Rucio overview

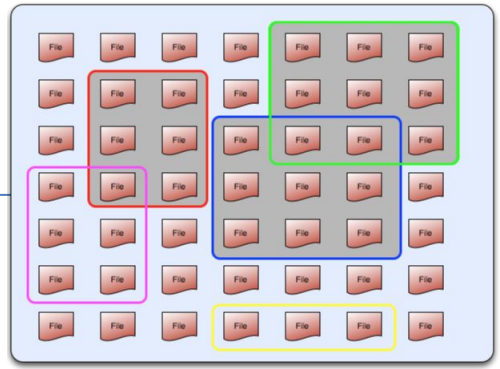
Martin Barisits for the Rucio team
martin.barisits@cern.ch

Rucio in a nutshell

- Initially developed by the High-Energy Physics experiment ATLAS
- Rucio provides a complete and generic scientific data management service
  - 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 10 years of operational experience in large-scale data management!

- Rucio manages multi-location data in a heterogeneous distributed environment
  - Creation, location, transfer, and deletion of replicas of data
  - Orchestration according to both low-level and high-level driven data management policies (usage policies, access control, and data lifetime)
  - Interfaces with workflow management systems
  - Supports a rich set of advanced features, use cases, and requirements

Namespace handling



- Data is federated in a single namespace
  - Ensure transparent access across multiple locations
- Data Identifier (DID) is the primary addressable unit
  - DIDs can be either files, collections (*datasets*), or collections of collections (*containers*)
  - Datasets only hold files, containers only hold datasets
- DIDs are standalone and partitioned
  - Files do not need to be in a dataset
  - Datasets do not need to be in a collection
- DIDs are globally unique
  - Files cannot have the same name as collections, and vice versa
  - Prevent reuse of modified files for consistently repeatable science results
- Collections can be organised freely
  - Files can be in multiple datasets, datasets can be in multiple containers

Metadata support

- We support different kinds of metadata
  - System-defined, e.g., size, checksum, creation time, status
  - Physics, e.g., number of events, lumiblock
  - Production, e.g., which task or job produced the file
  - Data management internal: necessary for the organisation of data, e.g., replication factor
- Metadata are custom attributes on data identifiers
  - Enforcement possible by type, e.g., enum
  - 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)

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 based on data access patterns
  - Popularity driven deletion
  - Management of disk spaces and data lifetime
  - Identification of lost data and automatic consistency recovery
- Administrators at the sites are not operating any local Rucio service
  - Sites only operate their storage
  - Users have transparent access to all data in a federated way
- Easy to deploy
  - Pip packages, Docker containers, Kubernetes

Storage abstraction

- Rucio Storage Elements (RSEs) are a logical entity of space
  - RSE names are arbitrary (e.g., "CERN-PROD_DATADISK", "AWS_REGION_USEAST", …)
  - No storage vendor/product lock-in — Can follow the market: EOS, dCache, DPM, DynaFed, S3, etc.
  - Can be dynamically added and removed, e.g., for temporarily available caching storage
- RSEs collect all necessary metadata for a storage
  - Protocols, hostnames, ports, prefixes, paths, implementations, authentication, …
- Support for adaptive physical representation of files
  - Function-based for horizontal scalability and namespace optimisation
  - Directly specified by the client for advanced use cases (instruments, existing data, …)
- RSEs can be tagged to describe quality of service, multi-region, accessibility, …
  - Key/Value pairs (e.g., *country=UK*, *type=TAPE/SSD*, *support=brian@unl.edu*)
  - Leads to implicit grouping as necessary (e.g, all SSDs in Australia)

Declarative data management 1/2

- Policy-based lifecycle management
- Express what you want with rules and subscriptions. Examples:
  - **Rules**

    "Three copies of this dataset, distributed evenly across three institutes on different continents, with two copies on DISK and one on TAPE"
  - **Subscriptions**

