

The Rucio File Catalog in Dirac implemented for Belle II

J. De Stefano, H. Ito, P. Laycock, R. Mashinistov, C. Serfon - **Brookhaven National Laboratory**

H. Miyake, I. Ueda - **KEK**

Y. Kato - **KMI**

M. Villanueva - **University of Mississippi**

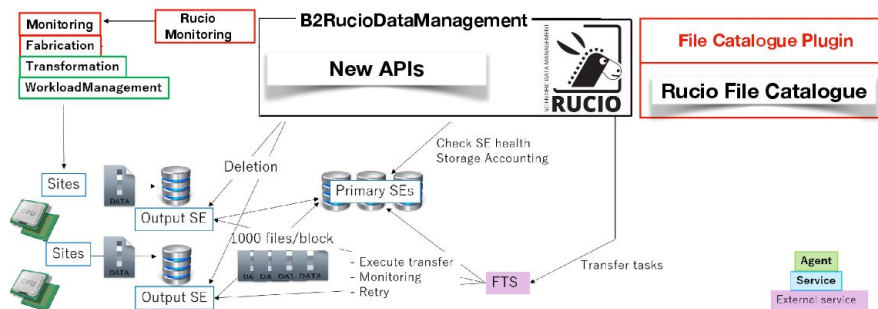
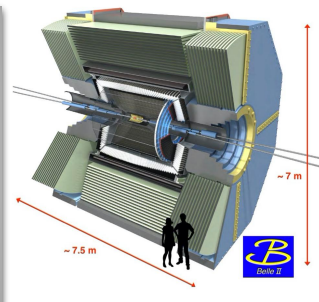


Introduction


- The Belle II collaboration use the distributed computing system called **BelleDIRAC**. It's an extension of **DIRAC** INTERWARE - a complete distributed computing system including Workload Management System and a Distributed Data Management (DDM) system
- Belle II originally used it's own DDM system to support data workflow
- Implementation of the Rucio File Catalog allowed to seamlessly migrate to **Rucio** - an advanced Distributed Data Management system
- More information on the integration of Rucio with BelleDIRAC in plenary talk [“Integration of Rucio in Belle II”](#)

Benefits with **Rucio**:

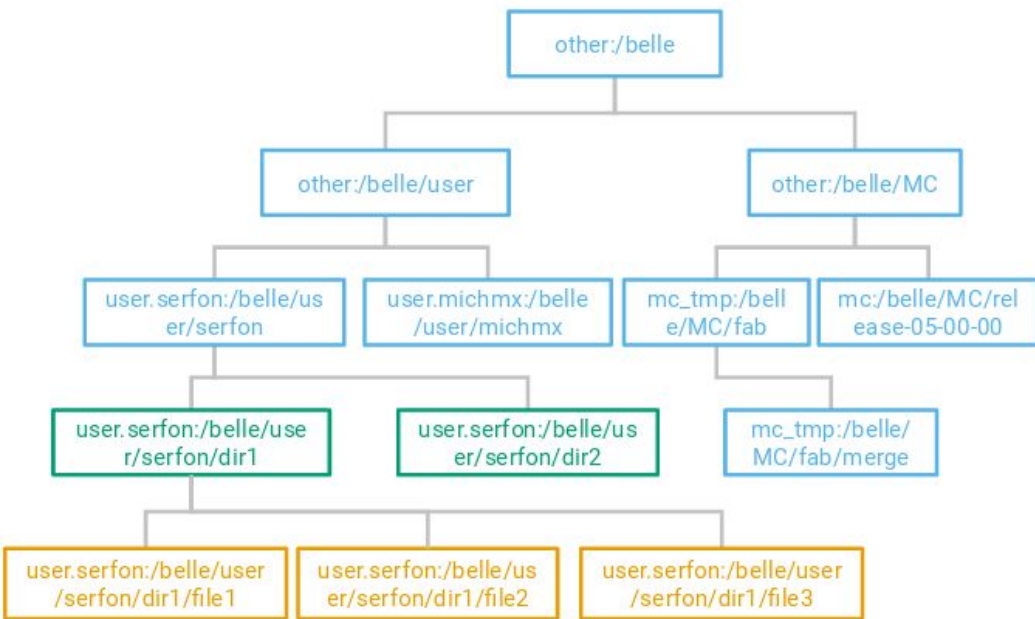
- Extremely scalable
 - Proved in ATLAS
- Policy-driven
 - Provides automation
- Provides monitoring tools



