Experiences with the new ATLAS Distributed Data Management System

V Garonne¹, M Barisits², T Beermann², M Lassnig², C Serfon¹ and W. Guan³
on behalf of the ATLAS Collaboration

¹ University of Oslo, Norway
² CERN, Geneva, Switzerland
³ Department of Physics, University of Wisconsin

E-mail: vincent.garonne@fys.uio.no

Abstract. The ATLAS Distributed Data Management (DDM) system has evolved drastically in the last two years with the Rucio software fully replacing the previous system before the start of LHC Run-2. The ATLAS DDM system manages now more than 250 petabytes spread on 130 storage sites and can handle file transfer rates of up to 30Hz. In this paper, we discuss our experience acquired in developing, commissioning, running and maintaining such a large system. First, we describe the general architecture of the system, our integration with external services like the WLCG File Transfer Service and the evolution of the system over its first years of production. Then, we show the performance of the system, describe the integration of new technologies such as object stores, and outline some new developments, which mainly focus on performance and automation.

1. Introduction
The ATLAS experiment[1] at the LHC is a general purpose particle physics detector designed to investigate physics at the energy frontier. The Distributed Data Management (DDM)[4] project manages ATLAS data on the grid (~130 storage sites). The current DDM system relies on the Rucio software project, developed during Long Shutdown 1 to address the challenges of Run-2. It provides a catalog for all ATLAS data on the grid and takes care of data transfers, deletions and many other data management operations. This version of the DDM system services has been put in production just before LHC Run-2. Therefore, we describe here our experience in running and extending such a complex system.

In this paper we give an overview of the ATLAS DDM system, covering core concepts and its architecture and technologies (section 2). In section 3, we will enumerate the new services and features which have been introduced during LHC Run-1. We will then discuss our experience in running and operating such a system by showing the performance results and comparing them to the ones obtained during LHC Run-1 (Section 4). We will then conclude and give our future plans (Section 5).
2. **Concepts and architecture**

In this section, we describe briefly the core concepts of Rucio and its architecture[2]. For Rucio, files are the smallest operational units of data. Files can be grouped into datasets and datasets can be grouped into containers. A Rucio Storage Element (RSE) is a repository for physical files. It is the smallest unit of storage space addressable within Rucio. Replica management is based on replication rules defined on data identifier sets (files, datasets, or containers). A replication rule defines the minimum number of replicas to be available on a list of RSEs. Rucio creates replica locks that satisfy the rule and can trigger a data transfer prior to the replica lock being satisfied. A replica lock is defined for a replica on a specific RSE and is owned by the issuer account. A Rucio account can represent individual users, a group of users or a service like the organised production activity for the whole ATLAS collaboration. In case a replica on a particular RSE has no associated replica locks anymore, it can be deleted. Locks disappear when accounts decrease the number of desired replicas in their replica rules, or the whole rule altogether.

For storage accounting, Rucio accounts will only be charged for the files on which they have set replication rules. Accounting and quota calculations use the replica locks generated from replication rules. Quotas are constraints on the replica locks, based on limits set per account. The permissions of the account to execute a given request are also defined by policy.

![Figure 1. Overview of the Rucio architecture.](image)

As illustrated in figure 1, the Rucio system is based on a distributed architecture and can be decomposed into four main components: Server, Daemons, Resources, and User Interface. The Rucio server is a passive component listening to incoming queries. It connects several Rucio core components together and offers a common REST interface for external interaction. The RSE abstraction layer is responsible for the interactions with different grid middleware tools and storage systems. It effectively hides the complexity of these tools and combines them into one interface used by Rucio. The Rucio daemons are active components that orchestrate the collaborative work of the whole system. One example daemon is the Conveyor which is in charge of file transfers. The persistence layer keeps all the logical data and the application states. Rucio uses SQLAlchemy[6] as an
object relational mapper and can therefore support several relational database management systems (RDBMS) like Oracle, MySQL or PostgreSQL.

