Rucio

Distributed Data Management in ATLAS

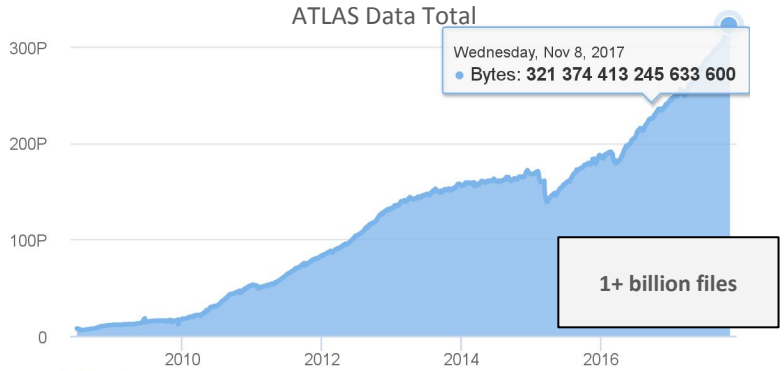
Martin Barisits
martin.barisits@cern.ch
on behalf of the Rucio team

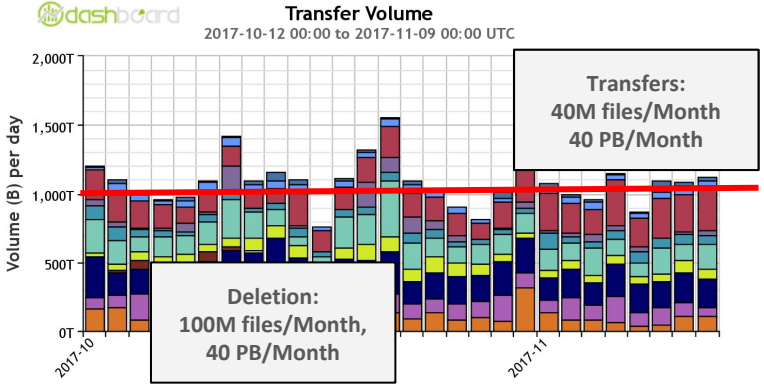
Why Rucio ?

- At time of inception, no global/commercial solution for distributed data handling available
- ATLAS developed its own tools for “Big Data” handling and computing
- Rucio was build using more than 10 years of experience in Data Management
  - Designed from experience from the previous data management system DQ2
  - Integrate new features and technologies
  - Modular, highly scalable, well supported

Rucio in a nutshell

- Charged with managing all data products
  - Detector, Simulation, Derivation, User
- Main functionalities
  - Discovery, Location, Transfer, Deletion
  - Quota, Permissions, Consistency
  - Monitoring, Analytics
  - Enforces the computing model
- Easy integration with workload management
- Automate everything to reduce operations
- Enables heterogeneous data management
  - No vendor/product lock-in
  - Able to follow the market





Upload:
150M files/Month
50PB/Month

Rucio development and commissioning

- ~5-6 FTEs development, operations, commissioning
- Long initial process
  - Design phase ~1 year
  - Initial development ~2 years
  - Commissioning ~1 year
- Gradual migration from predecessor system DQ2
- Now: Bi-weekly patch releases
- Major releases only during technical stops

In a Nutshell, Rucio...

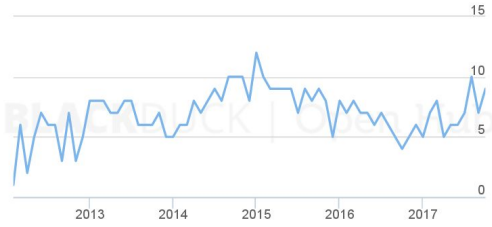
... has had 5,506 commits made by 40 contributors representing 101,729 lines of code