Dirac and File Catalogs

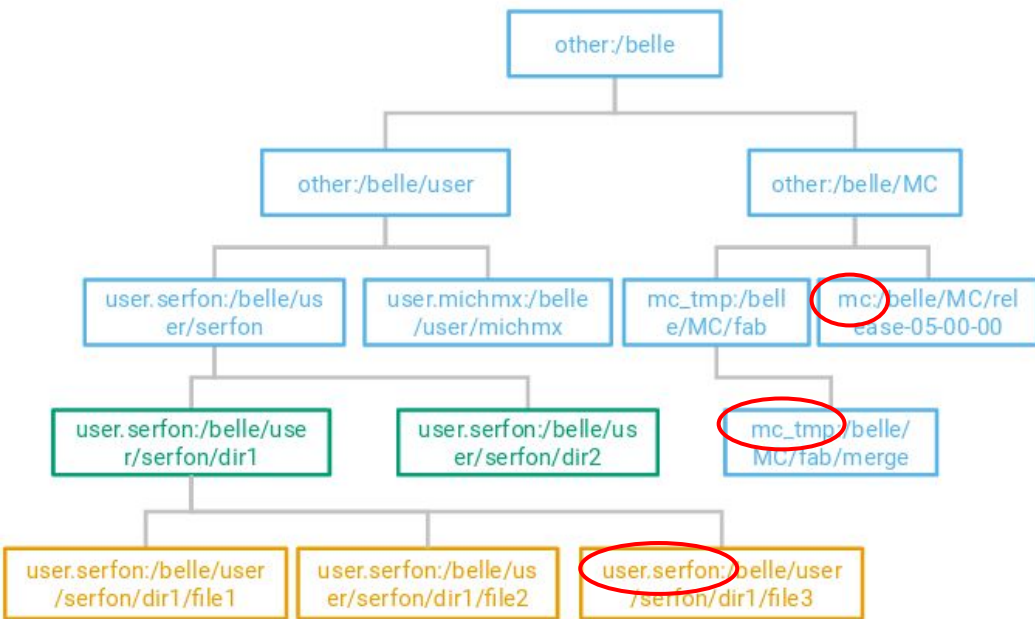
- A File Catalog maps a Logical File Name (LFN) to the Physical File Name (PFN) of the replicas at distributed computing sites
- To interact with a File Catalog,  provides a File Catalog interface
- Each File Catalog requires a plugin that implements the methods of the interface
- Originally two file catalogs were supported by DIRAC:
 - LCG (LHC Computing Grid) File Catalog (LFC) - previously used by Belle II. It is a hierarchical catalog that allows to organise the files into a directory structure. The LFC contains the file replica information and very limited metadata functionality
 - The native DIRAC File Catalog (DFC) combines both replica and metadata functionality. In the DFC metadata can be associated with any directory, and subdirectories inherit the metadata of their parents

Namespace organisation



- In contrast to the LFC, Rucio is not inherently a hierarchical namespace (e.g. ATLAS use it as a flat namespace)
- To mimic the LFC's hierarchical structure, the following mapping is used (LFC → Rucio) :
 - File → File
 - Directory containing files → Dataset
 - Directory containing directories → Container

Namespace organisation



- Scope is another Rucio concept to :
 - Partition the namespace between VOs, users and activities, with the possibility of introducing specific permissions for specific scopes
 - Used to apply replication policies and for accounting
- Since there was no concept of scope in the LFC, it needs to be hidden from end-users and applications:
 - Belle II chose a deterministic function that maps each LFN to one specific scope
 - The function is part of the Rucio "policy package", an experiment-specific configuration that includes specific permissions, file naming constraints, etc.

