ATLAS Replica Management in Rucio: Replication Rules and Subscriptions

CHEP 2013

Martin Barisits

CERN PH-ADP, Geneva, Switzerland
&
University of Innsbruck, Innsbruck, Austria

15. October 2013

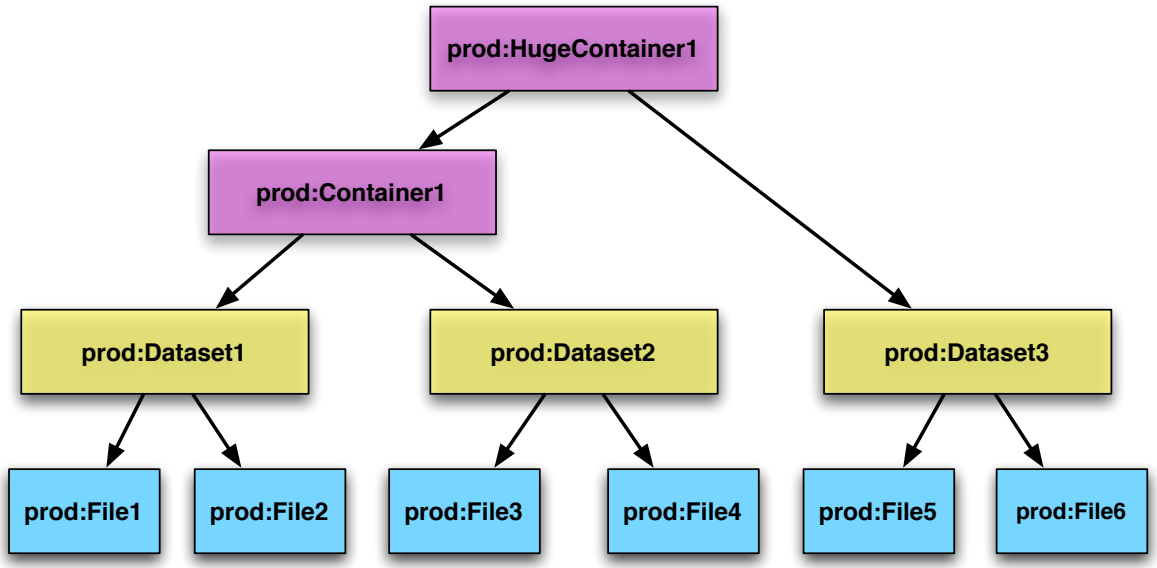
Rucio

- DQ2 is the current Distributed Data Management system of the ATLAS collaboration:
  - Manages 150 PB of data,
  - 750 storage endpoints on more than 120 sites globally,
  - 1000 users.
- DQ2 reaches its limits:
  - Scalability,
  - Operational burden,
  - Extension to new technologies.
- Rucio[1] is the next generation ATLAS distributed data management system to follow DQ2.

[1] *Garonne et al.* – Rucio - The next generation of large scale distributed system for ATLAS Data Management, CHEP2013

