The ATLAS Data Management System Rucio: Supporting LHC Run-2 and beyond

IOP Conf. Series: Journal of Physics: Conf. Series **1085** (2018) 032030 doi:10.1088/1742-6596/1085/3/032030

The ATLAS Data Management System Rucio: Supporting LHC Run-2 and beyond

M Barisits[1], T Beermann[2], V Garonne[3], T Javurek[4], M Lassnig[2] and C Serfon[3] on behalf of the ATLAS collaboration

[1] University of Innsbruck, Austria
[2] CERN, Geneva, Switzerland
[3] University of Oslo, Norway
[4] University of Freiburg, Germany

E-mail: martin.barisits@cern.ch

Abstract. With this contribution we present some recent developments made to Rucio, the data management system of the High-Energy Physics Experiment ATLAS. Already managing 300 Petabytes of both official and user data, Rucio has seen incremental improvements throughout LHC Run-2, and is currently laying the groundwork for HEP computing in the HL-LHC era. The focus of this contribution are (a) the automations that have been put in place such as data rebalancing or dynamic replication of user data, as well as their supporting infrastructures such as real-time networking metrics or transfer time predictions; (b) the flexible approach towards inclusion of heterogeneous storage systems, including object stores, while unifying the potential access paths using generally available tools and protocols; (c) machine learning approaches to help with transfer throughput estimation; and (d) the adoption of Rucio for two other experiments, AMS and Xenon1t. We conclude by presenting operational numbers and figures to quantify these improvements, and extrapolate the necessary changes and developments for future LHC runs.

1. Introduction

Rucio [1] is the Distributed Data Management (DDM) system in charge of managing all ATLAS [2] data on the grid. The main purpose of the system is to help the collaboration to store, manage and process LHC data in a heterogeneous distributed environment. Typical tasks are: Transfer data to/from sites, delete data from sites, enforce the experiment's computing model. The system manages 300 Petabytes of physics data across more than 130 data centers globally, with more than 830 million files. Over the last year multiple improvements have been introduced to support the collaboration's data management needs for LHC Run-2 and beyond. These changes focused on the integration of new technologies, the automation of the system for optimization and reduction of manual work, and machine learning studies to better understand the usage of the system.

The paper is organized as follows: In Section 2 we present the integration of object store technologies into Rucio. In Section 3 we discuss two new approaches in system automation: Dynamic creation of additional replicas and automatic rebalancing of data. Section 4 continues with machine learning approaches for data management. In Section 5 we give a short overview of Rucio beyond ATLAS. Finally we conclude the article in Section 6.

2. Object stores

Within the ATLAS DDM system, the integration of a new storage type [3] 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 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 boto python library and requires access keys. It is the prefered 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. 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) on object-based storage.

3. System automations

A significant part of the day to day operation is to optimize the usage of space. Rucio is currently using two daemons that automate the handling of replicas on the grid, either by automatically creating new replicas of datasets that are considered popular and therefore could benefit from extra replicas on the grid or by moving replicas around to balance the used and free space on storage.

3.1. Dynamic creation of additional replicas

This daemon [4] continuously monitors incoming user analysis jobs and the decides based on a multitude of parameters if and where to make extra copies of the input datasets. The system works as a cache for popular data. These extra replicas are created with a short lifetime, which guarantees that they can be deleted again quickly if not used anymore and space is needed. Because of the way the deletion works in Rucio if the replica continues to be used it will also stay on storage as long as enough space is available.

Every time a new job is defined a decision algorithm runs that uses the following information to make a decision:

- The past popularity of the input dataset. Rucio uses a tracer system that receives a trace every time a file is either used in grid jobs or downloaded. This information is then aggregated to get number of accesses for this dataset in the last 7 days.
- The existing replicas. If already more than 4 replicas exist on the grid the daemon will not create another one.
- The free space on potential destination sites.
- Network speed between potential sites hosting an existing replica and potential destination site.

The first two metrics are used to filter potential datasets that could benefit from extra replicas the latter two are combined to rank potential source and destination storage, where the ones with the most free space and a good connection are preferred. If a storage element has been picked to get a new replica it will get a penalty, so that for the next dataset it will be less likely to be picked again. It can prevent a single site with a lot of free space to be flooded with a lot of transfers. Figure 1 shows the kibana dashboard of this daemon.

Figure 1. Kibana overview dashboard to monitor the decision made by the dynamic creation of additional replicas daemon

3.2. Rebalancing of data

The need of automatic rebalancing of data [5] arises due to the fact that some storage systems are at their capacity limit, but do not have any secondary data to delete. These are the storage resources which need data rebalanced to other storage systems in order to be functioning again. There are three reasons why these situations arise:

- Imbalances between computation resources and storage resources at a site. Thus, due to the higher computing capacity at a site more data is transferred and generated at the site than the storage element can hold. Mostly Tier-1 sites are affected by this phenomenon.
- Non optimal placement of data. This can either happen due to policies or users requesting replication. The data distribution policies are rather static, thus they do not make the most optimal decision at every point in time.
- In some cases computation jobs also request too large transfers or the job output is too large for a site to handle. This brings a storage element also into a situation of crisis.

