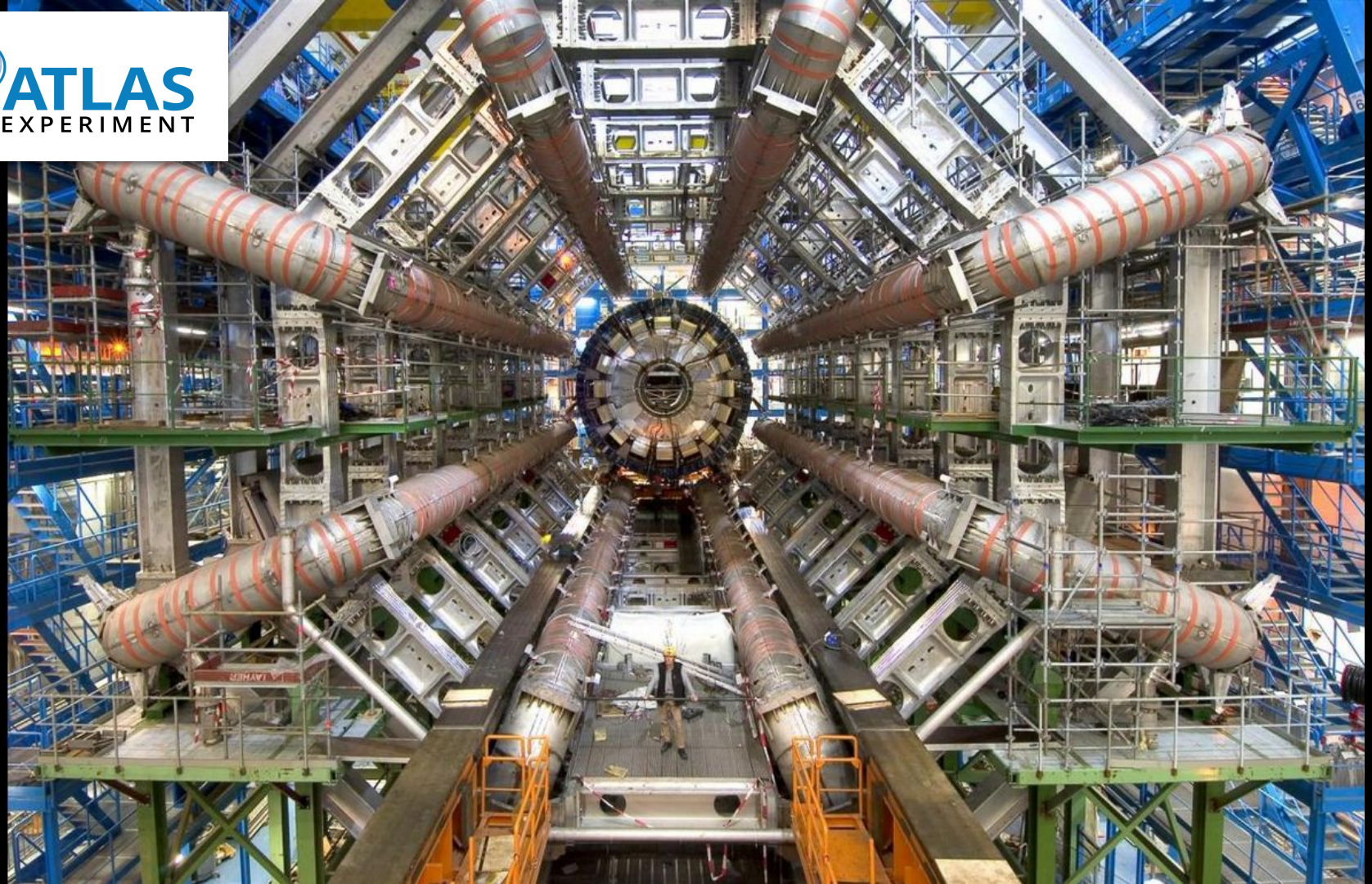
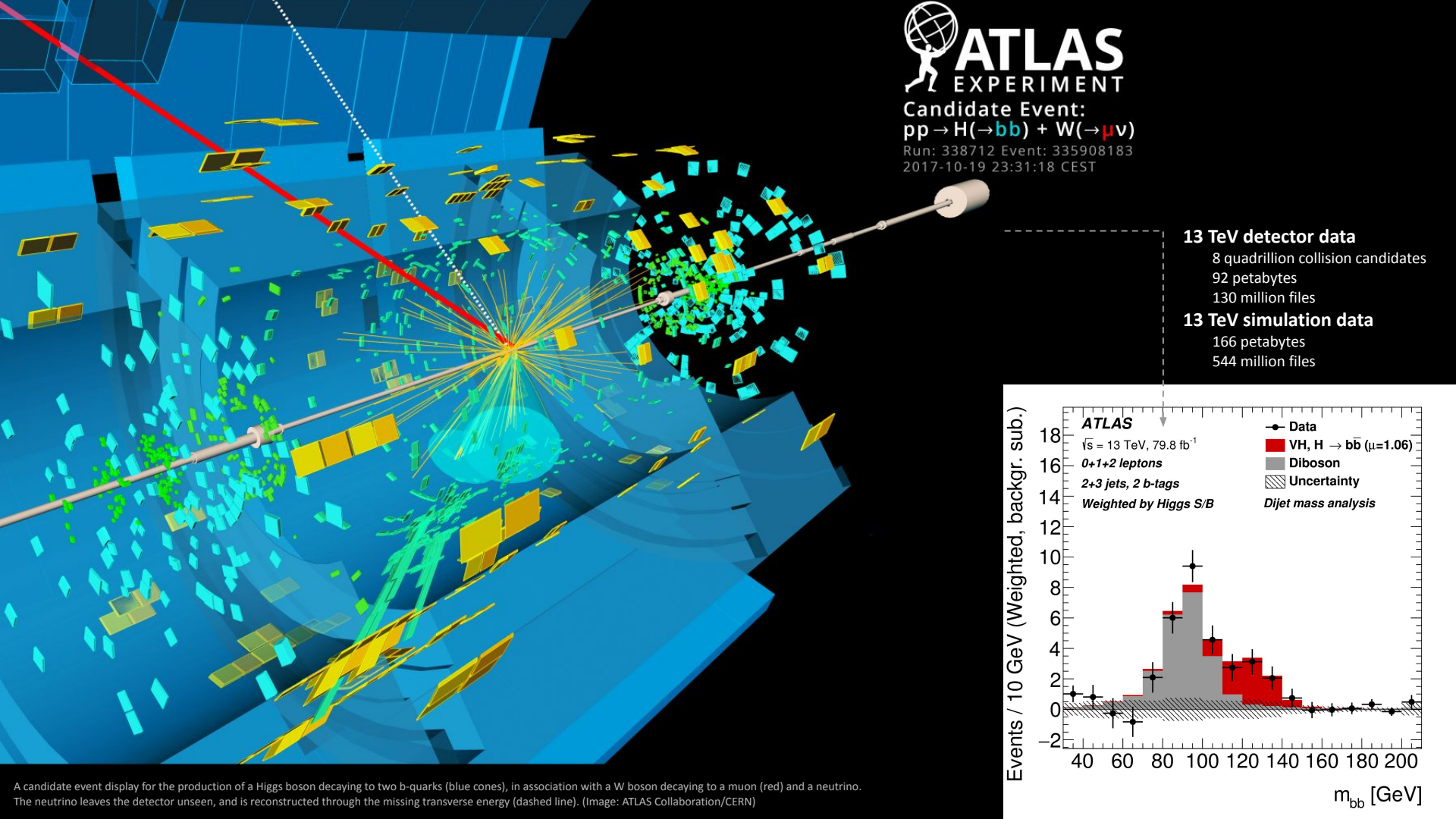
Scientific data management with Rucio

