

New protocols for exascale data management with Rucio

European Organization for Nuclear Research (CERN)
CERN Summer Student Programme 2021

Author:

David Población Criado <david.poblacion.criado@cern.ch>

Under the supervision of:

Mario Lassnig <mario.lassnig@cern.ch>

Martin Barisits <martin.barisits@cern.ch>

September 10, 2021

Abstract

The aim of this report is to cover the work done during my placement at CERN as a Summer Student from June 21 to September 10. It is related to Rucio, a Distributed Data Management (DDM) system made by CERN which manages the experimental data produced by ATLAS and it is used worldwide by a lot of other organizations and institutes. In the following years the data that Rucio will have to deal with is going to increase in one magnitude when the High Luminosity-Large Hadron Collider (HL-LHC) upgrade has been finished.

1 Introduction

Rucio [1], [2] is a Distributed Data Management (DDM) system that enables to manage huge volumes of data that are located in several heterogeneous places around the globe. Rucio is used for various organizations like CERN and Fermilab, among others [3], [4]. Originally, it was created to fulfill the needs of the ATLAS Experiment [5], allowing to manage large volumes of data in an efficient and scalable way. It is an open-source software licensed under the Apache 2.0 License with an active community on GitHub [6].



Figure 1: Rucio: Scientific Data Management

It provides several benefits for data storage. Some of them are [7]:

- Seamless integration of both scientific and commercial storage systems.
- Data is stored in global single namespace and can contain any potential payload.
- Facilities can be distributed at multiple locations belonging to different administrative domains, allowing transfers of data between them (including tapes, disks, clouds, HPCs, ...). It is not necessary to execute Rucio in the data centres, they can choose which storage suits them best.
- Management of the authentication and authorization for individual users and groups.
- ...

To give a magnitude of the data Rucio manages in the ATLAS Experiment [7]:

- More than a billion of files which in total are more than 500 Petabytes of data.
- Distributed in 120 different data centres, 5 HPCs, 2 clouds and more than 1000 active users.
- Each year more than 500 PB of data are transferred and deleted and about 2.5 Exabytes of data are uploaded and downloaded (per year).

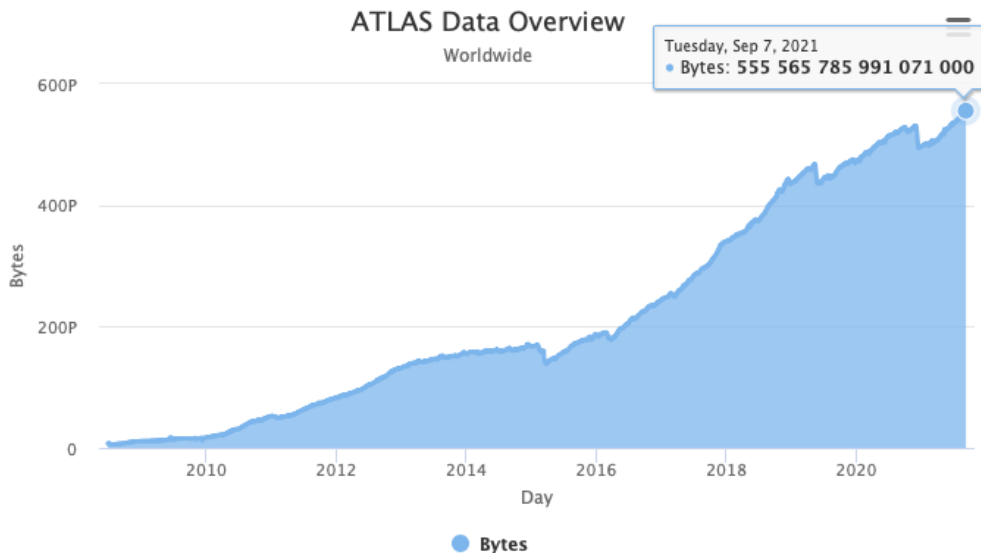


Figure 2: Data managed by Rucio in ATLAS

In the next upgrade of the Large Hadron Collider (LHC), called High Luminosity LHC (HL-LHC), the data generated by the experiment are going to increase one order of magnitude. Stage 2 of the upgrade is now in progress and stage 3 is planned to be finished at 2027 in ATLAS [8]. This would be a great challenge for the software layer, in order to manage and store this magnitude of data effectively.

