GOOGLE SUMMER OF CODE 2018

Rucio: Scalable and generic metadata

- Asket Agarwal (India)

MY PROJECT

Rucio produces large amounts of metadata for it’s files and datasets which is stored in a central Rucio server. However there is a **fixed set of metadata attributes** that can be stored currently. Rucio wanted a **generic metadata** catalogue with no restriction on the kind of metadata stored for the files. The project is to to design and implement a generic and scalable metadata component that integrates with the core transactional model of Rucio.

PROJECT WORKFLOW

Created 4 pull requests to the upstream code that has been merged.

- The generic metadata component was adapted for postgres and mysql databases. Also added Client Api's, some tests, Command line api's and the rest Apis.
- The generic metadata was adopted for oracle. This function is however only available for Oracle 12 and greater versions.
- Added some more unittests and documentation.
- Added support for sqlite. However currently only add, get and delete is supported.

New Client API's

1) **add_did_meta(scope, name, metadata)**: Adds new metadata or updates old metadata of a given Data Identifier.

Parameters:

- Scope
- Name
- Metadata: dictionary containing key-value pairs.

{ “key1” : “value1”, “key2” : “value2” …… }

2) get_did_meta(scope, name): Returns the whole set of metadata of given Data Identifier.

Parameters:

- Scope
- Name

Returns dictionary containing all the metadata for the DID.

3) delete_did_meta(scope, name, key): Deletes metadata from a given Data Identifier.

Parameters:

- Scope
- Name
- Key: Key to delete

If a key is not present in the json column it throws an Exception.

4) list_dids_by_meta(scope, metaQuery): Queries the JSON column using the select_query. Returns list of Data Identifiers.

Parameters:

- Scope
- metaQuery: dictionary containing key-value pairs.

{ “key1” : “value1”, “key2” : “value2” …… }

Returns a list of dids which satisfies all the key-value pairs in the query.

Rucio CLI

- rucio get-did-meta did1 did2 ……
- rucio add-did-meta --did didName --key key --value value
- rucio delete-did-meta --did didName --key key
- rucio list-dids-by-meta key1=value1 key2=value2 ……

Some project details and challenges

- The main challenge was adapting the json support in different databases to sqlalchemy.
- SqlAlchemy Json type currently has json type for only postgres and mysql.
- For postgres SqlAlchemy even has JSONB type support which is significantly faster JSON and also support indexes which can later be useful when this feature is deployed.
- For mysql index then can be supported when sqlalchemy supports JSONB for mysql.

- SqlAlchemy does not support json for **oracle** hence I have made use of the Textual SQL feature of sqlalchemy where I have passed raw queries to the database.
- The data is stored as **CLOB format** in Oracle so like postgres we can make use of indexes in the future.
- The JSON support is only present is versions 12.* and higher. I have also made sure that the lower instances of oracle won't break after the new feature is rolled out.

- For **sqlite** there is currently only support for add_did_meta, get_did_meta and delete_did_meta.
- In sqlite json will be stored as text type.
- Thus we would be able to add and read metadata for a single data identifier but won't be able to list data identifiers using the metadata.

Future Tasks

- Currently travis only runs tests for oracle 11 versions. When this is upgraded it will be able to run the metadata tests for oracle.
- Adding **list_did_meta support for sqlite** when json support is available for sqlite.
- **Refactor code for oracle and sqlite** when sqlalchemy starts supporting json type these databases.

My Experience

- What I learnt. (New Technologies, Tools and libraries)
- Working with an international team and learnt about development cycles of an organisation.
- Contributing code to an open source project.
- Post Gsoc (See project through, Get more people excited about Rucio).

THANK YOU