Mario Lassnig, Martin Barisits, Markus Elsing

Rucio talk at ALMA Headquarter in Santiago

22 March 2023





A candidate event display for the production of a Higgs boson decaying to two b-quarks (blue cones), in association with a W boson decaying to a muon (red) and a neutrino. The neutrino leaves the detector unseen, and is reconstructed through the missing transverse energy (dashed line). (Image: ATLAS Collaboration/CERN)

Rucio in a nutshell

Rucio provides a mature and modular scientific **data management federation**

- **Seamless integration** of **scientific and commercial** storage and their network systems
- Data is stored in a **global unified namespace** and can contain **any potential payload**
- Facilities can be **distributed at geographically independent locations** belonging to **different administrative domains**
- Designed with **more than a decade of operational experience** in very large-scale data management

Rucio is location-aware and manages data in a heterogeneous distributed environment

- Creation, location, transfer, deletion, annotation, and access
- **Orchestration of dataflows** with both low-level and high-level policies

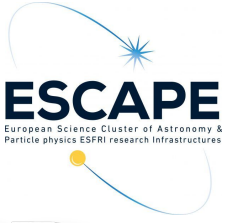
Principally developed by and for the ATLAS Experiment, now with many more communities

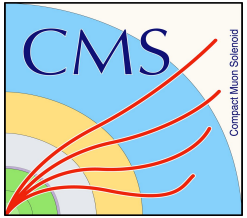
Rucio is **free and open-source software** licenced under *Apache v2.0*

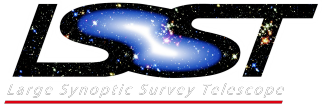
Open **community-driven** development process

Community

Science & Technology Facilities Council





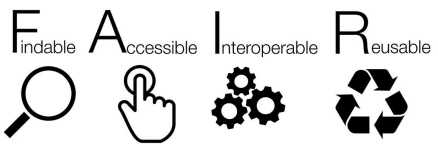


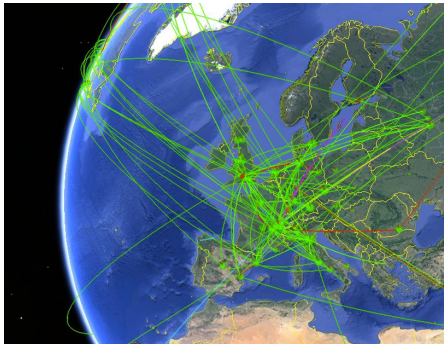
Rucio main functionalities

Provides many features that can be enabled selectively

More advanced features

- **Horizontally scalable catalog** for files, collections, and metadata
- Transfers between facilities including **disk, tapes, clouds, HPCs**
- **Authentication and authorisation** for users and groups
- **Many interfaces** available, including CLI, web, FUSE, and REST API
- **Extensive monitoring** for all dataflows
- Expressive **policy engine** with rules, subscriptions, and quotas
- Automated **corruption identification and recovery**
- Transparent support for **multihop, caches, and CDN dataflows**
- **Data-analytics based flow control**





Rucio is not a distributed file system, it **connects existing storage infrastructure** over the network

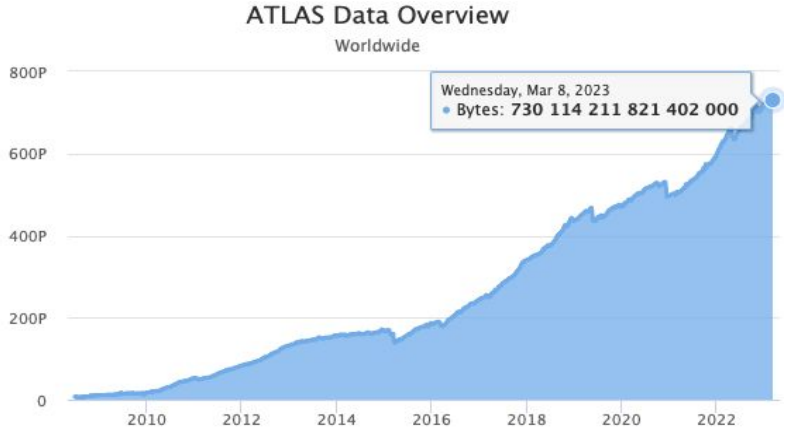
- No Rucio software needs to run at the data centres **(!)**
- Data centres are free to choose which storage system suits them best - **No Vendor Lock-In**