1.1 Concepts

Rucio makes several abstractions in order to be easier to administrate the data. There are some concepts related to this abstraction:

Namespace

All data are organized using Data Identifiers (DIDs), which can be of three types: files, which is the smallest unit that corresponds to the physical files saved in the file system; datasets, which is a logical unit (not present in storage) and it allows users to group only files (it is not possible to have one dataset inside other) and containers, which are used to group datasets and other containers.

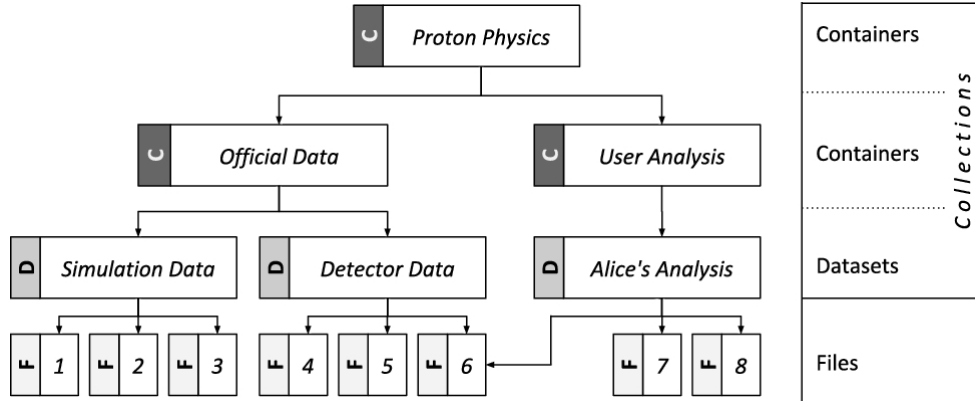


Figure 3: Data Identifiers in Rucio [1]

DIDs are composed of an scope and name (*scope:name*). This pair must be unique in the system and forever: if a DID is deleted, that pair cannot be reused.

Accounts

An account is needed to interact with Rucio. There are two types of accounts: for individual users or for groups. There are several types of authorisation: user and password, X.509 certificates, LDAP, SSH, Kerberos or CERN SSO. The permissions can be personalised for each account and each account have its own scope, similar to UNIX home directory.

Storage

Another abstraction is the management of storage. Rucio uses Rucio Storage Elements (RSEs) which corresponds with actual locations of data. That locations can be very heterogeneous: tapes, disks, clouds, HPCs, ... and Rucio allows to interact with all of them with different protocols (ROOT, HTTP/WebDAV, ...).

Subscriptions and Replication Rules

Subscriptions allow to make data placement requests for incoming DIDs by defining a metadata filter on matching DIDs and later the replication rules will be automatically created.

The replica management, data stored in a particular RSE, is based on replication rules which are defined on the DIDs. It defines the minimum number of replicas to be available.

1.2 Architecture

Clients layer

Consists in all interfaces that final users interact with: from the command line programs (**rucio** and **rucio-admin**) to the WebUI. Moreover, this layer has an Application Programming Interface (API),

