



Transfers: the new bearings of the Conveyor

Radu Carpa

11 Nov. 2022

Creation of transfer requests in Rucio

```
rucio add-rule bla:dataset1 2 *
```

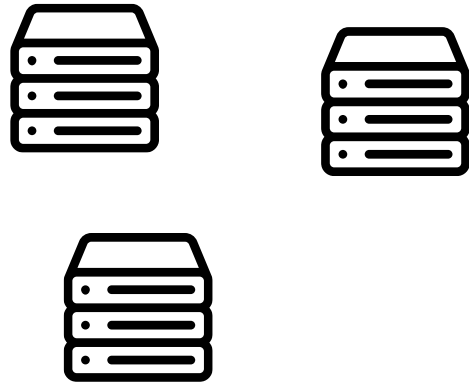
Rucio then picks the destination RSEs (matching the RSE-expression, here: *) for files contained in the dataset.

And schedules the transfer for each file:

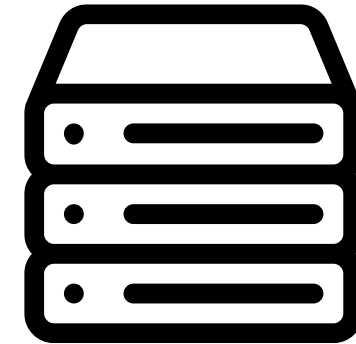
TransferRequest1	TransferRequest2
scope: bla	scope: bla
name: filename1.txt	name: filename2.txt
dst_rse: EU2	dst_rse: EU2
state: Queued	state: Queued
.....

Transfer request handling

Potential Source RSE



Destination RSE



bla:filename1.txt





Ensure that a file is copied to a desired destination

The “conveyor” daemons will:

- Find the best source
- Generate the source and destination URI (protocol://hostname/path)
- Ask *transfertool* to perform the transfer from source(s) URI(s) to destination URI
- Monitor the execution of the actual transfer and retry in case of failures

About transfertools

- **Rucio doesn't execute the transfer. It relies on an external “transfer tool” for that**
-  **FTS**
 - The “main” transfertool used by Rucio (the initial transfer machinery was built “for” FTS)
 - Very mature support in Rucio
-  **Globus Online Transfers**
 - Added by Matt Snyder (BNL) in a collaboration with the Rucio core team
 - The hard work already done and worked very well in recent “production tests” (full transfer lifecycle executed by the ATLAS production rucio instance)
 - Final touches and polishing needed in a production context by somebody who actually uses globus

The minimal transfer machinery

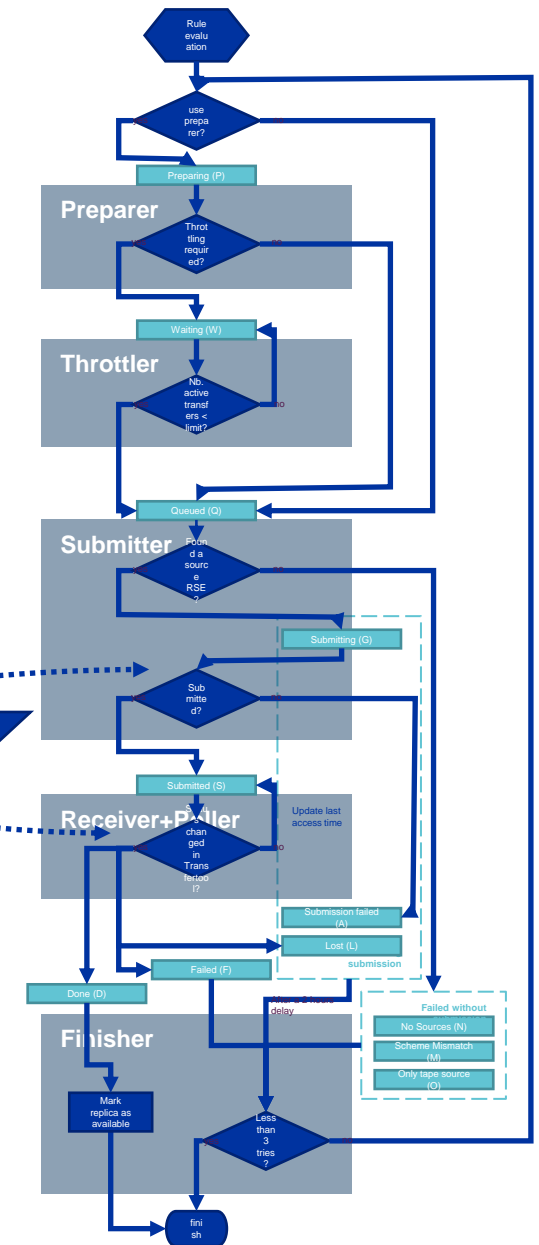
- **3 “conveyor” daemons:**
 - **Submitter:** finds best source RSE and submits jobs to a transfertool
 - **Poller:** regularly polls the transfertool for job state changes
 - **Finisher:** handles finished transfers and re-schedules them if needed