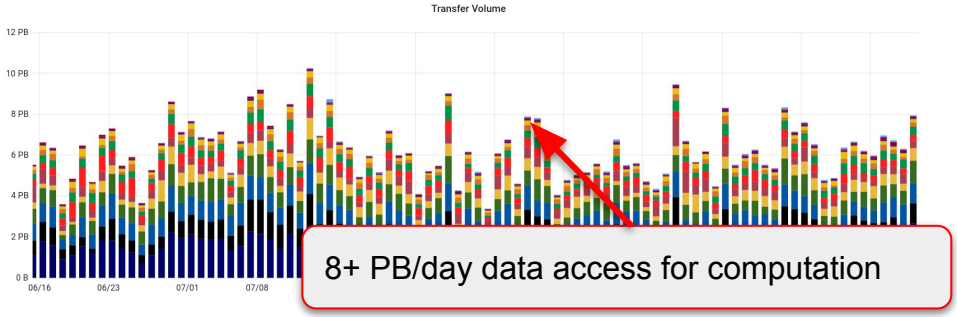
Scale

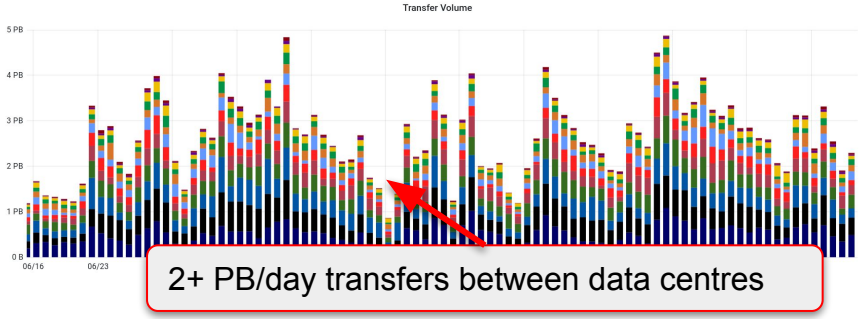
A few numbers showing the **ATLAS scale**

- 1B+ files, 700+ PB of data, 400+ Hz interaction
- 120 data centres, 5 HPCs, 2 clouds, 1000+ users
- 500 Petabytes/year transferred & deleted
- 2.5 Exabytes/year uploaded & downloaded

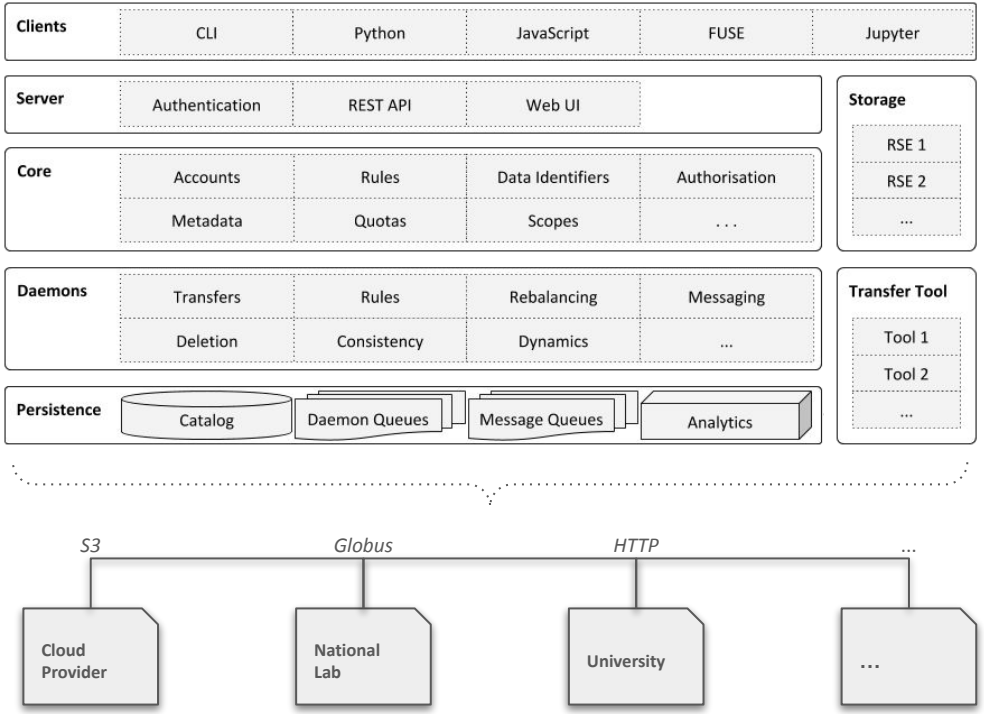
Will increase 1+ order of magnitude for HL-LHC







High-Level Architecture



Horizontally scalable component-based architecture

Servers interact with users

- HTTP API using REST/JSON
- Strong security (X.509, SSH, GSS, OAuth2, …)
- Many client interfaces available

Daemons orchestrate the collaborative work

- Transfers, deletion, recovery, policy, …
- Self-adapting based on workload

Messaging support for easy integration

- STOMP / ActiveMQ-compatible protocol

Persistence layer

- Oracle, PostgreSQL, MySQL/MariaDB, SQLite
- Analytics with Hadoop and Spark

Middleware

- Connects to well-established products, e.g., **FTS3, XRootD, dCache, EOS, Globus, …**
- Connects commercial clouds (**S3, GCS, AWS**)

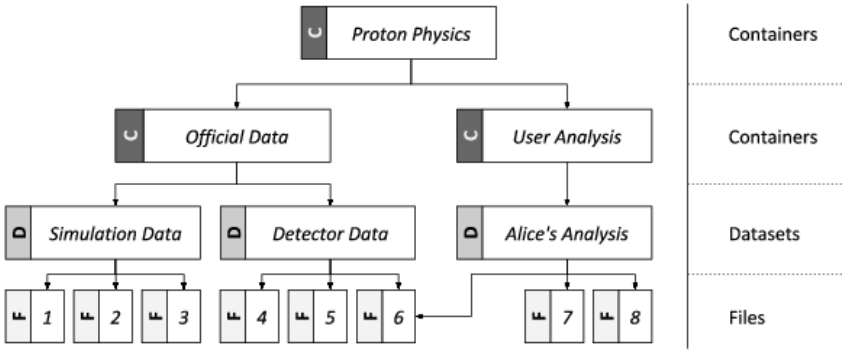
Rucio concepts - Namespace

All data stored in Rucio is identified by a Data IDentifier (DID)

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

Declarative data management

Express what you want, not how you want it

e.g., *"Three copies of this dataset, distributed across MULTIPLE CONTINENTS, with at least one copy on TAPE"*

e.g., *"One copy of this file ANYWHERE, as long as it is a very fast DISK"*

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

Cached replicas are **dynamically created replicas** based on traced usage over time