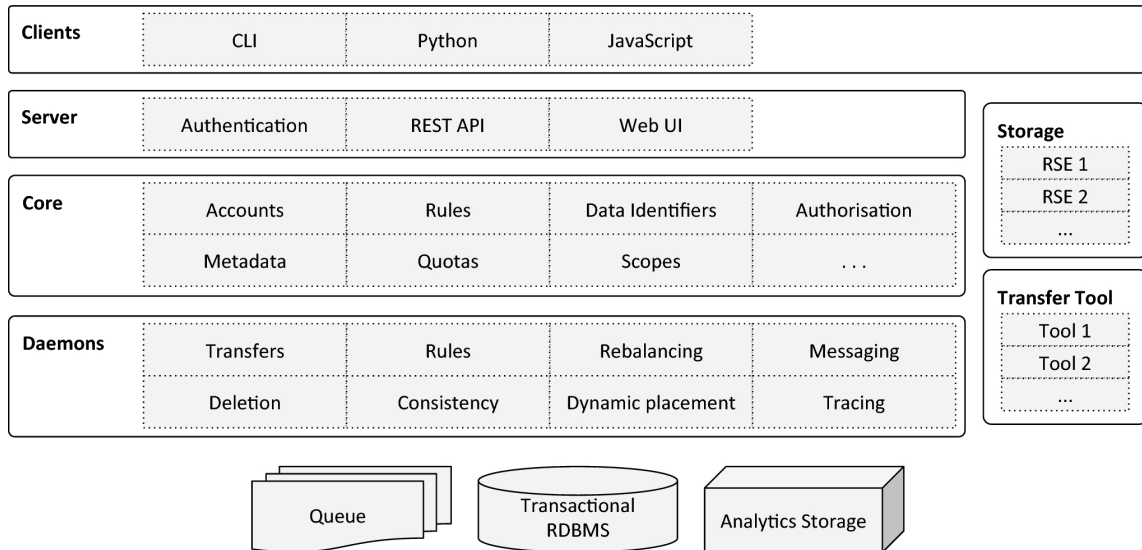


Figure 4: Architecture of Rucio [1]

organized in different classes, each one providing different functionalities: authentication, upload of files, subscription, ...

Server and core layers

The server listens incoming queries to the REST API and forwards them to the core, received by a web server and relayed to a WSGI container that executes the Rucio function and the result is streamed back to the WSGI container and later to the web server (HTTP response).

Daemons layer

Consists in all the components that are always executing and orchestrate the work of the entire system. Each daemon have its function: messaging, evaluate rules, ...

Storage layer

It is the responsible to manage the different protocols and attributes of a storage system with the Rucio Storage Element (RSE) abstraction.

Persistence layer

It is the lowest layer, where all the application states and the data are stored. Rucio uses SQLAlchemy that allows to manage different relational database management systems (RDBMS) such as SQLite, MySQL or PostgreSQL.

2 Work done

At the very beginning, I set up a local development environment on my own computer using Docker [9] for the execution of Rucio and PyCharm as IDE. During these months, I was working in several issues registered on GitHub, that can be either bugs found or features and enhancements. In particular, I worked and resolved the following tickets (ordered by ticket number).

#1834 - Add validator for traces - Traces

Ticket URL: <https://github.com/rucio/rucio/issues/1834>

Pull request URL: <https://github.com/rucio/rucio/pull/4740>

Clients must send a JSON to the tracer server (ActiveMQ) following a correct schema, but the schemes are not neither defined nor checked, so a bug in a client could trigger in an error in any system that use the traces. I added a validator for the different schema types depending the `eventType` key of the JSON.

#2630 - rucio.cfg vs config table - Core & Internals

Ticket URL: <https://github.com/rucio/rucio/issues/2630>

Pull request URL: <https://github.com/rucio/rucio/pull/4786>

The configuration parameters can be stored in 3 different places: Rucio configuration file (`rucio.cfg`), configuration table and RSE attributes. For the first one, it existed a function called `config_get` that retrieves the pair (`section`, `option`) from the configuration file. For the second one, it existed another function `get` that loaded the same pair from the configuration table.

The goal of the ticket was to combine these two methods in one named `config_get`: it first checks in the configuration file for the pair and, both if it is not found and the method is called from a daemon or server, looks in the configuration table (because the DB is only available from the server layer). Even if it is called from server/daemon, it is possible to avoid looking at the config table with the `check_config_table` argument in order to avoid circular imports and other issues such as trying to get config from the DB if the DB is not created yet (the section `database` cannot be retrieved from config table).

So now all the calls to get the configuration from the file and the table are managed with the same method (I had to refactor the old calls). Moreover, I created another method called `is_client` to check if a chunk of code is called from a client or from a server/daemon.

