle II, CMS,DUNE, EISCAT3D, LIGO/VIRGO, SKA, XENON", EPJ Web Conf. **245** 03035 (2020). doi: 10.1051/epjconf/202024511006

[6] T. Abe *et al.*, KEK-REPORT-2010-1, arXiv:1011.0352 (2010)

[7] H. Miyake *et al.* [Belle-II computing group], "Belle II production system," J. Phys. Conf. Ser. **664** (2015) no.5, 052028 doi: 10.1088/1742-6596/664/5/052028

[8] J.P. Baud, J. Casey, S. Lemaitre and C. Nicholson, "Performance analysis of a file catalog for the LHC computing grid", HPDC-14. Proceedings. 14th IEEE International Symposium on High Performance Distributed Computing, 2005., Research Triangle Park, NC, 2005, pp. 91-99, doi: 10.1109/HPDC.2005.1520941

[9] A. Tsaregorodtsev *et al.* [DIRAC], "DIRAC file replica and metadata catalog", J. Phys. Conf. Ser. **396** (2012), 032108 doi: 10.1088/1742-6596/396/3/032108

[10] M. Petrič, C. Haen and B. Couturier, "DIRACOS: a cross platform solution for grid tools", EPJ Web Conf. **245** (2020), 03020 doi: 10.1051/epjconf/202024503020