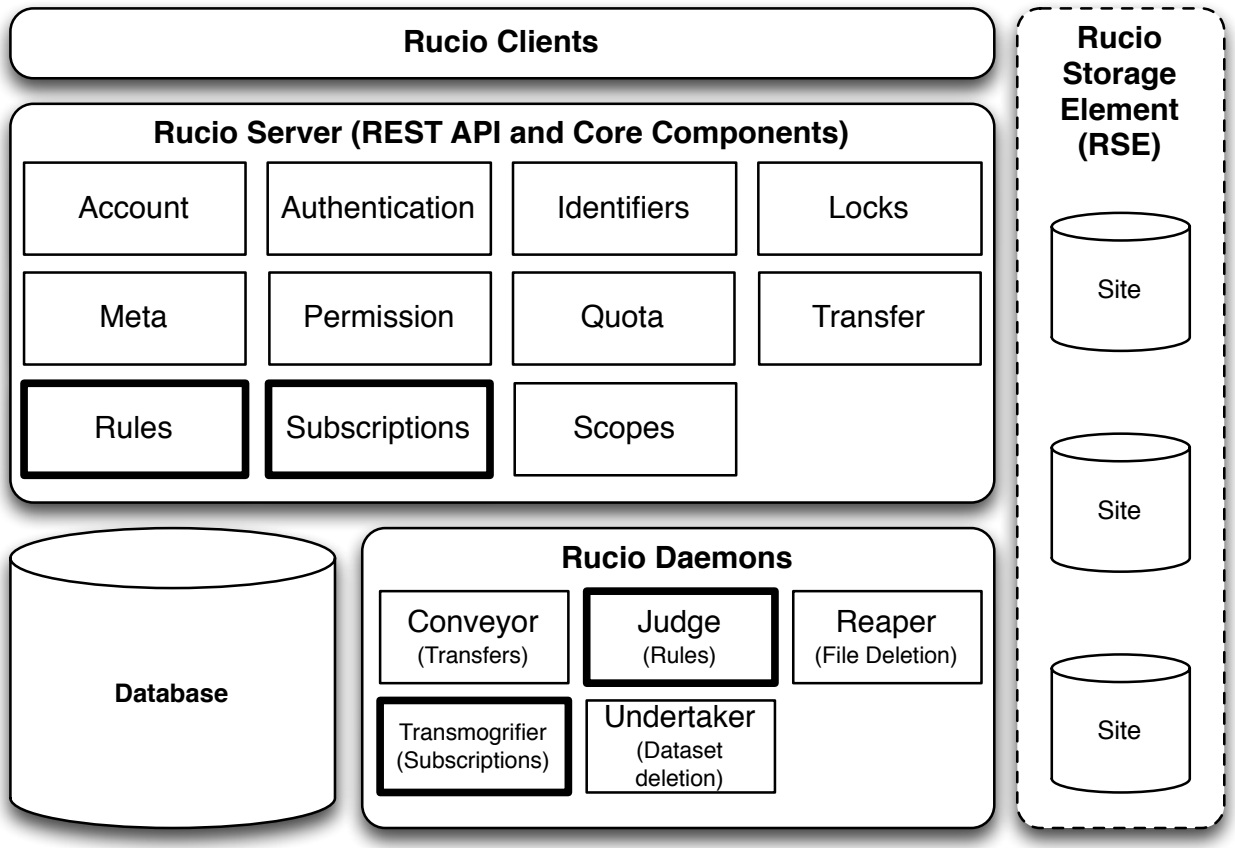
Rucio concepts

- Accounts represent users, groups or activities.
- Namespace: Data hierarchy with meta data support:
  - Data name space is partitioned by scopes.
  - $(scope, name)$ tuple identifies all data.
  - Files can be grouped into datasets.
  - Datasets can be grouped into containers.
- Rucio Storage Element – logical abstraction for storage space.
  - Can be assigned meta attributes.



Rucio architecture

- Distributed architecture.
- RESTful interface to server.
- Oauth model.
- Client / server / daemon / resource layer.



Replica Management – in general

Challenge: Satisfying the users need to express their data replication intentions while giving the system the freedom to make optimized decisions.

DQ2:

- Replication based on datasets.
- Replication requests are inflexible and specific.
- System very limited in optimising these requests.

Rucio:

- Replication based on files, datasets or containers.
- Replication requests can be both flexible and specific based on RSE expressions.
- System able to optimize storage space and network bandwidth.

Replica Management – Rucio

- Replica Management is based on **replication rules** set on files, datasets or containers.
- A replication rule consists of an RSE expression, defining a set of possible destination RSEs and the number of replicas it should create.
- Integrate the policy workflow into the data management system.

Example: A user wants to replicate a dataset to two tier-2 sites in the United Kingdom.

- Data identifier: `userA:ds1`
- copies: `2`
- RSE expression: `tier=2&country=uk`

$\rightsquigarrow$ Rucio picks two RSEs out of the set specified by the RSE expression.

Win/Win:

+ User does not have to 'hand-pick' RSEs.
+ Rucio has more freedom in picking the actual replica destinations.

Replication Rules

- Rules are owned by an account.
- Quota checks are based on the owner of the rule, not the creator of the data.
- When specified on a dataset or container, the rule will affect its whole content; Subsequent changes of content will be considered.
- A replication rule generates replica locks on the replicas to protect them from deletion.
- Replicas without locks will be picked up by the Reaper (deletion daemon).
- Changed datasets/containers have to be re-evaluated by the Judge (rule daemon) to apply new locks.
- Stuck replication rules (failed transfers) will be re-evaluated as well.