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., *"All derived products from this physics channel must have a copy on TAPE"*

Operations model

Objective was to **minimise** the amount of **human intervention** necessary

Large-scale and repetitive operational tasks can be **automated**

- **Bulk operations**, such as migrating/deleting/rebalancing data across facilities at multiple institutions
- **Popularity driven** replication and deletion based on **data access patterns**
- Management of **storage spaces** and **data lifetime**
- Identification of **lost data** and automatic **consistency and recovery**

Administrators at the sites are not operating any Rucio service

- Sites only operate storage and **expose common protocols** (`file://, root://, https://, davs://, s3://, …`)
- Users have **transparent access to all data** even without Rucio

Easy to deploy and monitor

- **Python** PIP packages, **Docker** containers, **Kubernetes**
- Integration with common **monitoring and analytics stacks** (e.g., ElasticSearch, Hadoop, Spark, ActiveMQ, …)

Metadata

Rucio supports different kinds of metadata

File **internal metadata**, e.g., *size, checksum, creation time, status*

Fixed **physics metadata,** e.g., *number of events, lumiblock, cross section, …*

Generic metadata that can be set by the users

Searchable via name and metadata, **aggregation** based on metadata searches

Metadata interfaces

Allow Rucio to be connected to **different metadata backends** (json-column, mongodb, external, …)

Metadata queries against Rucio are **relayed** to the matching backend and aggregated

Generic metadata can be restricted

Enforcement possible by **types and schemas**

Naming convention **enforcement** and automatic metadata **extraction**

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

Command Line Interfaces and Python APIs

More ways to interact with Rucio and the data

Rucio Documentation Python Client API REST API

- Welcome
- Getting Started
- User
  - Setting Up the Rucio Client
  - Using the Client
  - Using the Admin Client
  - Python Client API
  - Developing with Rucio
- Operator
- Developer
- About Us

Using the Client

Rucio provides several commands for the end-user. See executables.

Getting user information

The first thing you might try is to check who you are:

```
$ rucio whoami
status     : ACTIVE
account    : jdoe
account_type : SERVICE
created_at : 2014-01-17T07:52:18
updated_at : 2014-01-17T07:52:18
suspended_at : None
deleted_at : None
email      : jdoe@blahblah.com
```

You can switch between different accounts by setting the `RUCIO_ACCOUNT` variable:

```
$ export RUCIO_ACCOUNT=root
$ rucio whoami
status     : ACTIVE
account    : jdoe
account_type : SERVICE
created_at : 2014-01-17T07:51:59
updated_at : 2014-01-17T07:51:59
suspended_at : None
deleted_at : None
email      : root@blahblah.com
```

If you try to authenticate with an account that is not mapped with your credentials:

```
$ export RUCIO_ACCOUNT=janedoe
```

Rucio Documentation Python Client API REST API

- Welcome
- Getting Started
- User
  - Setting Up the Rucio Client
  - Using the Client
  - Using the Admin Client
  - Python Client API
  - Developing with Rucio
- Operator
- Developer
- About Us

User > Using the Admin Client

Using the Admin Client

Rucio provides a CLI for administrative tasks. The get methods can be executed by any user, but the set methods require some admin privileges. See the rucio-admin help page.

Account and identity methods

To create a new account::

```
$ rucio-admin account add --type USER --email jdoe@blahblih.com jdoe
```

You can choose different types in the list USER, GROUP, SERVICE. Different policies/permissions can be set depending on the account type. Once the account is created, you need to create and attach an identity to this account::

```
$ rucio-admin identity add --type X509 --id "/DC=blah/DC=blih/OU=Organic \
  Units/OU=Users/CN=jdoe" --email jdoe@blahblih.com --account jdoe
```

The list of possible identity types is X509, GSS, USERPASS, SSH::

```
$ rucio-admin account list-identities jdoe
Identity: /DC=blah/DC=blih/OU=Organic Units/OU=Users/CN=jdoe,        type: X509
```

You can set attributes to the users::

```
$ rucio-admin account add-attribute --key country --value xyz jdoe
```

Open "rucio.cern.ch/documentation/setting_up_the_rucio_client" in a new tab

Python Client API

- AccountClient
- AccountLimitClient
- BaseClient
- Client
- ConfigClient
- CredentialClient
- DIDClient
- DiracClient
- DownloadClient
- ExportClient
- FileClient
- ImportClient
- LifetimeClient
- LockClient
- MetaClient
- PingClient
- ReplicaClient
- RequestClient
- RSEClient
- RuleClient
- ScopeClient
- SubscriptionClient
- TouchClient
- UploadClient

Community-driven development

We have successfully moved to **community-driven development**

- Requirements, features, issues, release are **publicly discussed** (e.g., weekly meetings, GitHub, MM)
- **Component leads** (core team) coordinate contributions from the community **design / guidance / review**
- Usually 1-2 persons from a **community take responsibility** for a contribution
- to **develop** the software extension and also its **continued maintenance**

Communities are helping each other **across experiments**

- Effective across time zones due to worldwide involvement
- Automation and containerisation of development and deployment **lowers barrier of entry** for newcomers

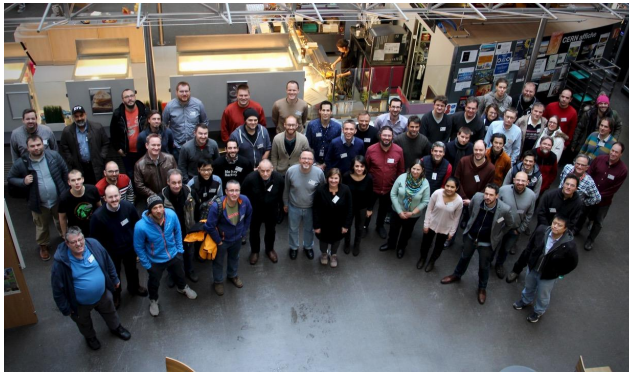
GitHub docker kubernetes

Regular events

Rucio Community Workshops

- CERN, Switzerland [2018]
- University of Oslo, Norway [2019]
- Fermilab, USA [2020]
- Zoom [2021]
- Lancaster University, UK [2022]