3. New features and services
During LHC Run-2, the DDM system has been extended with new services automating some manual and cumbersome operational tasks. One example of such work is when a Tier-1 disk runs full and there is nothing to delete. DDM Operations then need to rebalance manually the data to other Tier-1 disks to allow more activities. The task to select data with the least impact on users and while respecting replication policies has been automated and is fully described in a dedicated paper[6]. The system has been also enhanced with new features or capabilities like the pre-placement of popular data[7]. In this section, we describe the other new components and features introduced into the system.

3.1. Rucio WebUI - The Web Interface
The Rucio WebUI is a web interface and a convenient tool for different kinds of users to interact with the system and monitor it. For example, this HTTPS based interface allows to request dataset transfers or monitor account usage. The most important tools are:

Rucio Rule Definition Droid (R2D2). R2D2 is a tool for users to submit transfer requests and for site admins a way to control transfers to their sites using quotas and an approval system. It has basically three different views, starting with the request interface for new transfers. Users can create replication rules in four simple steps(section 2), starting with the selection of data to be transferred. It can either be done by performing a wildcard search and then selecting the wanted datasets/containers or by directly providing a list. Next, the destination RSE can be selected, either using a specific RSE or an RSE expression. The system then checks the quota on those RSEs and informs the user whether an approval request is necessary. In the third step users can provide additional options like number of replicas and a mandatory comment in case of a needed approval. In the last step an overview is given before the users can submit the request to the system.
The second view is the listing of rules. This can either be used by users to show the status of their rules, and get more details for single rules, or by site admins to list rules currently waiting for approval. Two buttons, approve or deny, are provided for each single rule. The last view is the quota management page. Here site admins can either set general quotas for all existing and future users of an RSE or do more fine-grained per-user adjustment.

Rule Backlog & Subscription Monitoring. An overview for shifters and operations shows the state of rules for official activities like T0 export. The users can quickly spot problematic rules and get more details for those including links directly to the FTS monitoring. File Transfer Service (FTS)[12] is a data movement service used by Rucio.

Group Account Usage. An overview for physics group admins shows the overall usage for each physics group including detailed information per storage endpoint.

Bad replica summary. This shows an overview of lost or corrupted files for site admins with the possibility to get details per storage endpoint and check single files. The information about the bad or suspicious files is obtained directly from the workload management system or is declared by site admins.

The Rucio WebUI has been implemented with web.py which is a minimalistic web framework used to
serve the sites and provide authentication. The JQuery Javascript library has been used to communicate
with the Rucio Rest API and for site manipulation directly from the browser. The tool has been well
received since its introduction in the beginning of 2015 with more than 300 users in August 2016
alone. The tool is continuously improved with one of the latest changes being the introduction of a
fine-grained quota management.

3.2. Integration of lightweight sites: Rucio cache

Setting up a new grid site with a storage element is a large amount effort and involves a steep learning
curve for some sites with very limited storage and manpower. From operational experience, the small
sites (<400 TB) and very small sites (<100 TB) generate more problems and operational load. For
these reasons, storage-less sites are becoming more popular as a way to provide grid computing
resources with less operational overhead. Rucio with its cache feature is an ideal solution for
lightweight grid sites in the ATLAS experiment. Rucio supports the concept of caches which are
controlled outside Rucio and may not be consistent. A Rucio cache is an RSE, tagged as volatile, for
which Rucio doesn’t control all file movements. The application populating the cache must register
and unregister file replicas in Rucio.
The ARC CE Cache[8] has been integrated with the ATLAS data management system. ARC CE cache
can publish its content to Rucio through add/delete messages. The cache service can create dumps of
cache content, and a separate script runs periodically to calculate the differences and send messages to
Rucio. The cache RSEs are associated to the CE’s PanDA[3] queue and so PanDA can broker jobs to
queues where the data is cached.

