Rucio

EUCLID CERN Technical Meeting

Mario Lassnig
on behalf of the Rucio team

RUCIO
SCIENTIFIC DATA MANAGEMENT

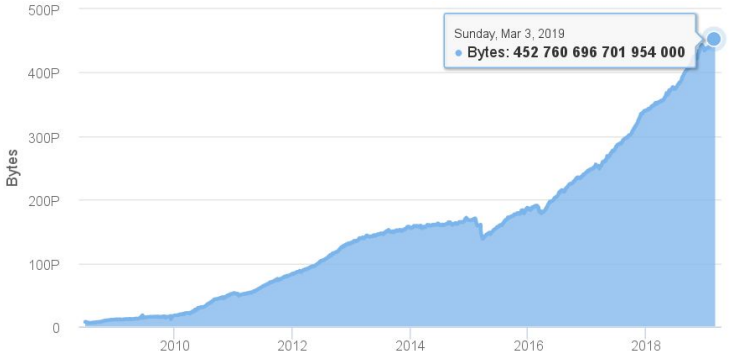
ATLAS
EXPERIMENT

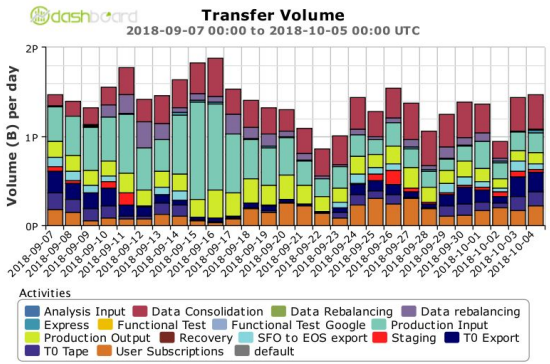
Rucio in a nutshell

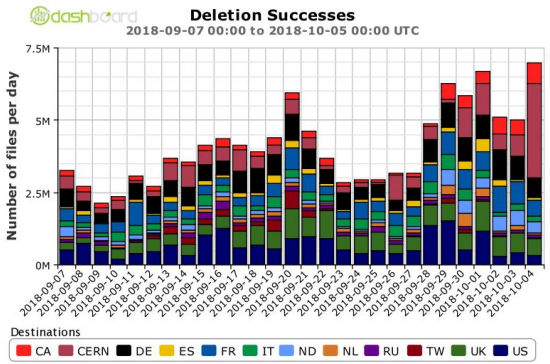
- Rucio provides a complete and generic scientific data management service
  - Seamless integration of scientific and commercial storage and network systems
  - Data can be scientific observations, measurements, objects, events, images saved in files
  - Facilities can be distributed at multiple locations belonging to different administrative domains
  - Designed with more than a decade of operational experience in very large-scale data management
- Rucio manages location-aware data in a heterogeneous distributed environment
  - Creation, location, transfer, deletion, and annotation of data
  - Orchestration of dataflows with both low-level and high-level policies
- Principally developed by and for the High-Energy Physics experiment ATLAS
- Rucio is open source and available under Apache 2.0 license
- Make use of established toolchains for and with the community

Data management for ATLAS

- A few numbers to set the scale
  - 1B+ files, 450 PB of data, 400+ Hz interaction rate
  - Up to 4M files/2.5 PB transferred per day
  - 10PB access from clients per day; >1000 active users
- Increase 1+ order of magnitude for LHC Run 4





