Full data management with dCache, Rucio & Co.

For the 12th international dCache workshop

Vincent Garonne

University of Oslo

On behave of the project team

Use Case: Multi-location data management

- A data-intensive instrument generates a high volume of data at one location X on some local dCache instance
- Some researchers need to analyse some data at location Y
  - location Y is in a different country than location X and belongs to a different administrative domain
- Examples of scientific collaborations with such need
  - European X-Ray Laser XFEL, EISCAT_3D, CTA, SKA European Regional Center

Caveat: Simplified use case!

A turnkey data management solution ?

- ➢ Some data management solutions exists already, e.g., LHC community
- ➢ But, they exists on top of an important infrastructure including many services (and technicalities) and can be complex to set up!
- ➢ The motivation of this talk is to ~~prototype~~ offer a complete data management solution running in just a few minutes
- ➢ Complement dCache with Rucio:
  - dCache: Storage (and more)
  - Rucio: Data orchestration (and more)

Rucio in a nutshell (for non-ATLAS people)

- ➢ Rucio provides a complete and generic scientific data management service
  - Catalogue of files, datasets and metadata, transfer and deletion, replication policies, end-users tools to access data
- ➢ For ATLAS, Rucio currently manages ~370 PB
  - ~1 billion files shared over >100 sites and more than 3000 users
- ➢ "1st Rucio Community Workshop", March, this year
  - 90+ persons registered representing more than 14 communities
  - Using: AMS, Xenon1T
  - Evaluating: CMS, SKA, Ligo, icecube, XDC and EGI
  - Docker images: https://hub.docker.com/u/rucio/

Few Rucio concepts

- ➢ Data is federated in a single namespace providing a logical view and transparent access of data across multiple locations
  - Files, datasets and containers (DIDs) are identified by {scope}:{name}
- ➢ An RSE (Rucio Storage Element) is an abstraction of a storage endpoint
  - Each RSE can support multiple data access protocols, e.g., srm, dav, swift
- ➢ RSEs can be tagged to describe quality of service, multi-region, connectivity, ...
  - Key/Value pairs, e.g., country=UK, type=TAPE/SSD
  - Leads to grouping, e.g., all SSDs in Australia

Rucio Replication Policies

- ➢ Replica management is based on replication rules
  - A replication rule defines the number of replicas to be kept on a set of RSEs
- ➢ Example:
  - Account jdoe wants 2 replicas of file user.jdoe:file_001 on any DE TAPE system
  - Description language: `country=de&type=TAPE`
  - Multiple ownership, optimize storage space, minimize number of transfers
- ➢ Support to request replication for future data based on pattern or metadata, e.g., to automate data distribution

Transfer tools

- ➢ The possible Transfer tool implementations are:
  - FTS3 ✓
  - ARC data delivery service ✗ — Bringing a workflow management solution
  - Globus Connect ✘
- ➢ Rucio provides a generic transfer tool API for third party copy
  - Asynchronous interface to any potential third-party tool
  - FTS3 has docker images available

dCache: Features

- ➢ dCache is a well supported storage system following the advances in open and standard storage technologies which offers constantly new features
- ➢ Some new features interesting in this context, cf. yesterday’s talks
  - Inotify's REST based management interface
  - New authentication capabilities, e.g., openid, Macaroons
  - Multi-protocol supports
  - ...

.. and now putting everything together in a demo !

About the demo

- Instructions: https://github.com/vingar/rucio/tree/development/etc/docker/standalone
- Prerequisites:
  - docker-compose
  - Configured dCache instances
  - Grid credentials ☹
- What brings the demo:
  - Rucio: REST, daemons, webui, postgresql
  - FTS: REST, server,webui, mysql
  - ActiveMQ

Workflow: Data registration & Export

1. Some files are uploaded with `rucio-upload` in a directory on a dcache instance NDGF-PIGGY
2. Rules are declared to Rucio
3. Rucio generates the transfers to another remote instance DESY-PROMETHEUS and ensures files are transferred correctly

- ➢ That corresponds to the first use case :)
- ➢ NB: It's also a common administrative operational task to bulk migrate/rebalance data across facilities, e.g., decommissioning, disk to tape

Workflow: dCache & Inotify

1. Files are uploaded in an upload directory with Cyberduck on a dcache instance DESY-DISCORDIA
2. A watcher daemon, Panoptes, waits for inotify events and registers the data in Rucio
3. The files are automatically replicated to another location DESY-PROMETHEUS

- ➢ `rsync` use case with flexibility for the replication, throttling, failover, etc
- ➢ No need for special clients like `rucio upload` to keep the DDM layer in sync with the storage
- ➢ Users can use whatever clients they like, e.g., cyberduck, curl, ...

Example of more advanced workflow

1. Instrument generates data, places it on output buffers and the datasets are registered in Rucio
2. Rucio picks up new file and automatically creates new rule to distribute the file to a data centre following the policy (e.g., online, nearline, archive)
3. Rucio ensures file is transferred correctly and notify when rule is satisfied
4. Rucio automatically removes file on buffer after grace period

➢ We can have more elaborated scenarii wrt workflow management system
- Starting analysis job after rule completion notification
- Pre-filling cache prior to job execution
- Rucio automatically adapts replication factor based on data access
- ...

Going Further ?

- ➢ Working data management solution that you can take, interest ?
- ➢ Good basis to explore new (buzzwords) features and go into details
  - Complement and extend the deployment, i.e., scale, kubernetes, monitoring
  - Capability authentication, e.g., openId, Macaroons
  - QoS semantics: data importance, popularity wrt storage QoS
  - Content sync., dark data
  - Network, monitoring, popularity
  - ?

The END

further reading

www.dCache.org