... is mostly written in Python with an average number of source code comments

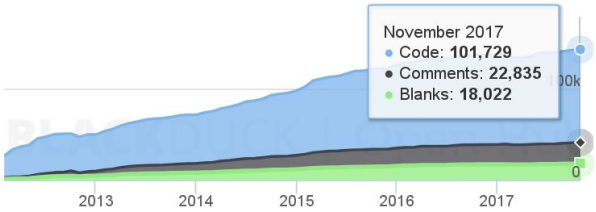
... has a well established, mature codebase maintained by a large development team with decreasing Y-O-Y commits

... took an estimated 25 years of effort (COCOMO model) starting with its first commit in February, 2012 ending with its most recent commit about 23 hours ago

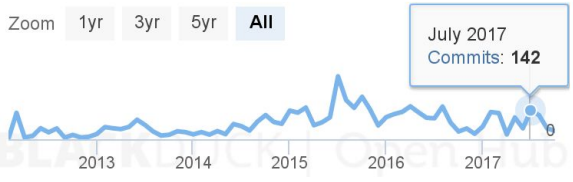
Contributors per Month



Lines of Code



Commits per Month

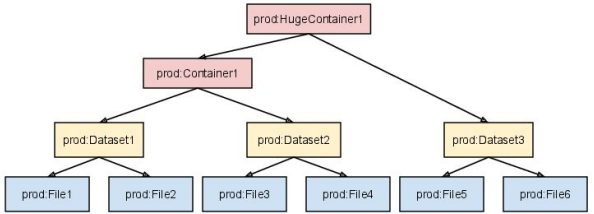


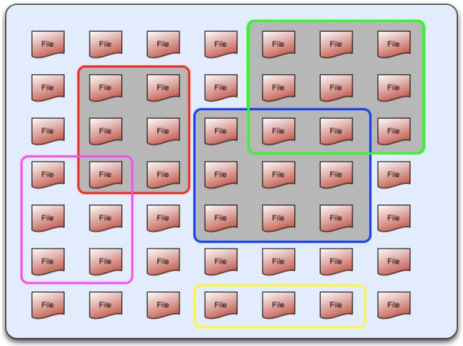
Data management operations model

- Most of the operational tasks are automated
  - Data distribution, deletion, staging
  - Data rebalancing, replication based on popularity, management of dataset lifetime
  - Identification of lost and dark files (Data consistency)
- People at the sites are not operating any local Rucio service
  - Sites only operate their storage (and network)
  - Configure their sites in central information catalogue
- ATLAS DDM central team operates more than 340 PB with 2 FTEs + shifters
  - Identify problems and communicate with the sites
  - Provide feedback to Rucio developers
  - Provide user support
  - Evolve the replication policies
  - Configure and run Rucio services

Namespace handling

- Smallest addressable unit is the file
- Files can be grouped into datasets
- Datasets & containers can be grouped into containers
- Namespace is partitioned by scopes
  - To distinguish different users, physics groups, or activities
  - Accounts can be mapped to users/ groups/activities
- Multiple data ownership across accounts
  - Prevent deletion of data
  - Also enforces quota
- Large set of available metadata, e.g.,
  - Data management: size, checksum, creation time, access time, …
  - Physics: run identification, derivation, number of events, …