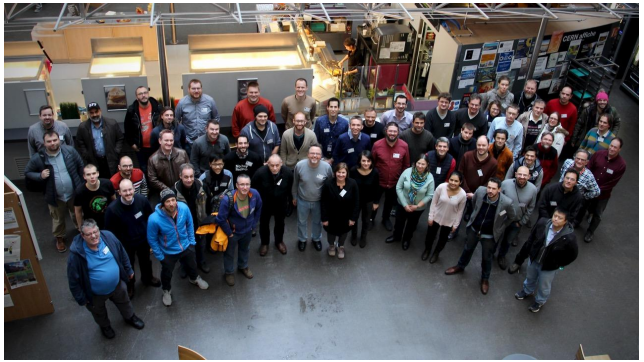


Community


Advanced European Network of E-infrastructures for Astronomy with the SKA
LCLS
Belle II
Science & Technology Facilities Council
COMPASS
CERN
ARC
ATLAS EXPERIMENT
Alpha Magnetic Spectrometer
AMS-02
Europe • Asia
North America
Xe
XENON
Dark Matter Project
CMS
Compact Muon Solenoid
DUNE
DEEP UNDERGROUND NEUTRINO EXPERIMENT
egi
3D
EISCAT
cta
cherenkov telescope array
LSST
Large Synoptic Survey Telescope
IceCube
SOUTH POLE NEUTRINO OBSERVATORY
FTS
eXtreme DataCloud
Fermilab
SKA
SQUARE KILOMETRE ARRAY
LIGO
Open Science Grid
BROOKHAVEN
NATIONAL LABORATORY
EUXDAT

Community

- 1st Rucio Community Workshop was held at CERN on March 1st-2nd 2018 to present Rucio to scientific communities
- Rucio Coding Camp 2018 in November 2018
- 2nd Rucio Community Workshop was held in Oslo on Feb 28th - Mar 1st
- Rucio Coding Camp 2019 in October
- Weekly Development meetings



Rucio main functionalities

- Provides many features that can be enabled selectively
  - File and dataset catalog
  - Transfers between facilities including disk, clouds, and tapes
  - Web-UI, CLI, and API to discover/download/upload/transfer/annotate data
  - Extensive monitoring for all dataflows
  - Support for caches and CDN workflows
  - Expressive policy engines with rules and subscriptions
  - Automated corruption identification and recovery
  - Data popularity based replication
  - …

More advanced features

- Rucio can be integrated with Workload and Workflow Management System
  - Already supporting PanDA, the ATLAS WFMS
  - Belle-II is driving the integration with DIRAC

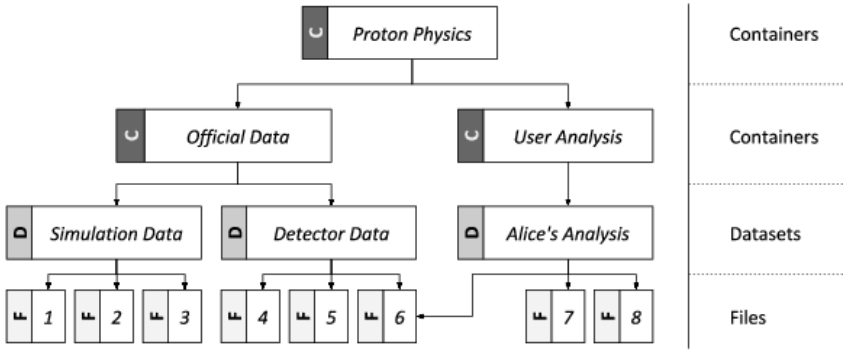
Operations model

- Objective was to minimise the amount of human intervention necessary
- Large-scale and repetitive operational tasks can be automated
  - Bulk migrating/deleting/rebalancing data across facilities at multiple institutions
  - Popularity driven replication and deletion based on data access patterns
  - Management of disk spaces and data lifetime
  - Identification of lost data and automatic consistency recovery
- Administrators at the sites are not operating any Rucio service
  - Sites only operate their storage exposed via protocols (POSIX, ROOT, HTTP, WebDAV, S3, gsiftp, …)
  - Users have transparent access to all data in a federated way
- Easy to deploy
  - PIP packages, Docker containers, Kubernetes

Rucio concepts - Namespace

- All data stored in Rucio is identified by a Data IDentifier (DID)
  - With both fixed and generic metadata support

- There are different types of DIDs
  - Files
  - Datasets Collection of files
  - Container Collection of dataset and/or container
- Each DID is uniquely identified and composed of a scope and name, e.g.:




```
detector_raw.run34:observation_123.root
-------------------------------- ------------------------------------
```

scope name

Rucio concepts - Declarative data management

- Express what you want, not how you want it
  - *e.g., "Three copies of this dataset, distributed evenly across multiple continents, with at least one copy on TAPE"*
- Replication rules
  - Rules can be dynamically added and removed by all users, some pending authorisation
  - Evaluation engine resolves all rules and tries to satisfy them by requesting transfers and deletions
  - Lock data against deletion in particular places for a given lifetime
  - Primary replicas have indefinite lifetime rules
  - Cached replicas are dynamically created replicas based on traced usage and popularity
  - Workflow system can drive rules automatically, e.g., job to data flows or vice-versa
- Subscriptions
  - Automatically generate rules for newly registered data matching a set of filters or metadata
  - e.g., *project=data17_13TeV* and *data_type=AOD* uniformly across *T1s*