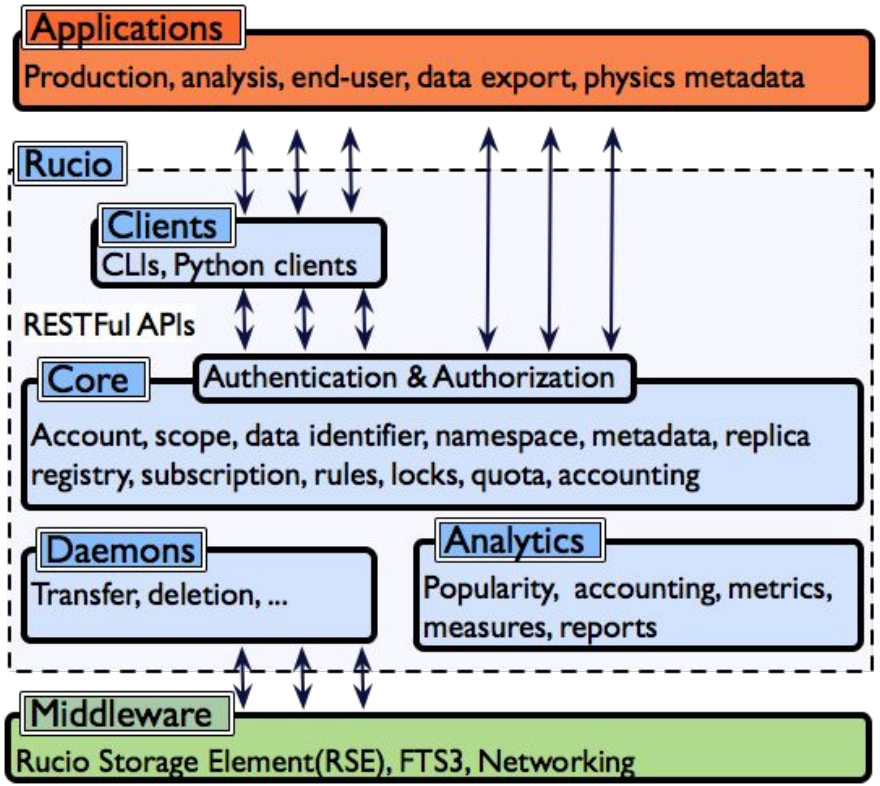
    "Three copies (two on DISK, one on TAPE) of every new DIDs that will be produced that match the set of metadata datatype=RAW and scope=data"
- Support for different data replication policies, e.g.,
  - Archive: difficult/expensive to recreate data
  - Primary cache: data that should be readily available, job inputs/outputs, …
  - Secondary cache: extra replicas created and deleted based on system usage for performance

Declarative data management 2/2

- Rules allow a fully dynamic and automated data distribution
  - Rules can be dynamically added and removed by all accounts, some pending authorisation
  - Rucio constantly evaluates all rules and tries to satisfy them
  - Rules enforce data lifecycles with lifetimes (e.g., automatically delete temporary data after a week)
  - Rules enforce user and group quotas (e.g, 50 PB globally for a physics group, 10 extra PB at a site)

- Description language is set theory complete with additional metadata filters, e.g.,
  - `((continent=eu)&type=SSD)\country=uk`
- Grouping option, e.g., express data operational transfers blocks
- Support for weighted data distributions, e.g., send 80% of data to reliable sites

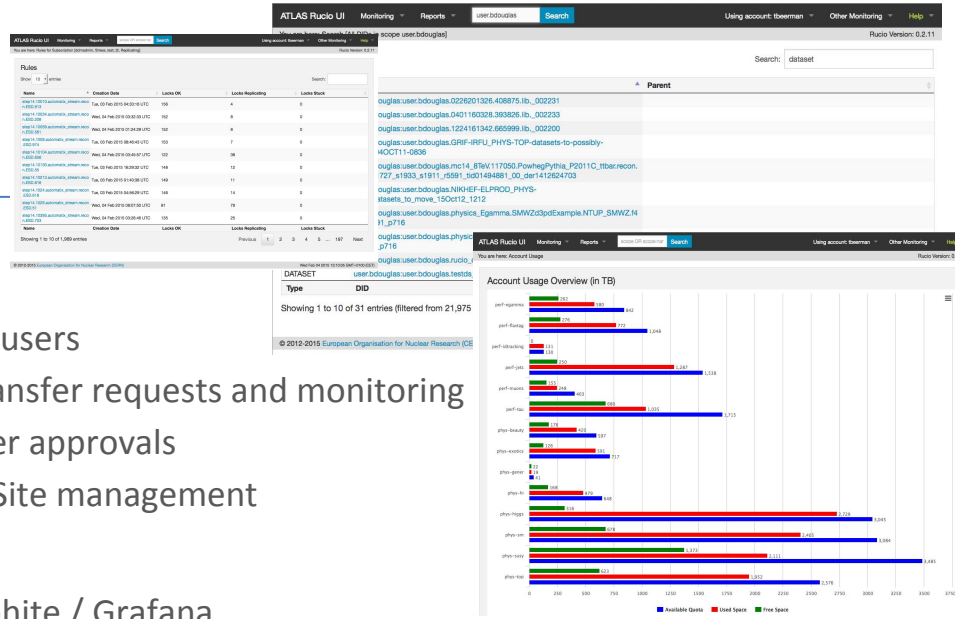
Architecture : Clients, Servers & Daemons



Fully built on open standards and frameworks!