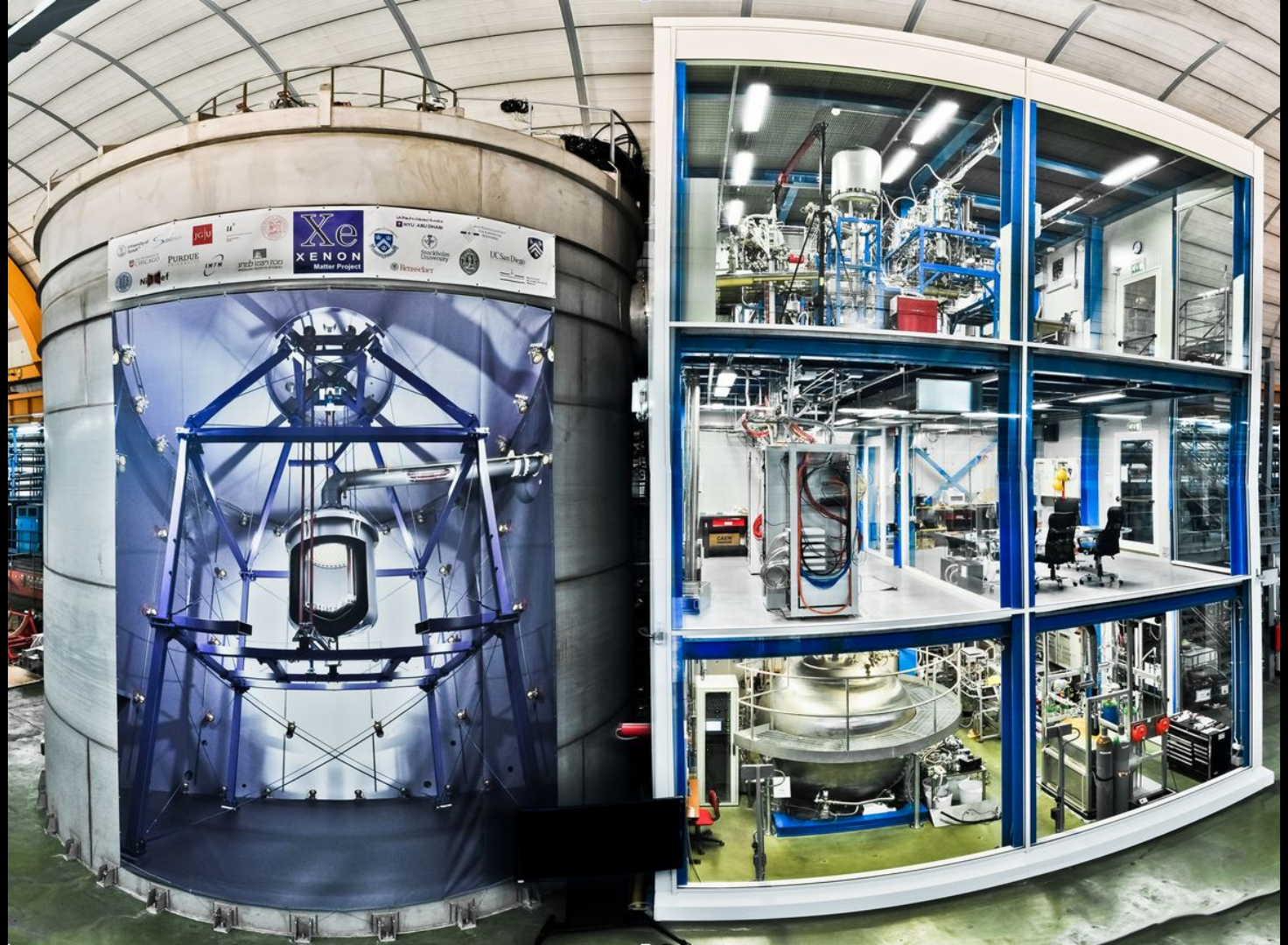
Upcoming 2023 Workshop at KEK, Japan, 16 – 20 October 2023