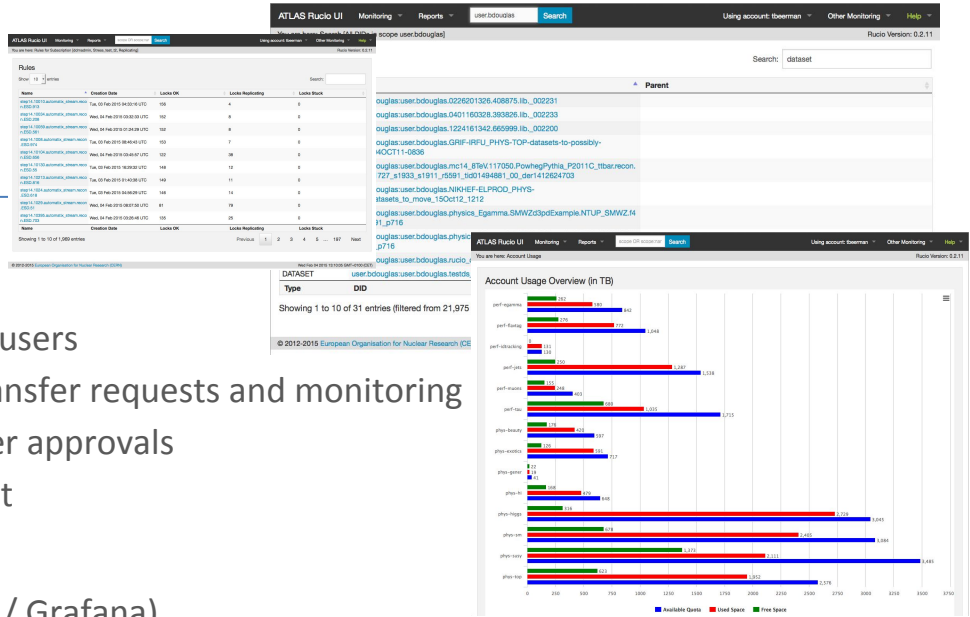




Declarative data management

- Express what you want, not how you want it
  - *e.g., "Three copies of this dataset, distributed evenly across two continents, with one copy on TAPE"*
  - 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
- Replication rules
  - Lock data against deletion in particular places for a given lifetime or pin
  - Primary replicas have indefinite lifetime rules
  - Secondary replicas are dynamically created replicas based on traced usage and popularity
- Subscriptions
  - Automatically generate rules for newly registered data matching a set of filters or metadata
  - e.g., *project=data17_13TeV* and *data_type=AOD* evenly across *T1s*

Monitoring and 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
  - Admin: Account / Identity / RSE management
- Monitoring
  - Internal system health monitoring (Graphite / Grafana)
  - Transfer / Staging / Deletion monitoring for ATLAS using CERN IT Unified Monitoring Architecture (ActiveMQ / Kafka / Spark / HDFS / ElasticSearch / InfluxDB / Grafana)
- Analytics
  - Periodic full database dumps to Hadoop (pilot traces, transfer events, …)
  - Used studies, e.g., transfer time estimation which is now already in a pre-production stage

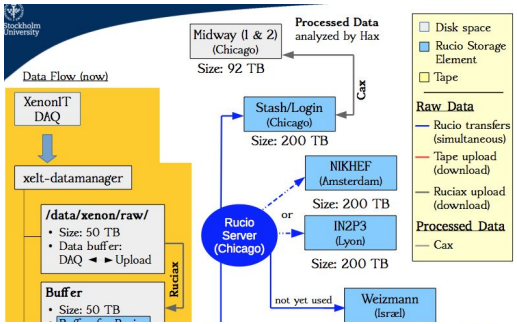
Rucio beyond ATLAS

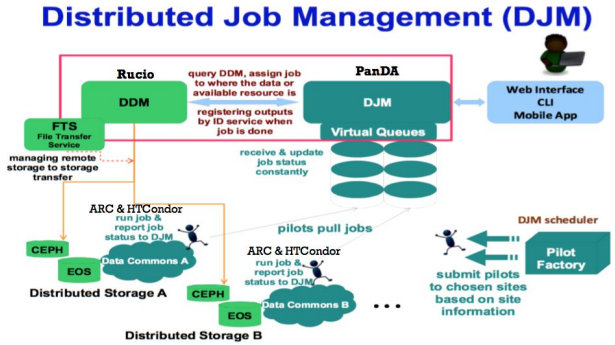
- The AMS and Xenon1T experiments are using Rucio:

  Xenon1T Dark Matter Search
  - Thousands of files across 6 sites (Europe and US), using the MariaDB backend, operated by UChicago

  AMS (Alpha Magnetic Spectrometer)
  - Millions of files across 10 sites, using the MySQL backend, operated by ASGC Taiwan
- CMS using the PostgreSQL backend operated by UChicago to evaluate Rucio
- + COMPASS, LSST and some others