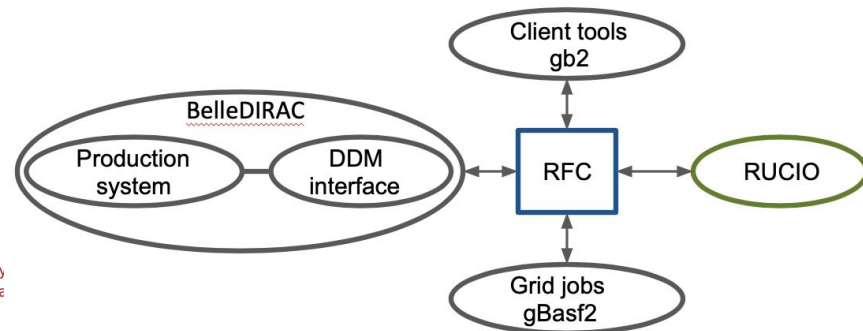
Rucio File Catalog plugin functionality

- The current implementation of the RFC plugin only contains the replica methods supporting bulk operations but not all of the metadata methods
 - There is ongoing work to support metadata
 - The RFC plugin uses the Rucio client which need to be installed on the DIRAC server
- Storage Elements and Users registered in DIRAC need to be created on the Rucio server
 - In Belle II this is done automatically by a new DIRAC agent that creates the Rucio Storage Elements (RSE) and their associated protocols, as well as the user accounts according to the configuration in DIRAC

```
class RucioFileCatalogClient(FileCatalogClientBase):
```

```
    READ_METHODS = FileCatalogClientBase.READ_METHODS + ['isLink', 'readLink', 'isFile', 'getFileMetadata',
                                                         'getReplicas', 'getReplicaStatus', 'getFileSize',
                                                         'isDirectory', 'getDirectoryReplicas',
                                                         'listDirectory', 'getDirectoryMetadata',
                                                         'getDirectorySize', 'getDirectoryContents',
                                                         'resolveDataset', 'getLFNForPFN', 'getUserDirectory']
```

```
    WRITE_METHODS = FileCatalogClientBase.WRITE_METHODS + ['createLink', 'removeLink', 'addFile', 'addReplica',
                                                           'removeReplica', 'removeFile', 'setReplicaStatus',
                                                           'setReplicaHost', 'createDirectory', 'removeDirectory',
                                                           'removeDataset', 'removeFileFromDataset', 'createData',
                                                           'changePathOwner', 'changePathMode']
```

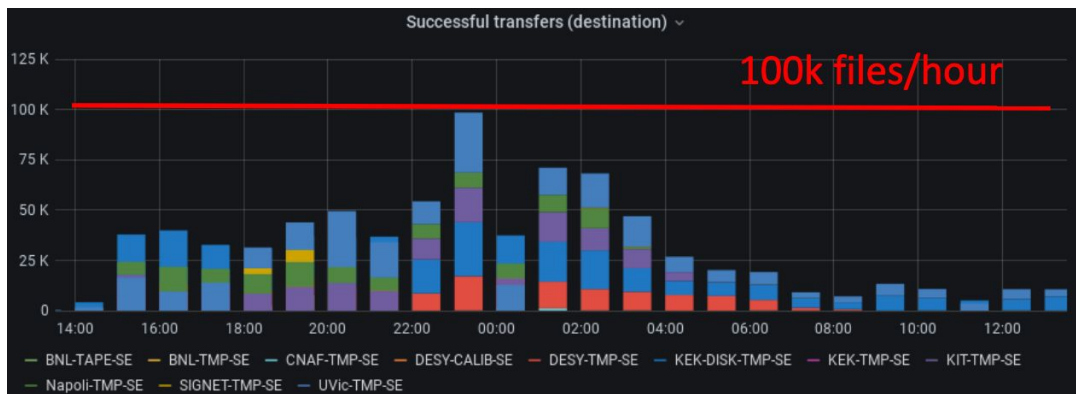


Methods overview

- Most of the methods have the same behaviour as the LFC (in particular the read methods that are 100% compatible)
- There are small differences for deletion due to the Rucio rules concept
 - Rule-based declarative system—tell it where you want the data and Rucio will get it there. A replication rule blocks the replicas of a files from deletion on a RSE. If the file has no replicas at the RSE the rule also transfers them there.
- 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 of its replicas is done asynchronously by a separate Rucio daemon
- More details about the methods are on backup slides

Conclusion

- A new Rucio File Catalog plugin is a key component to integrate DIRAC with Rucio. The plugin is being used successfully by the Belle II collaboration in production
- The current implementation only supports the replica functionality and not the metadata functionality
 - 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)
- Once the Rucio File Catalog will be included into generic DIRAC, many other scientific communities would be able to use Rucio



Thank you for your attention

감사합니다 Natick
Danke Ευχαριστίες Dalu
Grazie Thank You Köszönöm
Спасибо Dank Gracias
谢谢 Merci Seé
ありがとう Obbrigado

Backup slides

Container constraint

- Containers in Rucio cannot contain files, therefore a mixture of files and sub-directories in the same parent directory is not supported, e.g. :

```
lfc-ls -l /grid/belle/MC/fab/
```