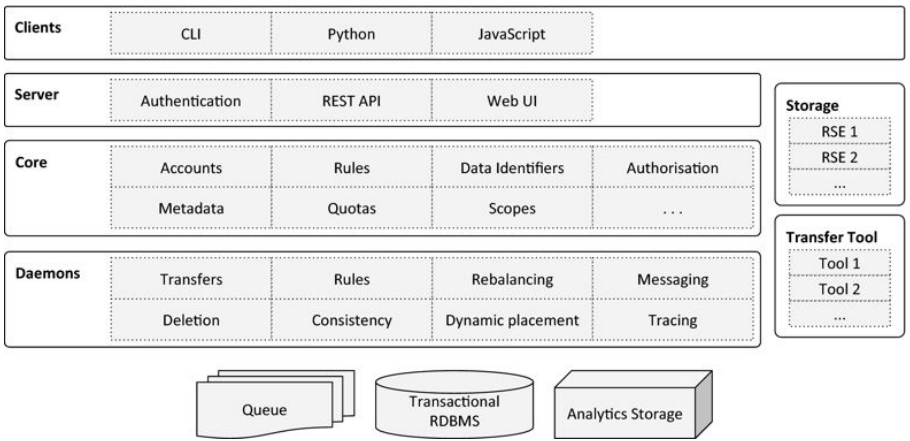
Rucio concepts - RSEs

- Rucio Storage Elements (RSEs) are logical entities of space
  - No software needed to run at the facility except the storage system, e.g., EOS/dCache/S3, …
  - RSE names are arbitrary (e.g., "CERN-PROD_DATADISK", "AWS_REGION_USEAST", …)
  - Common approach is one RSE per storage data class at the site
- RSEs collect all necessary metadata for a storage system
  - Protocols, hostnames, ports, prefixes, paths, implementations, …
  - Data access priorities can be set, e.g., to prefer a different protocol for LAN-only access
- RSEs can be assigned metadata as well
  - Key/Value pairs (e.g., *country=UK*, *type=TAPE*, *support=brian@unl.edu*)
  - You can use RSE expressions to describe a list of RSEs (e.g. `country=UK&type=TAPE`) for the replication rules

Rucio concepts - Metadata

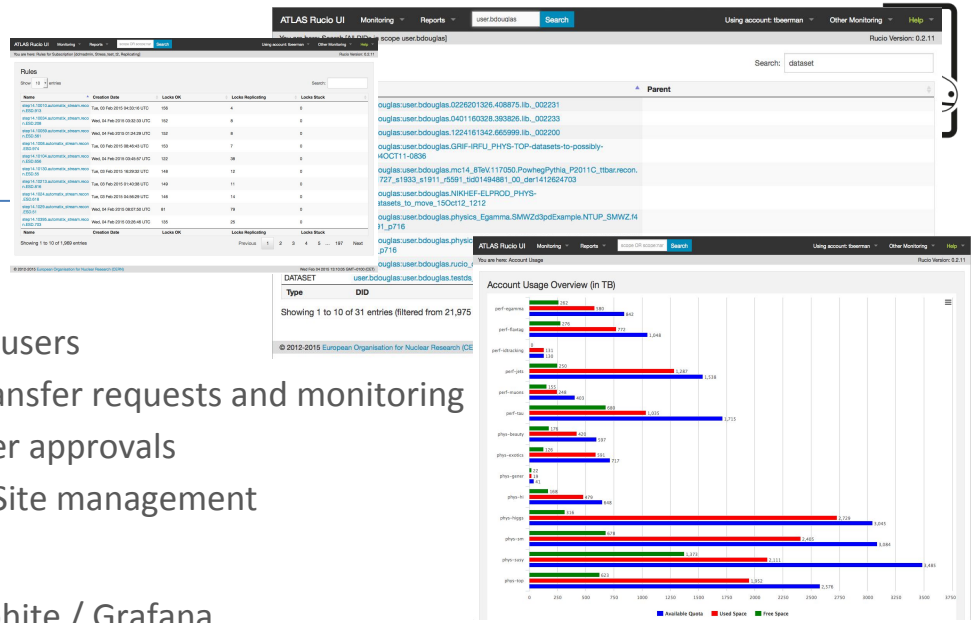
- Rucio supports different kinds of metadata
  - File internal metadata, e.g., *size, checksum, creation time, status*
  - Physics metadata, e.g., *number of events, lumiblock, cross section*
  - Workflow metadata, e.g., which *task or job* produced the file
  - Management internal metadata: necessary for the organisation of data, e.g., *replication factor*
- Metadata are custom attributes on data identifiers
  - Enforcement possible by types and schemas
  - Naming convention enforcement and automatic metadata extraction
- Provides additional namespace to organise the data
  - Searchable via name and metadata
  - Aggregation based on metadata searches
  - Can also be used for long-term reporting, e.g., evolution of particular metadata selection over time