3.3. Rucio Auditor - Consistency Enforcement in the ATLAS Distributed Data Management System

Ensuring the consistency between the storage and the DDM catalog is a long standing issue of grid
computing. Two types of inconsistencies exists:

- Files in the catalog but not physically on the storage. We call this category: ‘lost Files’.
- Files on the storage but not registered in the catalog, named as ‘dark data’.

We describe here the work done to automatically detect these inconsistencies and fix them. This
automatic consistency check is based on dumps comparison. Each site is supposed to provide storage
elements dumps on a regular basis (monthly or quarterly). Rucio dumps of the file replicas supposedly
located at the site are generated every day. We compare the Storage Element dump with the Rucio
dumps at -X days and +X days. The files in the two Rucio dumps but not in the SE dumps are lost
files. The files in the SE dump but in none of the Rucio dump are Dark Data. The process of
identifying the dark data and lost files is explained in figure 2.
Figure 2. Figure explaining how the inconsistencies (lost files and dark data) are identified between the Rucio catalog and a storage with the dumps.

A daemon called Auditor compares the dumps. It automatically checks the dumps uploaded on the storage for each ATLAS site, gets their creation date and downloads them. It gets the Rucio dumps for \( \pm X \) days stored on a Hadoop cluster. It runs the comparison between the 2 dumps. The comparison time scales from a few seconds for the small sites to a few hours for the biggest one (70M files). Another daemon collects the list of dark data and automatically deletes them. The files identified as lost are reported to the site support for investigation. Another daemon takes care of the files confirmed as lost. If another replica exists, the lost files are automatically recovered. If no other replica exists, the files are removed from the datasets they belong to and the owner of the dataset, as well as the ATLAS Metadata Information system, (AMI[12]) are notified.

The consistency service has been running in production for more than 12 months. It allowed to identify and clean \(~4\) petabytes of dark data (out of \(~250\) petabytes of data registered).

3.4. Integration of object-based storage

Within the ATLAS DDM system, the integration of a new storage type is a constant need. The vast majority of cloud storage available in the market leverages an object-based storage architecture. It differs in many points from traditional file system storage and offers a highly scalable, simple and most common storage solution for the cloud. Object-based storage supports other authentication mechanisms than X509, usually based on (secret) access keys. The main idea was to integrate object-based storage as regular DDM end-point RSE and to focus on the Amazon S3 (Simple Storage Service) protocol. Rucio allows the integration of non-posix namespace and has a flexible mapping protocol-endpoint. Two protocols have been implemented: \( s3:// \) and \( s3+rucio:// \). The protocol \( s3:// \) has been implemented with the \textit{boto} python library and requires access keys. It is the preferred way on central machines we have access to in a secure way, e.g., central deletion. The protocol \( s3+rucio:// \) has been implemented with pure http python library and relies on pre-signed URLs, which guarantee (temporary read/write) access to the object.

Currently, several institutes host object-based storage for ATLAS: BNL, Lancaster, RAL, CERN, MWT2. These storages have a Ceph implementation[9]. Two use cases are supported in production: i) upload and download of log files and ii) deletion of objects generated by the ATLAS Event Service.
(AES)[9] on object-based storage.

4. Experience, results and performance

The ATLAS DDM system has demonstrated large-scale data management capabilities. At the end of year 2016, it managed more than 250 petabytes distributed across more than 130 storage sites. Because of the exceptional performance of LHC in 2016, we collected 50% more data than expected. This increase of data is shown in figure 3 during the run-2 period. The system has been able to cope with this sudden increase (from 120 petabytes to 250 petabytes) and has maintained good performance. Rucio manages now more than 1 billion file replicas spread on 130 sites.