→ Rucio Community Workshop: March 1-2, 2018



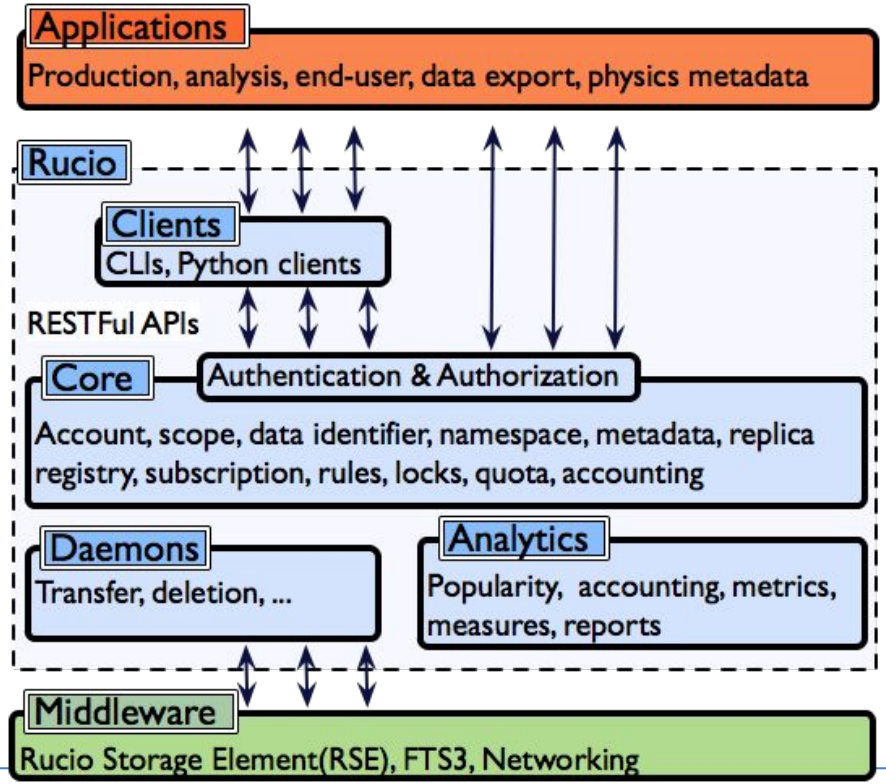


Thank you!

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/
Developer contact	rucio-dev@cern.ch

Backup

Software stack overview



Strong use of open and standard technologies!

Server:

- RESTful APIs /WSGI/HTTP Caching
- Token-based authentication (oauth)

Daemons:

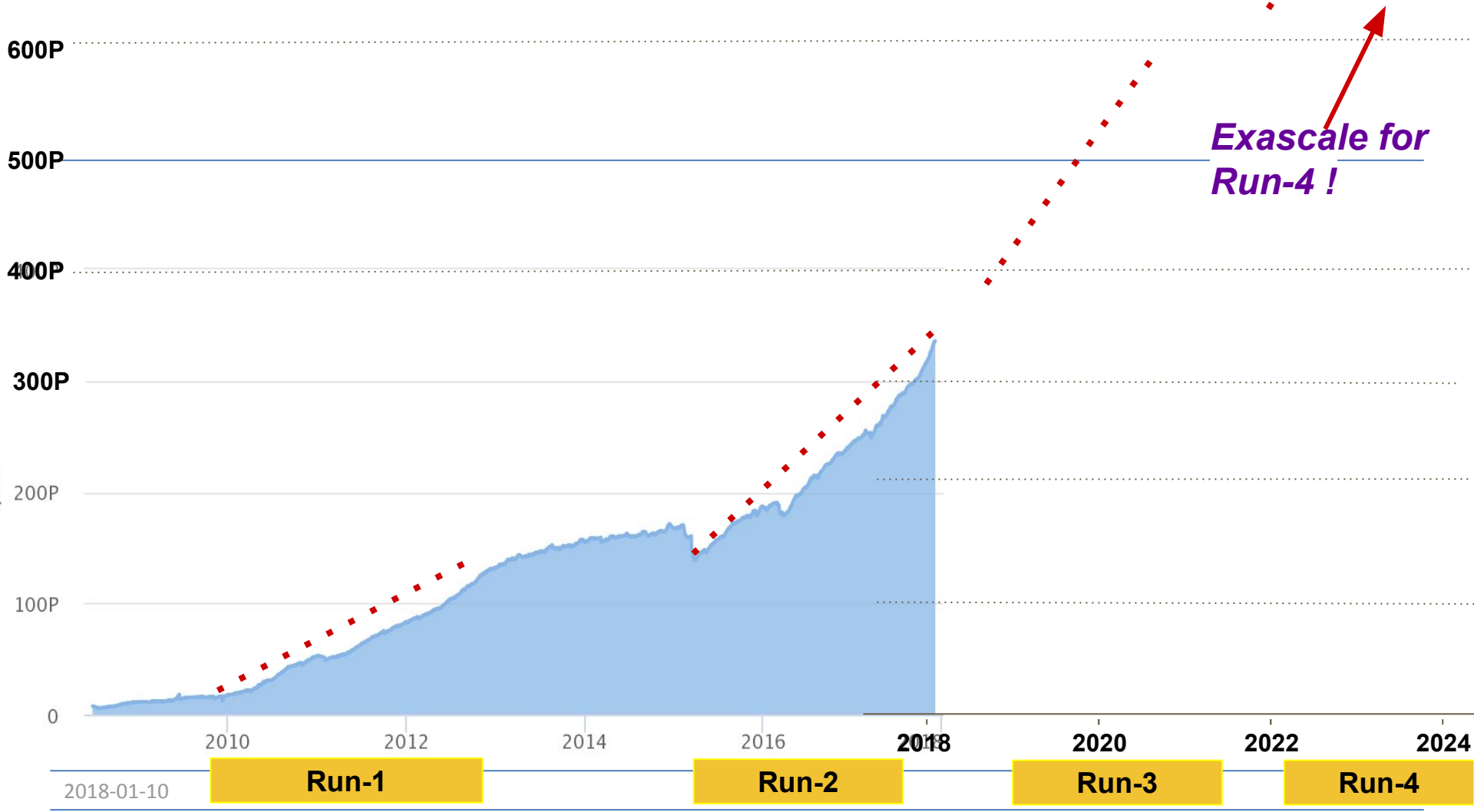
- Orchestrating remotely the collaborative work of all the system, e.g., transfers
- Lightweight, thread-safe and scale horizontally

Persistence (DB):

- Object Relational Mapper (ORM)
- Oracle, PostgreSQL, MySQL, MariaDB

Topology & Transfers

- Storage systems are abstracted as *Rucio Storage Elements (RSEs)*
  - Logical definition, not a software stack
  - Mapping between activities, hostnames, protocols, ports, paths, sites, …
  - Define priorities between protocols and numerical distances between sites
  - Can be tagged with metadata for grouping
  - Files on RSEs are stored deterministically via hash function, but can be overridden (e.g., for TAPE)
- Rucio provides a generic transfertool API
  - `submit_transfers()`, `query_transfer_status()`, `cancel_transfers()`, …
  - Independent of underlying transfer service
  - Asynchronous interface to any potential third-party tool
  - Currently only available implementation of transfertool API is FTS3
    - Additional notification channel via ActiveMQ
    - Potential to include GlobusOnline for improved HPC data transfers


600P
500P
400P
300P
200P
100P
0
2010
2012
2014
2016
2018
2020
2022
2024
Exascale for
Run-4 !
2018-01-10
Run-1
Run-2
Run-3
Run-4

Network Evolution

- We'll be more and more reliant on our foundation of the network
- It's coherent with our approach to use it more and more
  - E.g., remote i/o
- By 2020, 800 Gbps waves should be possible but not from everywhere..

Network Use in ATLAS