#2631 - Page listing config table and RSE Attribute parameters - Documentation

Ticket URL: <https://github.com/rucio/rucio/issues/2631>

Pull request URL: <https://github.com/rucio/documentation/pull/36>

It consisted in the creation of a page in the docs website indexing all possible parameters of Rucio configuration, whose are divided in three different locations: configuration file (`rucio.cfg`), configuration table in the database and RSE attributes. Firstly, I searched on the code every pair (`section`, `option`) of configuration access (in any of the three places). Secondly, I created an index with these pairs and later I infer a description from the code, if it is a mandatory or optional field, default value and, if possible, give an example. The page is published in: <https://rucio.cern.ch/documentation/configuration-parameters>.

#3431 - list-file-replicas --all-states should show the states - Clients

Ticket URL: <https://github.com/rucio/rucio/issues/3431>

Pull request URL: <https://github.com/rucio/rucio/pull/4813>

The `rucio list-file-replicas` command with the `--all-states` flag did not print the states of the requested DIDs in each RSE. I added that information in the output in a short format because the length of the table was long enough. I also updated the help to explain the legend of the states.

#3987 - Some daemons CLI are missing a sleep-time option - Clients

Ticket URL: <https://github.com/rucio/rucio/issues/3987>

Pull request URL: <https://github.com/rucio/rucio/pull/4722>

Many of the daemons did not have the option to set a different sleep time, only could work with the default hardcoded value. The modification consisted in create a new option in the binaries of the daemons and propagate that value from the binaries to the server code.

#4012 - No timeout for calls to rucio auth server - Clients

Ticket URL: <https://github.com/rucio/rucio/issues/4012>

Pull request URL: <https://github.com/rucio/rucio/pull/4729>

All requests to the authorization server did not have a maximum time, so the client can enter an active wait (wasting resources) if it does not get any response from the server. The modification I did is that every request to the server must be performed through the `_send_request` function, which takes into account the maximum time (10 minutes). I had to adapt the function to the new calls.

#4073 - Expose the RSE limits through rucio-admin - Clients

Ticket URL: <https://github.com/rucio/rucio/issues/4073>

Pull request URL: <https://github.com/rucio/rucio/pull/4704>

The RSE limits were not visible through the command line and it was not possible to add or delete the limits from the command line client. I made those limits available from `rucio-admin` and added two new commands to the `rucio-admin` `rse: set-limit` and `delete-limit`, which allow to add and delete a limit, respectively. Moreover, I modified the Flask API to add the possibility to delete a limit.

#4243 - Abacus : Add option to set limit - Core & Internals

Ticket URL: <https://github.com/rucio/rucio/issues/4243>

Pull request URL: <https://github.com/rucio/rucio/pull/4776>

The `rucio-abacus-collection-replica` have not got the option to change the size of the limit, that was hardcoded. I modified the binary and propagated that value to the server.

#4460 - Query hints for update queries - Database

Ticket URL: <https://github.com/rucio/rucio/issues/4460>

Pull request URL: <https://github.com/rucio/rucio/pull/4749>

The optimizer hints for Oracle databases did not work in `update` and `delete` SQL sentences. I had to change `with_hint` with `prefix_with` because the first ones that were used only worked for `select` sentences. In addition, I updated the SQLAlchemy syntax of those sentences to the 2.0 version, the new version of this framework.

#4466 - Correct and Avoid redefining of built-in function - Core & Internals

Ticket URL: <https://github.com/rucio/rucio/issues/4466>

Pull request URL: <https://github.com/rucio/rucio/pull/4735>

Many of the predefined functions and reserved names of Python such as `filter` or `type` were been redefined with local variables, which is a programming bad practice. Because of that and following the Python style guide I refactored all the re-definitions of reserved names for more descriptive ones or adding an underscore to the name (`type_`).