![Figure 3. Evolution of the total ATLAS space on the grid.](image1)

During Run-1, the previous system called DQ2 was reaching its limits in terms of scalability. These limits were observed mainly on the database part. DQ2 was hard to extend with new technologies and use cases. Last year performance has clearly shown the advantages of the new DDM system. In terms of transfers, 40M files representing roughly 40 petabytes were transferred each month. This is comparable to Run-1, but the files are bigger. We observed more peaks generating more load and hotspot on the network. Figure 4 illustrates the increase of the network usage from ATLAS during Run-2. From May 2015 to May 2016, the volume of transferred data increased by a factor of 3, raising from 12 petabytes to 38 petabytes.

![Figure 4. Transfer Volume of ATLAS during Run-2 (Petabytes per Month).](image2)
The total storage capacity did not increase as fast as the rate of data taking. This situation induces a very high rate of data deletion: 100M files/month, representing more than 40 petabytes. The deletion rate increased by a factor of 4 since Run-1 and indicates the pressure on disk spaces is much higher. The rate of deletion per month is shown in figure 5.

![Deletion Done Volume](image1)

**Figure 5.** Volume of deletion (per month and cloud).

The replicas are divided into different categories. The disk-resident replicas, called primary data, are required to fulfill the ATLAS computing model or to serve ongoing activities like reprocessing or production. The cache data, or secondary data, can be deleted in a Least Recently Used (LRU) manner if space is needed. This category can be pre-placed to serve more unscheduled activities like user analysis as described in [7]. The idea is to keep additional copies of files to reduce the response time and bandwidth usage.

![Number of Physical Bytes](image2)

**Figure 6.** Evolution of the ATLAS data volume in bytes per replica category (from the bottom up: cache data in green, tape in red, disk resident in orange) during Run-2.
Figure 6 illustrates the evolution of the three categories of data during part of Run-2. Although the cache data category stayed constant, the disk resident increased by 70 petabytes, which indicates the pressure put on disk space and network. The usage of tape has also seen a significant increase.

5. Conclusion
In this paper, we have shown that Rucio, with its concepts and technologies, scales well according to the needs of ATLAS during Run-2. Therefore Rucio can be considered feature-complete, high-performance and reliable. It has been operating robustly, stably and effectively since the beginning of 2016. The necessary and increasing usage of the network and tape systems will bring some interesting challenges in the coming years. One example is data access over the world area network with failover. We need to gain one order of magnitude in computing capability to take ATLAS forward into the next years of high luminosity LHC running.

References
[1] The ATLAS Collaboration 2008, Journal of Instrumentation 3 S08003
[2] Barisits M et al 2014 Replica management in Rucio: Replication rules and subscriptions Journal of Physics: Conference Series 513 042003
[3] Wenaus T et al 2012 PanDA: Next generation workload management and analysis system for big data SC Companion (IEEE Computer Society) pp 1521–1522
[4] Garonne V et al 2013 Rucio - The next generation of large scale distributed system for ATLAS Data Management, Journal of Physics: Conference Series 513 4
[5] Copeland R 2008 Essential sqlalchemy (O’Reilly)
[6] Barisits M et al 2016 Automatic rebalancing of data in ATLAS distributed data management CHEP 2016 San Francisco
[7] Beermann T et al 2016 C3PO - A Dynamic Data Placement Agent for ATLAS CHEP 2016 San Francisco
[8] Ellert M et al 2007 Advanced Resource Connector middleware for lightweight computational Grids Future Generation Computer Systems 23 pp 219–240
[9] Tsulaia V et al 2016 Production Experience with the ATLAS Event Service CHEP 2016 San Francisco
[10] Weil Sage A et al 2006 Ceph: A Scalable, High-performance Distributed File System Proceedings of the 7th Symposium on Operating Systems Design and Implementation Seattle Washington 307-320
[11] Molnár Zsolt 2012 Next generation WLCG File Transfer Service (FTS) Journal of Physics: Conference Series 396 3
[12] Albrand S et al 2008 The ATLAS Metadata Interface J.Phys.Conf.Ser. 119 072003