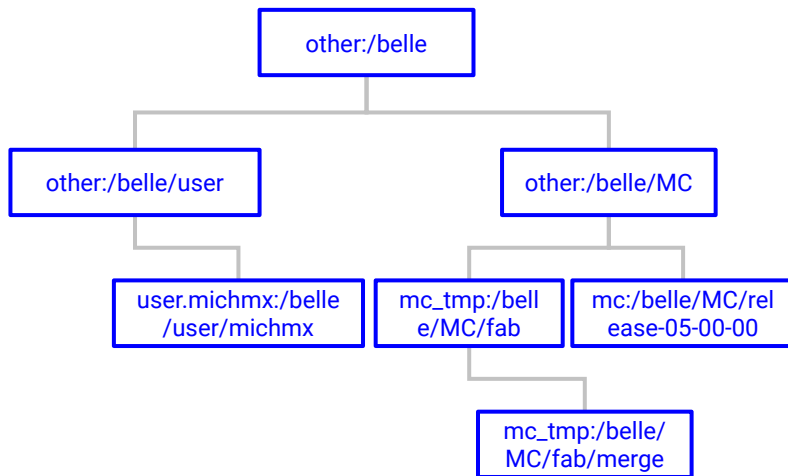
drwxrwxr-x	1 292	147	0 Oct 13 2015 merge	
-rw-rw-r--	1 292	147	45240230 Jan 10 2017 output_15634.root	← Not possible
drwxrwxr-x	2 353	147	0 Jun 23 2017 prerelease-00-09-00b	
drwxrwxr-x	1 353	147	0 Dec 12 2017 prerelease-01-00-00b	

- During the migration to Rucio such files were identified and not migrated
- This constraint was introduced into Belle II grid analysis tools to prevent users from encountering this feature

Write methods

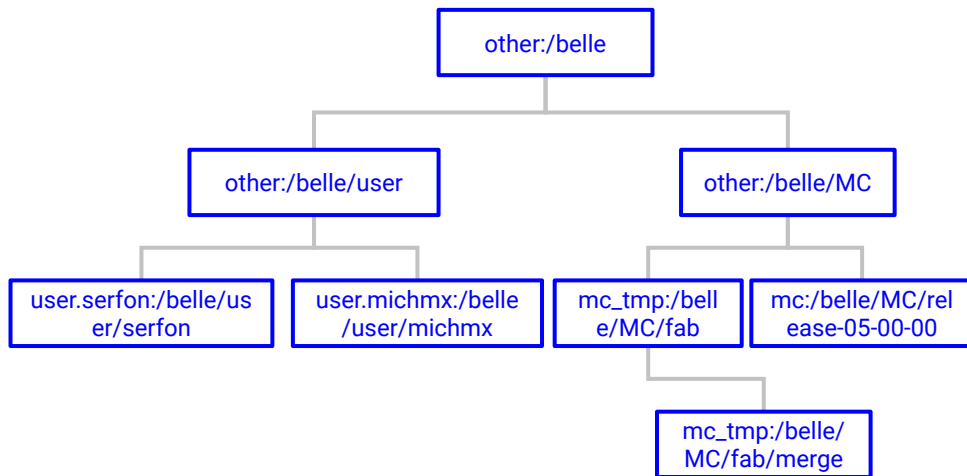
- Write methods implemented :
 - addFile :
 - Complex method since Rucio and LFC concept are quite different, that does multiple things (backup slides for details)
 - New method was introduced on the Rucio side to provide atomicity + bulk
 - addReplica :
 - Add the replica + create a rule on the file
- 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

Addfile method



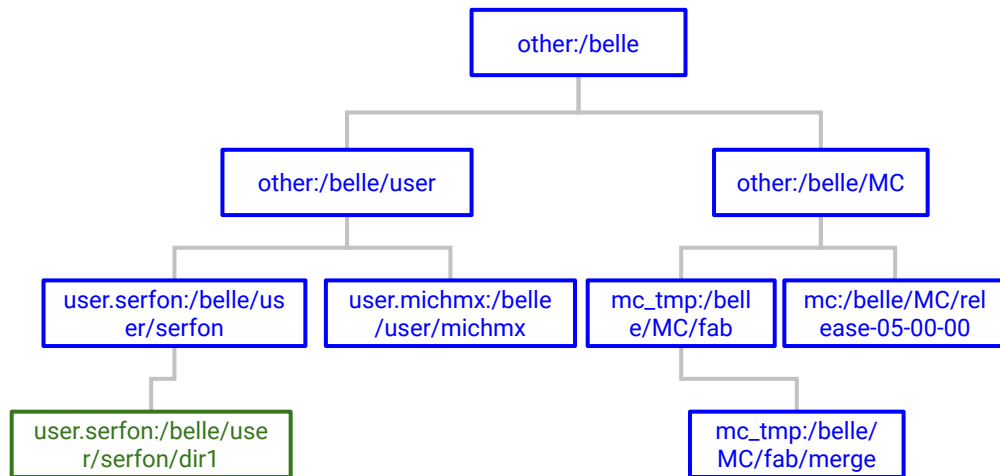
- Example : Creation of files
/belle/user/serfon/dir1/file{1-3} on
SE A
- The addfile method is a bulk and
atomic method that :