Replication Rules – further options

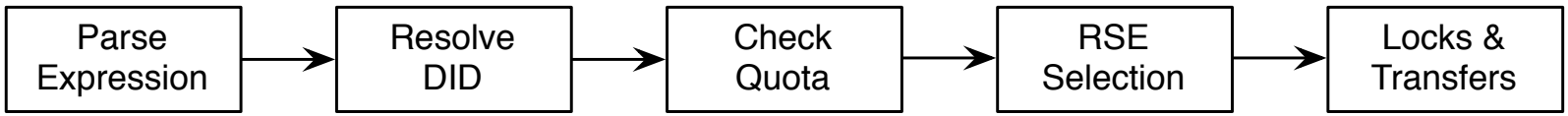
- **grouping** (for datasets or containers):
  - `NONE`: Files will be randomly replicated to RSEs defined in the RSE expression.
  - `ALL`: All Files will be replicated to the same RSE .
  - `DATASET`: Files in the same dataset will be replicated to the same RSE, but different RSEs will be used for different datasets.
- **weight**: A weighting attribute can be specified, influencing the selection of the destination RSEs. (e.g.: MoU share)
- **lifetime**: Rule lifetime, after which it gets removed.
- **locked**: Additional protection from accidental deletion of the rule.

Replication Rules – RSE expression

- RSE expressions use the RSE expression language to define a set of RSEs.
- RSE names can be directly used: `SITEA|SITEB`
- Meta-attributes on RSEs: `type=tape&country=de`
- Set operators:
  - $\cap$ (Intersect) represented by `&`
  - $\cup$ (Union) represented by `|`
  - $\setminus$ (Complement) represented by `\`
- Order of operations can be given by brackets `(,)`
- Examples:
  - `(type=tape&country=de)\SITEA`
  - `(tier=2\country=de)|CERN_EOSDISK`

Replication Rules – Workflow & Architecture

- Adding replication rules is done synchronously in the core.
- Re-Evaluation & deleting replication rules is done asynchronously by the Judge (rules daemon).



RSE Selection based on:

- Existing locks and scheduled transfers,
- Weighting option,
- Selection strategy: Minimises used storage volume
- Alternative strategies: Prioritise strong network links, Prioritise geographical dispersion, mixed strategy, ...

Replication Rules – Evaluation

- Evaluation based on a load emulation framework for scaling tests[2].
- Tested on nominal and 2x peak load of DQ2.
- Conclusion: After tuning, replication rules prototype performed without problems and without generating backlogs.