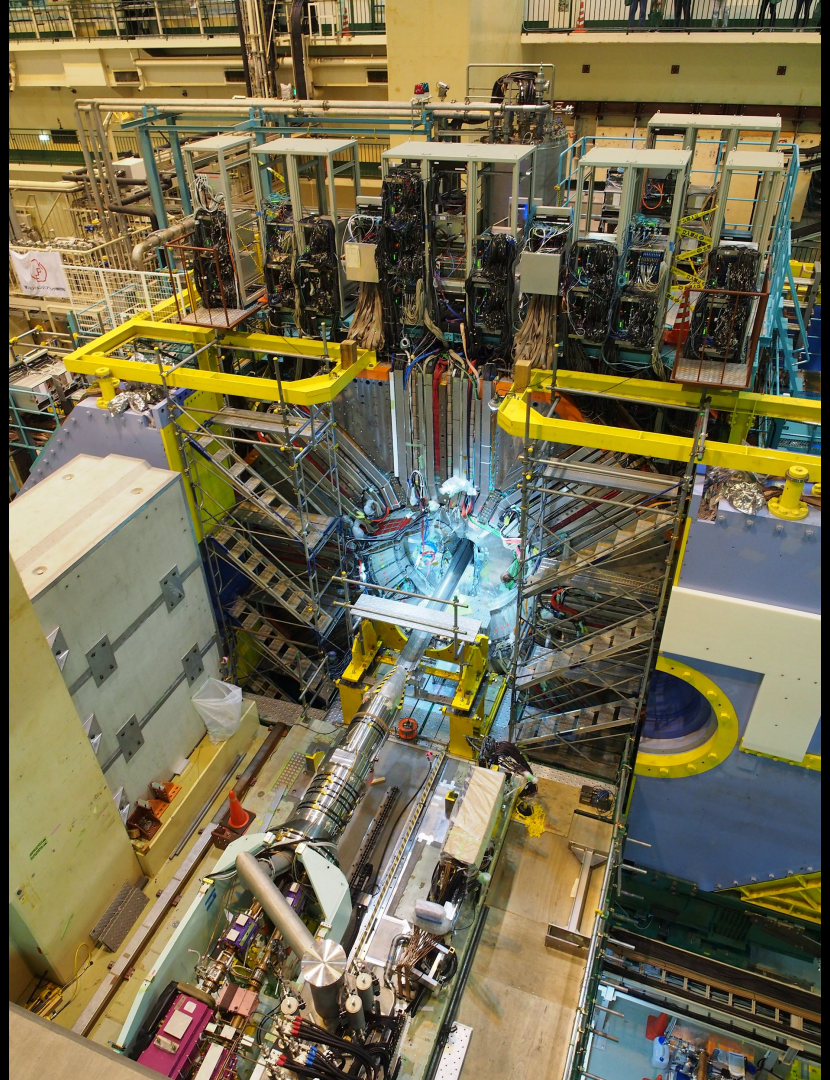
DiCOS

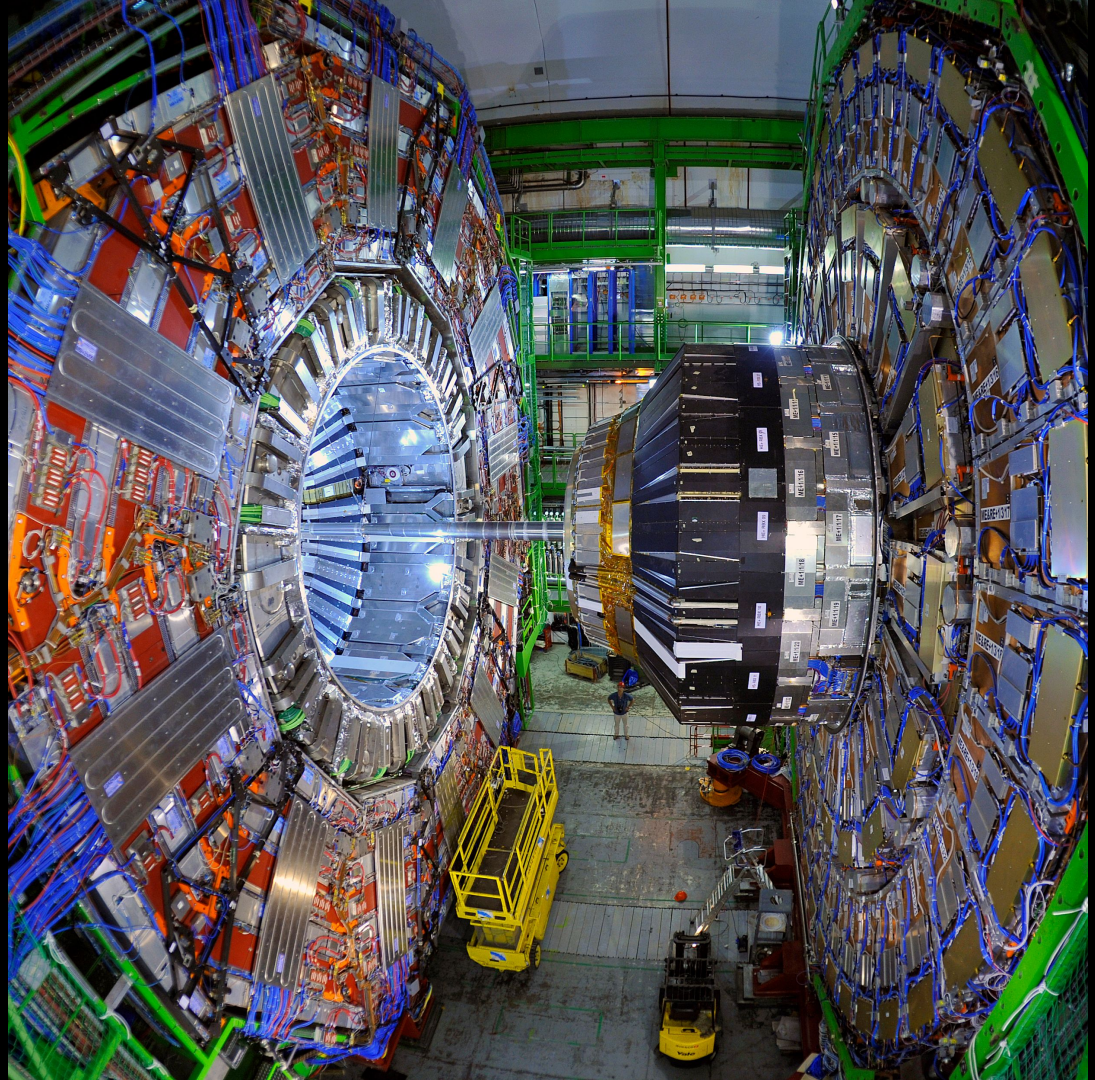
Distributed Cloud Operating System

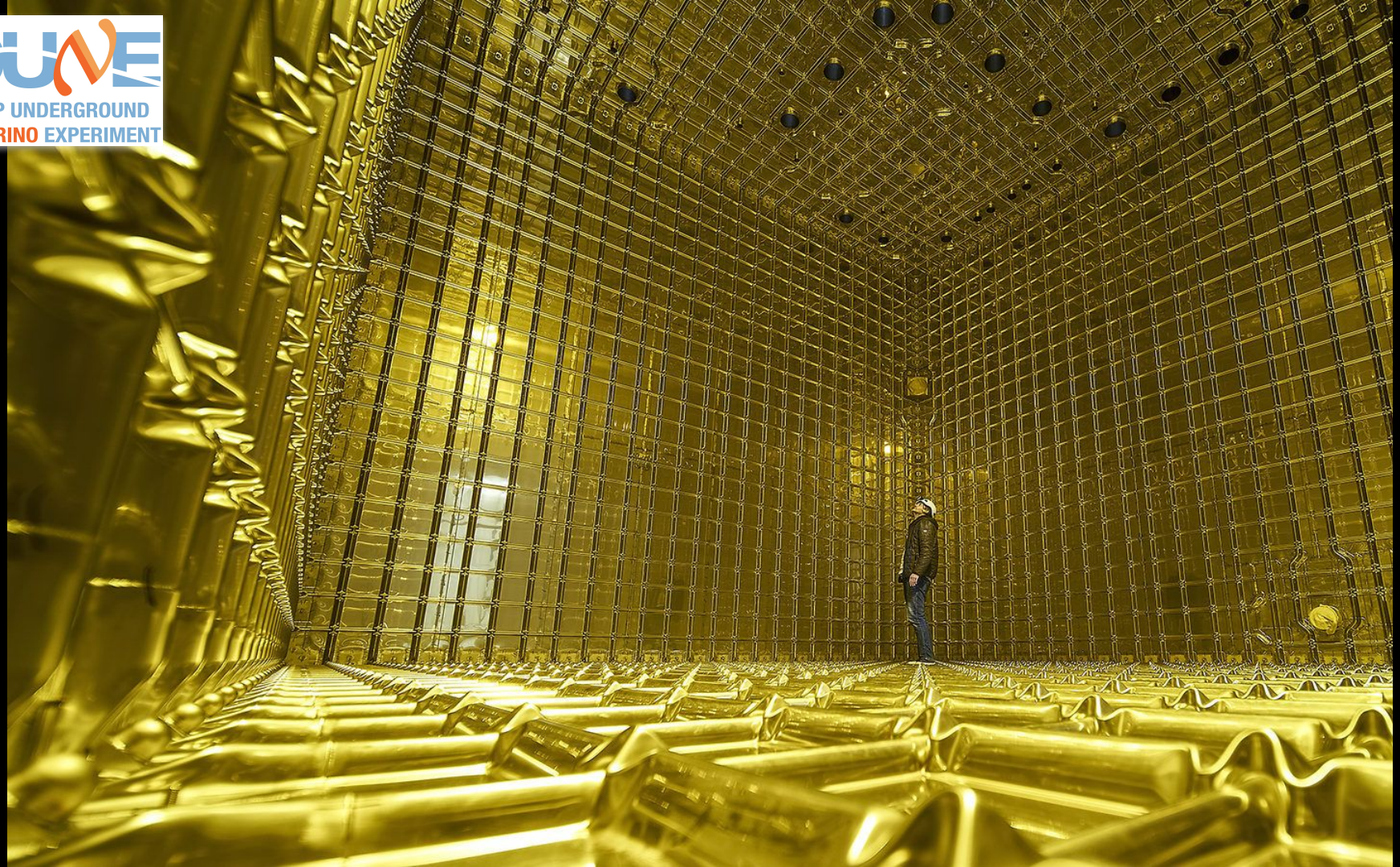
Xe
XENON
Dark Matter Project



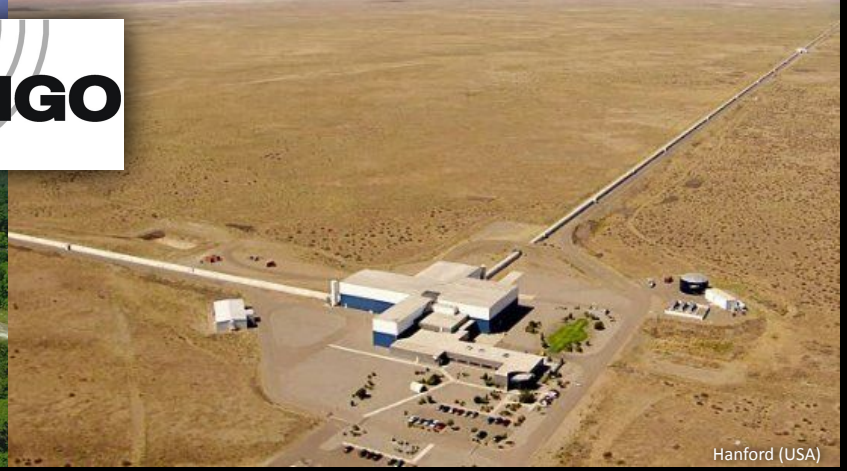
Belle II






DUNE
DEEP UNDERGROUND
NEUTRINO EXPERIMENT

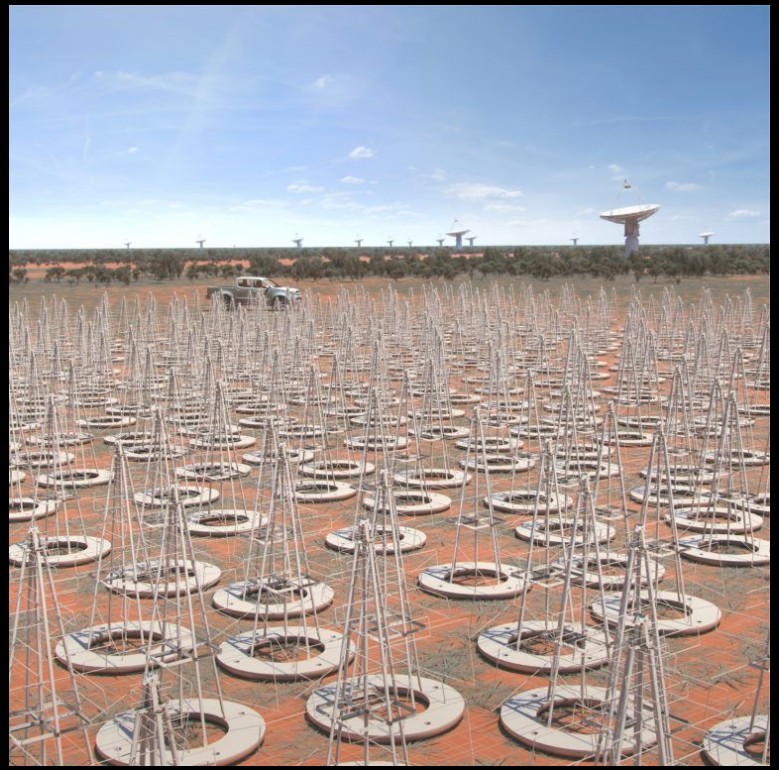
Livingston (USA)



Hanford (USA)

Cascina (IT)

SKA
SQUARE KILOMETRE ARRAY



SKAO REGIONAL CENTRE NETWORK

SKA Regional Centres

Dr Rosie Bolton

Head of Data Operations group, SKAO

7/11/2022

SKA Observatory context

- Global collaboration of 16 countries to build and operate next-generation radio astronomy observatory
- SKAO Inter-Governmental Organisation governed by treaty (d.o.b. 4/2/2021)
- 7-8 year construction schedule. Cost ~€2B (2021 euros) for first 10 years
- Internal data rates up to 2Pb/s
- Output data product rate ~100Gb/s per site: ~700PB per year

SKA Regional Centre Capabilities

Science Enabling Applications
Analysis Tools, Notebooks, Workflows execution Machine Learning, etc

Distributed Data Processing
Computing capabilities provided by the SRCNet to allow data processing

Data Discovery
Discovery of SKA data from the SRCNet, local or remote, transparently to the user

Visualization
Advanced visualizers for SKA data and data from other observatories

Support to Science Community
Support community on SKA data use, SRC services use, Training, Project Impact Dissemination

Interoperability
Heterogeneous SKA data from different SRCs and other observatories