#4616 - CLI/python Clients - Recursive Upload of an input folder - Clients

Ticket URL: <https://github.com/rucio/rucio/issues/4616>

Pull request URL: <https://github.com/rucio/rucio/pull/4702>

When uploading files to Rucio, it was not possible to add them recursively, that is, specify a folder and automatically upload every subfolder and files in a recursive way. To solve that, I modified the `rucio upload` command, adding a new flag `-recursive` that has this behaviour. To design the functionality, I had to distinguish between folders that only had either files or folders, because it is only allowed to have 3 types of data: containers, datasets and files. Containers can only have datasets inside and datasets can only have files, so it is impossible to have both folders and files at the same time.

#4700 - Tracer: Add the possibility to point to a different host than rucio host - Clients

Ticket URL: <https://github.com/rucio/rucio/issues/4700>

Pull request URL: <https://github.com/rucio/rucio/pull/4741>

The host used to send the generated logs is the same in which Rucio is installed, it was not possible to change it. I added a new parameter in the configuration file, `tracer_host` that allows to set a new host to send the traces, also modifying the clients to get it work.

#4742 - Delete distance API - Clients

Ticket URL: <https://github.com/rucio/rucio/issues/4742>

Pull request URL: <https://github.com/rucio/rucio/pull/4743>

Although it was a method to delete the distance between two RSE, it was not possible to access it because it was not exposed to the REST API. I modified the API, adding a new method to allow this and implementing the call from `rucio-admin`.

#4755 - Fix add header script for bin files - Clients

Ticket URL: <https://github.com/rucio/rucio/issues/4755>

Pull request URL: <https://github.com/rucio/rucio/pull/4756>

The `add_header` script adds the header to the Python code sources, updating the authors that have work in each file. The issue was that it only added the UTF-8 encoding line to the files whose name finished in `.py`, so it was not added to the binaries, even though they are Python files too. This triggered some errors when adding my name to the list of authors because it has an accent mark. I modified the script to add the encoding line to the files in the `bin` folder.

#4757 - Change timeout behaviour of hermes - Messaging

Ticket URL: <https://github.com/rucio/rucio/issues/4757>

Pull request URL: <https://github.com/rucio/rucio/pull/4770>

The sleep behaviour of the daemon was inefficient because it slept each 10 seconds if there was not a bulk received. I changed this conduct to work as in the other daemons (with the `daemons_sleep` method) and sleep 1 minute, to reduce the waiting.

#4758 - Make limit variable of the judge-evaluator configurable - Rules

Ticket URL: <https://github.com/rucio/rucio/issues/4758>

Pull request URL: <https://github.com/rucio/rucio/pull/4763>

The `rucio-judge-evaluator` daemon did not have the option to change the limit of DIDs to evaluate, it was hardcoded. I modified the binary and propagated this value to the server.

#4787 - Make limit variable of the judge-evaluator configurable - Rules

Ticket URL: <https://github.com/rucio/rucio/issues/4787>

Pull request URL: <https://github.com/rucio/rucio/pull/4788>

The proxy certificate and the client certificate had different permission levels (600 and 400, respectively), so the ATLAS template for the configuration file did not work correctly. I removed those parameters which were unnecessary from the templates and updated them.

#4802 - Update client to support default accounts - Clients

Ticket URL: <https://github.com/rucio/rucio/issues/4802>

Pull request URL: <https://github.com/rucio/rucio/pull/4823>

Rucio allowed default accounts but when executing the client it was mandatory to specify an account to sign in instead of using the default one (only worked for X.509 auth type). I modified the clients (`baseclient`, `uploadclient` and `subscriptionclient`) to allow to use a default account and the API to retrieve the default account if another account is not set. It was quite easy for all authentication types except for OIDC, that I was not be able to implement it because the identity string is not known when getting the token (or it cannot be retrieved).

#4810 - X509 auth in the WebUI fails multiple accounts - Authentication & Authorisation

Ticket URL: <https://github.com/rucio/rucio/issues/4810>

Pull request URL: <https://github.com/rucio/rucio/pull/4811>

Logging in the WebUI with a X.509 certificate failed if there are more than one account registered in Rucio. The failure is produced if first log in with one account (e.g. `ana`) and later try to log in with another account (e.g. `alice`), it seems that Rucio tries to map the second identity with the first account. It was quite hard to replicate the error on my own computer but at the end I got it. I am quite sure that the error is produced because of the cookies: the `select_account_name` method in `rucio.web.ui.flask.common.utils` first gets all the possible accounts that an identity can be mapped to but later it tries to retrieve the account from a cookie but that account maybe is not mapped with the identity given as argument of the method, raising the error. I submitted a PR that could fix the error (in my local computer it does, maybe in production it does not): if the account stored in the cookie it is not equal with any of the accounts got from the identity string, the cookie is ignored.