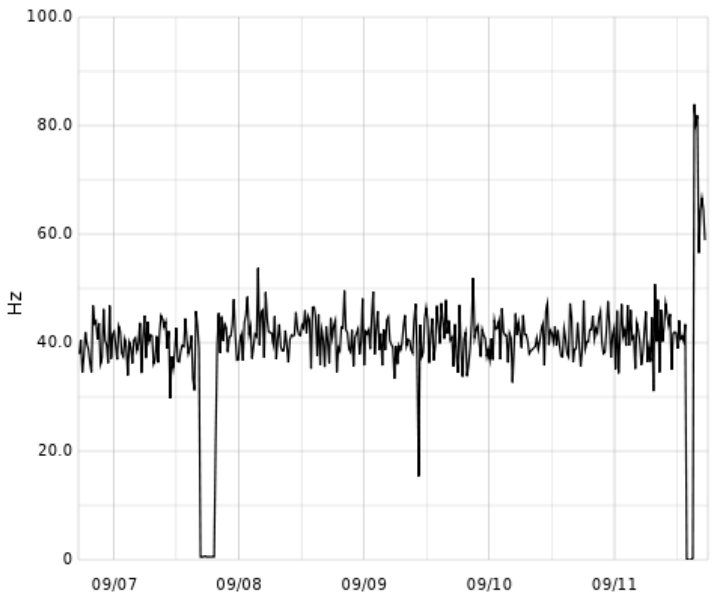


Figure : Rule deletion

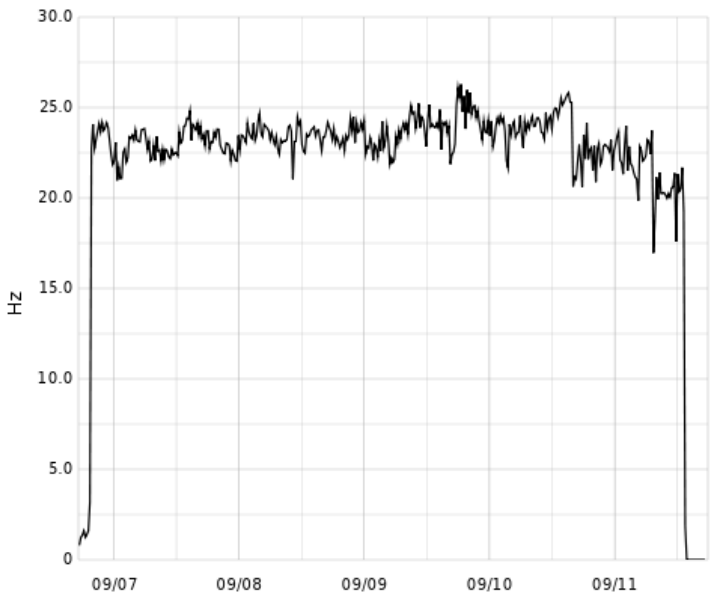


Figure : Rule evaluation

[2] *Vigne et al.* – DDM Workload Emulation, CHEP2013

Subscriptions

- Replication rules manage existing data.
- Subscriptions are responsible for defining replication behaviour for future data. (e.g. Collaboration policies)
- A subscription is defined by a meta data string which it matches with the meta attributes of new data.
- The subscription generates replication rules on all positively matched data.

Example subscription:

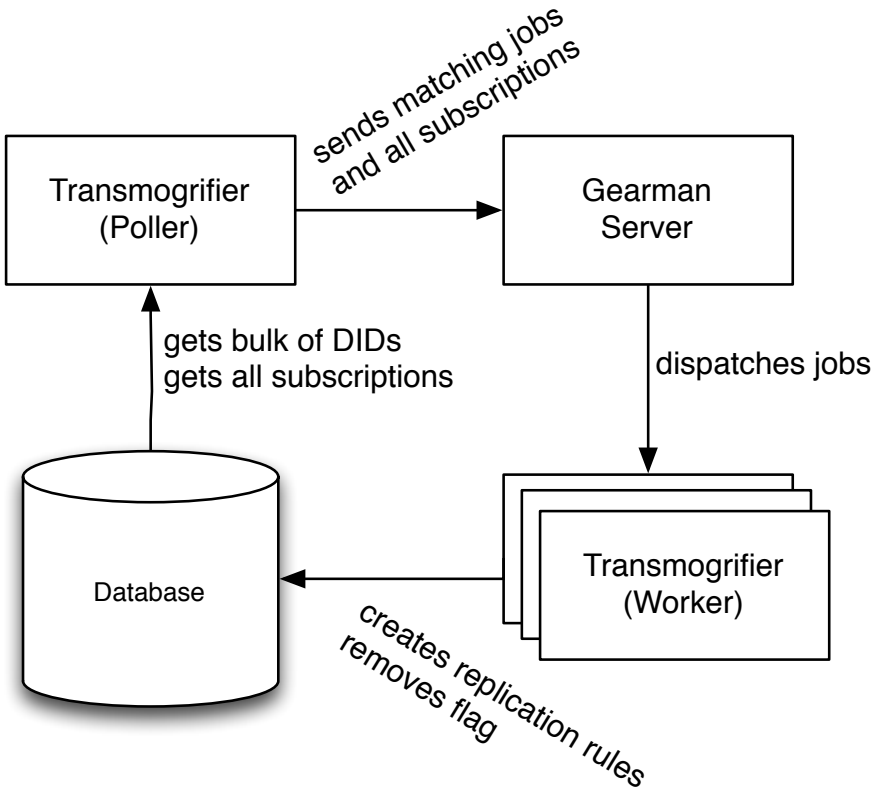
- Match: `project=data12_8TeV,dataType=RAW,stream=physics,DIType=dataset`
- Rule: `2` copies on `tier=2`

Typical subscriptions:

- Tier-0 export
- Group export

Subscriptions – Workflow & Architecture

- New data is marked as `new` when created in Rucio.
- The transmogrifier daemon picks up new data and matches all existing subscriptions with it.
- Gearman is used to balance the load over multiple Transmogrifier workers.



Subscriptions – Evaluation

- Evaluation also based on the emulation framework used for scaling tests.
- Nominal and 2x peak load.
- A set of subscriptions, representing the current ATLAS data distribution policy was used.
- The system had no difficulties processing the subscriptions on new data in time and creating the respective replication rules.

Conclusion

- Rucio is the next generation of the ATLAS distributed data management system.
- Two ways for managing replicas:
  - Replication rules, for existing data,
  - and Subscriptions, for data created in the future.
- Replication rules use RSE expressions to describe the replication intention.
- Gives the system the freedom to select RSEs in a more optimised way.
- Integrating policy workflow into Rucio.
- Subscriptions match meta data with newly created data and automatically create replication rules.
- Ideal for collaboration policies and campaigns.

ATLAS Replica Management in Rucio: Replication Rules and Subscriptions

CHEP 2013

Martin Barisits

CERN PH-ADP, Geneva, Switzerland
&
University of Innsbruck, Innsbruck, Austria

15. October 2013