Architecture



- **Servers**
  - HTTP REST/JSON APIs
  - Token-based security (x509, ssh, kerberos, …)
  - Horizontally scalable
- **Daemons**
  - Orchestrates the collaborative work
    e.g., transfers, deletion, recovery, policy
  - Horizontally scalable
- **Messaging**
  - STOMP / ActiveMQ-compatible
- **Persistence**
  - Object relational mapping
  - Oracle, PostgreSQL, MySQL/MariaDB, SQLite
- **Middleware**
  - Connects to well-established products,
    e.g., FTS3, DynaFed, dCache, EOS, S3, …
- **Python**
  - Support for Python2 and Python3

Monitoring & analytics



- RucioUI
  - Provides several views for different types of users
  - Normal users: Data discovery and details, transfer requests and monitoring
  - Site admins: Quota management and transfer approvals
  - Central administration: Account / Identity / Site management
- Monitoring
  - Internal system health monitoring with Graphite / Grafana
  - Transfer / Deletion / … monitoring built on HDFS, ElasticSearch, and Spark
  - Messaging with STOMP
- Analytics and accounting
  - e.g., Show which the data is used, where and how space is used, …
  - Data reports for long-term views
  - Built on Hadoop and Spark

Development

- Release cycle and support period
  - Bi-weekly patch releases (Bugfixes, minor enhancements)
  - ~3 feature (named) releases per year (Features, major changes)
  - Once a year a feature version is designated as a Long-Term Support (LTS) release
- Development organized as open-source community project
  - Weekly development meetings; Release roadmap for each feature release
  - Contributors describe their planned developments, receive comments from community
  - Extensive integration and unit testing across all supported databases

Current developments

- Multi-experiment data management
  - Smart sharing of infrastructure across experiments
  - Smart sharing of data across experiments
- Quality of Service
  - Replication rules can take storage quality descriptions into account
- Expanding support for commercial cloud providers
  - Transparent Google Cloud and S3 (OpenStack, AWS) integration already supported
- Capability-based authentication, authorisation, and permissions
  - Data embargoes, JWT bearer tokens, Macaroons, OpenID, EduGain
- Network interface integration with SDNs
- Integration with research publication databases, e.g., Zenodo

EUCLID and Rucio?

- The scientific value of the data collected by EUCLID goes far beyond cosmology
- Helps out with targets and sources for observatories such as SKA
- Rucio is already seeing widespread adoption in the astro & astroparticle fields

- Can Rucio help
  - with EUCLIDs data management and distribution?
  - connect the astro missions for easy data sharing?

- We are happy to help setting up a demonstrator!

More information

Website	http://rucio.cern.ch
Documentation	https://rucio.readthedocs.io
Repository	https://github.com/rucio/
Images	https://hub.docker.com/r/rucio/
Online support	https://rucio.slack.com/messages/#support/
Developer contact	rucio-dev@cern.ch
Journal article	https://doi.org/10.1007/s41781-019-0026-3
Twitter	https://twitter.com/RucioData