#4815 - Fix deletion of old tokens - Authentication & Authorisation

Ticket URL: <https://github.com/rucio/rucio/issues/4815>

Pull request URL: <https://github.com/rucio/rucio/pull/4822>

The deletion of expired tokens did not work properly because of deadlocks and the hints did not run for Oracle databases. I rewrote the queries using the new SQLAlchemy syntax in order to get the hints work and split each of them in a `select` and a `delete` statement.

3 Acknowledgments

I would like to express my gratitude to my supervisors, Martin Barisits and Mario Lassnig, who guided me throughout this project. I would also like to thank Rucio Community that provided me helpful insights and to CERN that allowed me this amazing opportunity.

References

- [1] M. Barisits, T. Beermann, F. Berghaus, B. Bockelman, J. Bogado, D. Cameron, D. Christidis, D. Ciangottini, G. Dimitrov, M. Elsing, V. Garonne, A. di Girolamo, L. Goossens, W. Guan, J. Guenther, T. Javurek, D. Kuhn, M. Lassnig, F. Lopez, N. Magini, A. Molfetas, A. Nairz, F. Ould-Saada, S. Prenner, C. Serfon, G. Stewart, E. Vaandering, P. Vasileva, R. Vigne, and T. Wegner, “Rucio: Scientific Data Management,” *Computing and Software for Big Science*, vol. 3, no. 1, p. 11, Dec. 2019, ISSN: 25102044. DOI: [10.1007/s41781-019-0026-3](https://doi.org/10.1007/s41781-019-0026-3). [Online]. Available: <https://doi.org/10.1007/s41781-019-0026-3>.
- [2] European Organization for Nuclear Research (CERN), *Rucio - Scientific Data Management*. [Online]. Available: <https://rucio.cern.ch/> (visited on Aug. 21, 2021).
- [3] M. Lassnig, M. Barisits, P. J. Laycock, C. Serfon, E. W. Vaandering, K. Ellis, R. Illingworth, V. Garonne, J. White, J. A. Clark, G. Fronze, R. Joshi, I. Johnson, and B. Bauermeister, “Rucio beyond ATLAS Experiences from Belle II, CMS, DUNE, EISCAT3D, LIGO/VIRGO, SKA, Xenon,” Tech. Rep., 2019. [Online]. Available: <https://indi.to/KtXJD> (visited on Aug. 20, 2021).
- [4] European Organization for Nuclear Research (CERN), *Managing scientific data at the exascale with Rucio — CERN*. [Online]. Available: <https://home.cern/news/news/computing/managing-scientific-data-exascale-rucio> (visited on Aug. 19, 2021).
- [5] —, *ATLAS Experiment at CERN*. [Online]. Available: <https://atlas.cern/> (visited on Aug. 19, 2021).
- [6] Rucio Community, *Rucio - Scientific Data Management*. [Online]. Available: <https://github.com/rucio/rucio> (visited on Aug. 19, 2021).
- [7] M. Lassnig, “Rucio - Scientific Data Management,” Tech. Rep. [Online]. Available: https://indico.cern.ch/event/948776/contributions/3986808/attachments/2092795/3516935/rucio_bsw.pdf (visited on Aug. 17, 2021).
- [8] European Organization for Nuclear Research (CERN), *The HL-LHC project — High Luminosity LHC Project*. [Online]. Available: <https://hilumilhc.web.cern.ch/content/hl-lhc-project> (visited on Aug. 19, 2021).
- [9] Rucio Community, *Setting up a Rucio development environment*. [Online]. Available: <https://github.com/rucio/rucio/blob/master/etc/docker/dev/README.rst> (visited on Aug. 26, 2021).