- **Servers**
  - HTTP REST/JSON APIs
  - Token-based authentication (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**
  - Clients: 2.6, 2.7, 3
  - Server: 2.7, (3 coming soon)

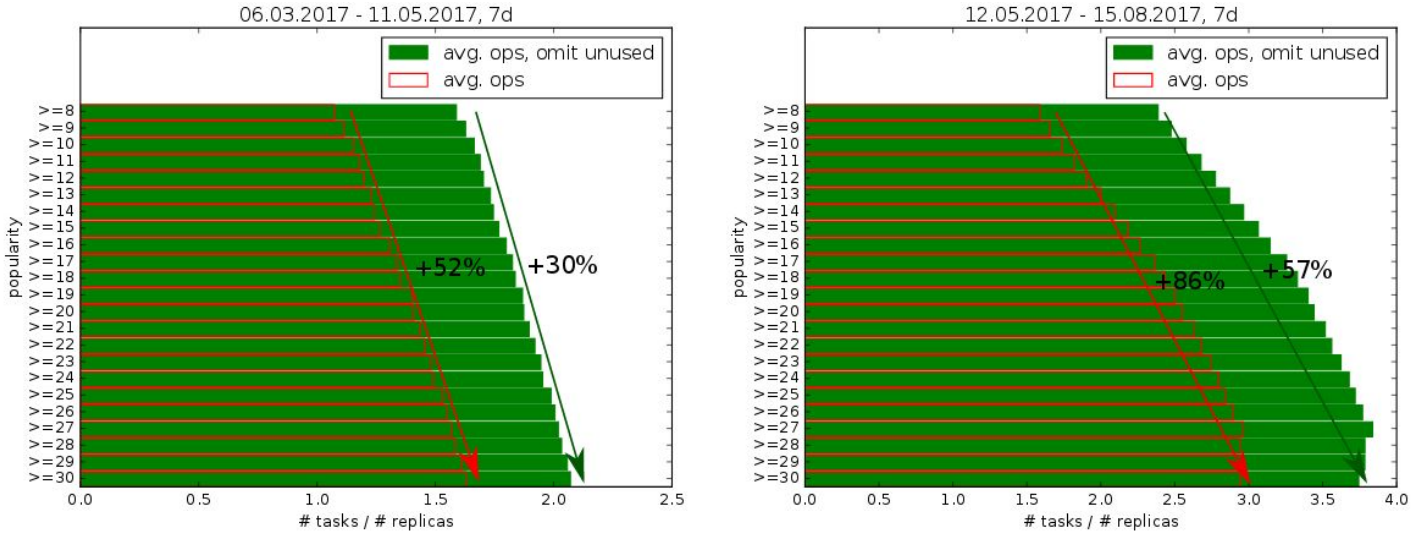
Monitoring & analytics



- 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

Data access patterns

- Every data access produces a trace that is recorded by Rucio
- Data access patterns are analysed and new rules are created automatically
- Create new copies of files on fast storage, subject to network queues



Rucio Community

Google Cloud Platform

- Used in production by ATLAS, AMS, Xenon1T
  - From very large scale (ATLAS: 365 Petabytes, 1 Billion files)
  - To small-scale (Xenon1t: 5.6 Petabytes, 100k files)
- Under evaluation by many others
- First Community Workshop happened in March
- Core development team: 10 (part-time) ~5 FTE
  - + external contributions

AMS-02 · CMS · ATLAS EXPERIMENT · Xe XENON Dark Matter Project · LIGO · NA62 · Open Science Grid · DUNE DEEP UNDERGROUND NEUTRINO EXPERIMENT · SKA · Aeneas Advanced European Network of E-infrastructures for Astronomy with the SKA · cta cherenkov telescope array · COMPASS · 3D EISCAT · IceCube SOUTH POLE NEUTRINO OBSERVATORY · ARC

DIRAC + Rucio

- Potential to offer VOs an integrated experience who wish to use DIRAC as a WM system and Rucio as a DDM system
- Could benefit both communities
- Need to understand how concepts of the two systems fit together
- Some manpower available on Rucio side to make changes (if necessary) to allow a smooth DIRAC interconnection
  - Python or REST APIs, Message callbacks to DIRAC, Download clients for data staging with the pilot, etc.
- Would need interested partners on DIRAC side to make this possible
  - Quick to establish an integration playground (Docker/Kubernetes deployment)

More information

Website	http://rucio.cern.ch
Documentation	https://rucio.readthedocs.io
Repository	https://github.com/rucio/
Continuous Integration	https://travis-ci.org/rucio/
Images	https://hub.docker.com/r/rucio/
Online support	https://rucio.slack.com/messages/#support/
Developer contact	rucio-dev@cern.ch