The full transfer machinery

- 6 “conveyor” daemons:

- **Preparer**: Finds the best sources. Selects the transfertool. Decides if the request must be processed by Throttler, or directly by Submitter.
- **Throttler**: Gradually feeds requests to Submitter while respecting administrative constraints (ex: max number of active transfers for an RSE).
- **Submitter**: Submits jobs to the transfertool selected by Preparer.
- **Receiver**: Listens to events sent by FTS.
- **Poller** (1 per transfertool): regularly polls the transfertool for job state changes (in case of FTS, the jobs missed by Receiver)
- **Finisher**: handles finished transfers and re-schedules them if needed

Transfertool
(FTS/Globus)



- [1] <https://rucio.cern.ch/documentation/transfers-overview>

Recent transfer machinery evolution

- **Rucio 1.26**
 - Complete re-write of the Submitter. Many major bug-fixes and some functional changes.
- **Rucio 1.27**
 - Transfertools-agnostic code in the core. Improvements in the Receiver.
- **Rucio 1.28**
 - Introduction of a new database table for multi-hop transfers. Fixing long-standing issues with multi-hop transfers and enabling multi-hop between transfertools.
- **Rucio 1.29**
 - Preparer becomes more intelligent. If used, performs path computation and source selection. Improves multi-transfertools and throttler workflows.
- **Rucio 1.30 (upcoming release)**
 - Throttler should become more usable.

The Preparer

Alternative workflow activated by setting the config value ``conveyor/use_preparer = true`` globally (all servers and daemons). May become mandatory in a future version of Rucio.

Preparer:

- releases submitter from part of its work of source selection;
- uses transfertool-specific RSE attributes (fts/globus_endpoint_id) to decide which transfertool can be used for the transfer;
- decides if throttling is needed thanks to the entries in the “transfer limits” database table.



The Throttler

Used for rate-limiting. Limit how many parallel transfers are allowed to/from an RSE.

Old component. Occasionally used by ATLAS in the past. Was difficult to configure and counter-intuitive to use (ex: automatic removal of rules).

The reworked Throttler

A big part of the rework already available since 1.29.7 and tested in ATLAS:

- No configuration needed anymore. If throttling rules exist in the database, they are automatically applied
- The same throttler instance can do both source[*] and destination throttling

New in 1.30.0:

- Throttling rules per “RSE expression”: quota shared by all matching RSEs.

Note: All existing throttling rules will be deleted on update from previous versions.

[*] source throttling is best-effort (fts reorders “at will” sources in multi-source transfers) and can have bad performance (most probably fixable with an additional database index).

The Receiver

Listens to an ActiveMQ topic for messages from FTS:

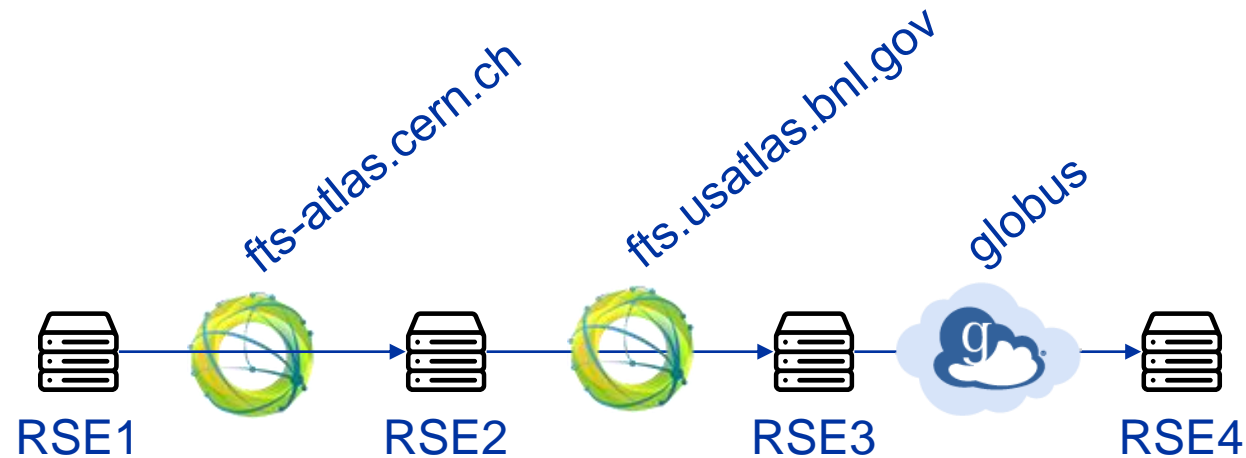
- Drastically improves the speed of marking the transfers successful/failed
- Reduces the load on FTS

Many corner cases fixed in recent Rucio releases. No more reasons to avoid using it!

Recent changes in multi-hop handling

Native “path” relationship in Rucio.

Allows parts of the multi-hop to be handled by a separate transfertool (FTS/Globus).



Summary

The Rucio transfer machinery was almost entirely re-written in the last year

Previously optional components are now highly recommended (preparer, receiver)

Reminder: we should start using Globus



home.cern