Addfile method



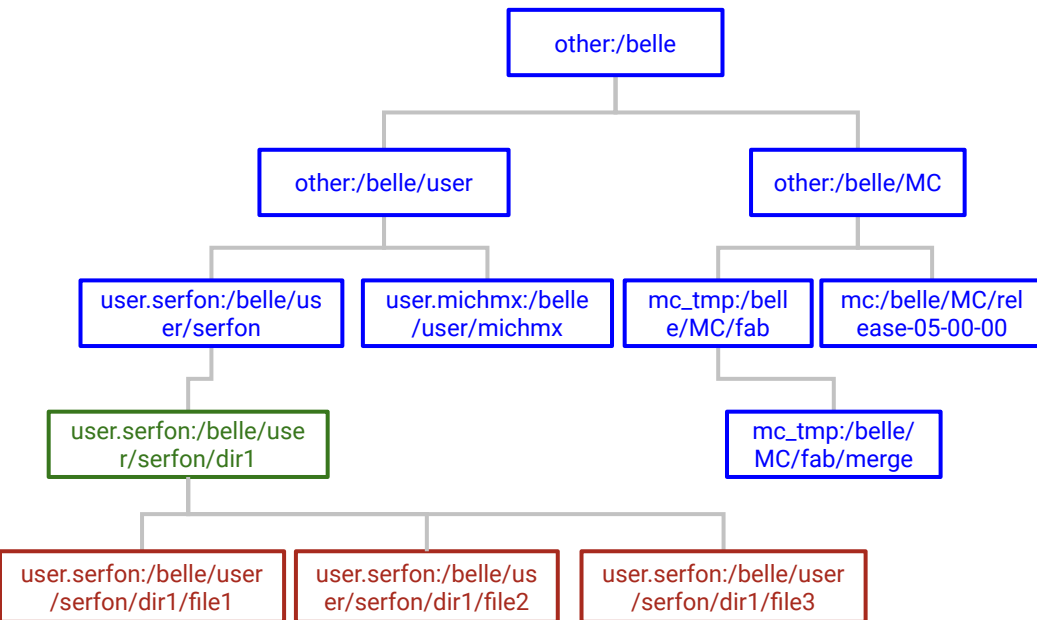
- Example : Creation of files
/belle/user/serfon/dir1/file{1-3} on
SE A
- The addfile method is a bulk and
atomic method that :
 - Creates all the non-existing parent
directories in the hierarchy

Addfile method



- Example : Creation of files
/belle/user/serfon/dir1/file{1-3} on
SE A
- The addfile method is a bulk and
atomic method that :
 - Creates all the non-existing parent
directories in the hierarchy
 - Creates the directory containing the files
if it doesn't exist and create a Rucio
rule (RSE expression ANY and grouping
NONE)

Addfile method



- Example : Creation of files
/belle/user/serfon/dir1/file{1-3} on
SE A
- The addfile method is a bulk and
atomic method that :
 - Creates all the non-existing parent
directories in the hierarchy
 - Creates the directory containing the files
if it doesn't exist and create a Rucio
rule (RSE expression ANY and grouping
NONE)
 - Create the files and their replicas on A

Delete methods

- Delete methods have some different behaviour because of the different concept in the LFC/Dirac and Rucio, e.g. :
 - removeReplica : In Rucio, not possible if the file belongs to a dataset and there is a rule on this dataset. If this is the case, the RFC doesn't do anything. If a rule exists on the file, remove it
 - removeFile :
 - 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

AMGA metadata catalog

- The Belle II distributed computing system includes AMGA (ARDA Metadata Grid Application) - a stand-alone Grid metadata catalogue for support of metadata description, discovery and archiving of large-scale scientific data.
- In Belle II AMGA handles physics analysis oriented metadata rather than standard basic files-metadata like checksum and size