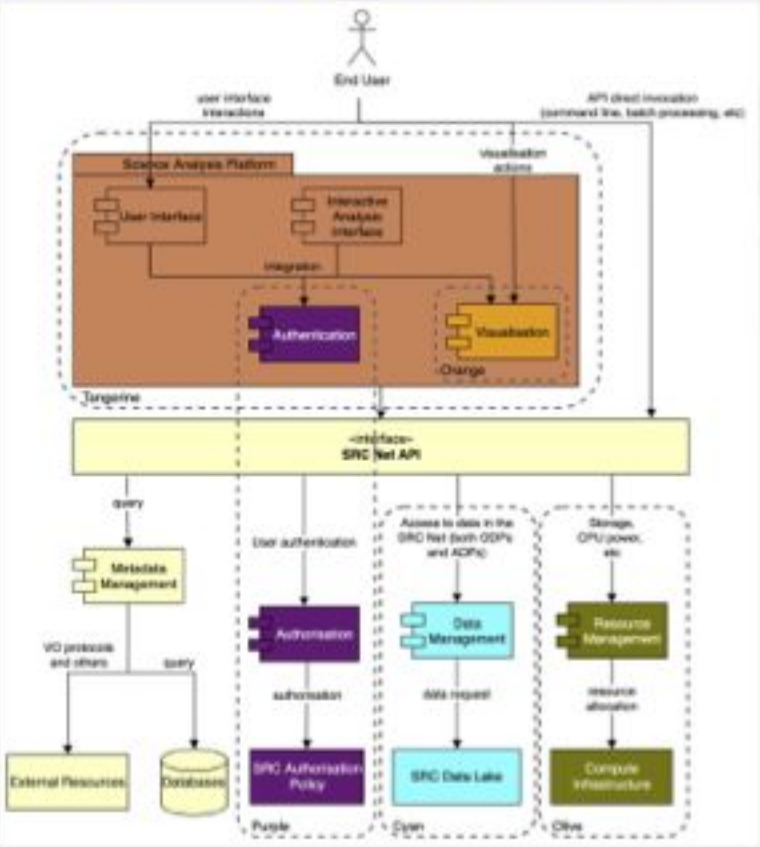
Data Management
Dissemination of Data to SRCs and Distributed Data Storage

SRC Network global capabilities

Collectively meet the needs of the global community of SKA users

Anticipate heterogeneous SRCs, with different strengths

Defining an architecture (WIP)



Users

Science Platform

Interface (API layer)

Metadata query - Science Data Discovery

Authentication Who? Permissions?

Data Logistics Globally distributed storage sites

Compute Resource Management Work sharing

Progress - SRC Rucio prototype

RSEs in SKA Rucio (SToRM webdav or dCache. S3 under investigation)

- *Not a production instance! Development and testing playground only!*
- Rucio is well suited to centralised Operations model for data management
- Performed long-haul transfers, Rucio stress tests, subscriptions (via our automated test framework)
- Integrating storage from national SRC efforts to increase understanding and inform assessment
- Continuing to assess suitability to SRC use cases. We've moved to a fully token-based system!

shout to Xavi Espinal and the ESCAPE project for giving us the skills and confidence to bring our own system to life!

Rucio metadata interaction

- Work done on metadata filtering, now possible to search metadata using a range of operators
  - e.g. ranges, logical OR, compound inequalities
- Existing metadata plugin system extended to support external postgres (RDBMS) and mongodb (NoSQL) backend technologies
- Using this, some **promising exploratory work** done on exposing IVOA TAP services with data taken from an external postgres instance

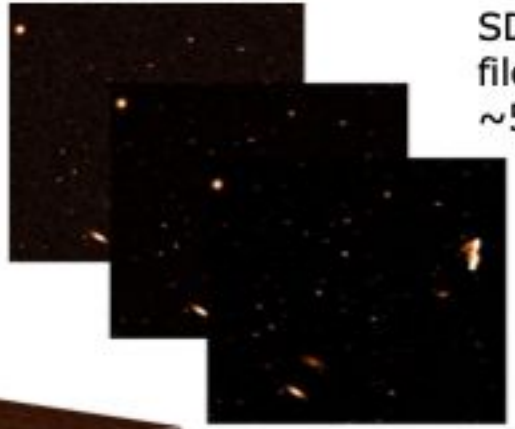
shout to Rob Barnsley leading Rucio metadata SIG, and Dave Morris: see Rucio Community Workshop later...

SKA Science Data Challenges

Science Data Challenges are a great way to involve science community with simulated SKA data products.

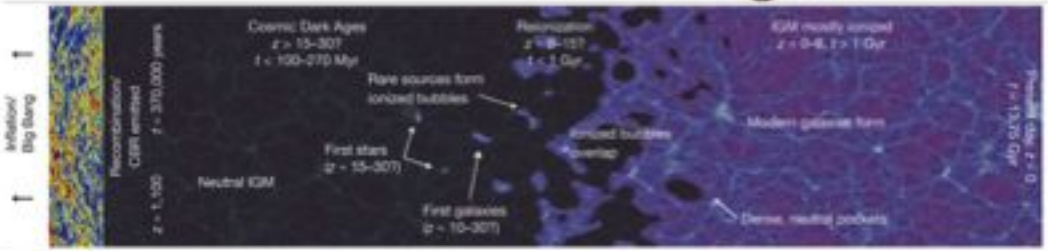
Successive challenges planned increasing in complexity and accuracy

Moving to a model where SKA community working groups define the challenges



SDC1 - each file ~4GB, ~50GB total

SDC2 - 1TByte simulated image cube



SDC3a - ~6TByte visibility data set

Summary

Rucio is an open, reliable, and efficient data management system

- Supporting the world's largest scientific experiments, but also a good match for smaller communities
- Extended continuously for the growing needs and requirements of the sciences

Strong cooperation between physics and multiple other fields

- Diverse communities have joined, incl. astronomy, atmospheric, environmental, …
- Community-driven innovations to enlarge functionality and address common needs

Benefit from advances in both scientific computing and industry

- Lower the barriers-to-entry by keeping control of data in scientist hands
- Seamless integrations with scientific infrastructures and commercial entities
- Detailed monitoring capabilities and easy deployment have proven crucial

Additional information

Website	http://rucio.cern.ch
Documentation	https://rucio.cern.ch/documentation
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

CERN