The general idea of data rebalancing in ATLAS distributed data management is to move data away from the problematic storage endpoint while still adhering to the original data replication policy as much as possible. Data integrity is a key requirement in this process, thus, the original data is not deleted until the newly created data has been transferred and validated. Currently three different modes are supported for data rebalancing: Manual (emergency) rebalancing, automatic (emergency) rebalancing and automatic background rebalancing. The *manual rebalancing of data* mechanism gives DDM operators a tool to easily rebalance a given amount of data away from a storage element. This can be either done in emergency situations, such as a storage element being full without anything to delete, or also due to other data distribution needs. A rebalancing operation can be issued via the command line client interface by just specifying the storage from which to rebalance data, and the amount of bytes to rebalance. The *automatic emergency rebelancing* mode relieves DDM operators by autonomously detecting storage systems in crisis and rebalancing data away from them to make room for new data. Figure 2 shows a problematic data utilization workflow. The primary data occupancy (shown

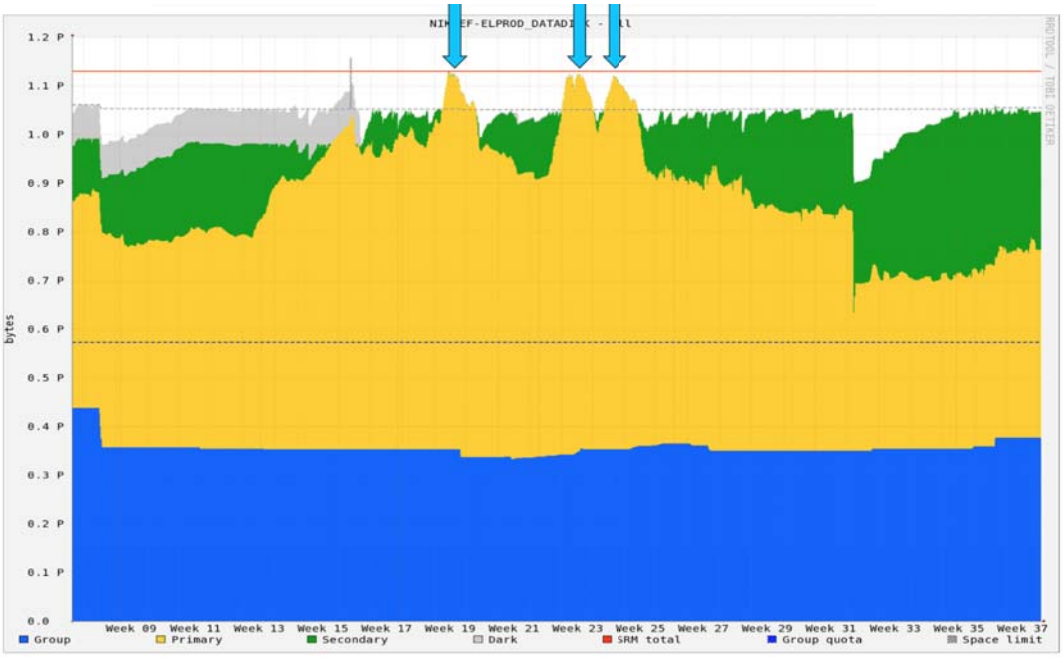

Figure 2. Problematic data utilization of a storage system.

as yellow and blue in the plot) rises, eventually replacing all secondary (shown as green) data and reaching the volume limit of the storage element. The concept of the automatic emergency rebalancing mode is to look for storages which are over the watermark and which do not have any secondary data to delete. In these cases, the mechanisms rebalance a fixed amount of data away from the storage. The idea of the *automatic background rebalancing* is to prevent emergency situations like this from even occurring. To this effect the automatic background rebalancing daemon trys to balance the amount of primary to secondary replicas at a set of storages, such as all Tier-1 storage elements. The idea behind this is, if the ratio is in balance, all storages have relatively the same amount of secondary replicas available for deletion.

4. Machine learning for data management

To quantify any improvements in data placement we need an estimator of potential transfer times. A global minimum solution for all expected transfers yields, in theory, an optimum placement strategy when assuming equivalence of computational resources. Geographical distance or network bandwidth are insufficient quantifiers though. The dynamic nature of both the involved applications and the networks themselves, as well as their physical deployment, require a more sophisticated method. Currently there is no sensible model available to estimate the duration of a particular collection of file transfers between data centres.

Long Short-Term Memory (LSTM) networks seem particularly suited to annotated time series data with recurring but non-periodic bursts [6]. We selected Keras as the software framework for the implementation since it supports LSTMs and opens the possibility for deep learning at later steps. We created the LSTM with one hidden layer at 512 neurons and one dense layer with one output, activated by a rectified linear unit (ReLU). The training input variates are *source*, *destination*, *activity*, *bytes*, *start timestamp*, and *end timestamp* of one transfer, using *end timestamp* as the label. The result of the model is shown in Figure 3. The absolute prediction error is less than 5 minutes across the majority of activities and links. There exists some multimodality, however it is constrained within an acceptable error range [7].

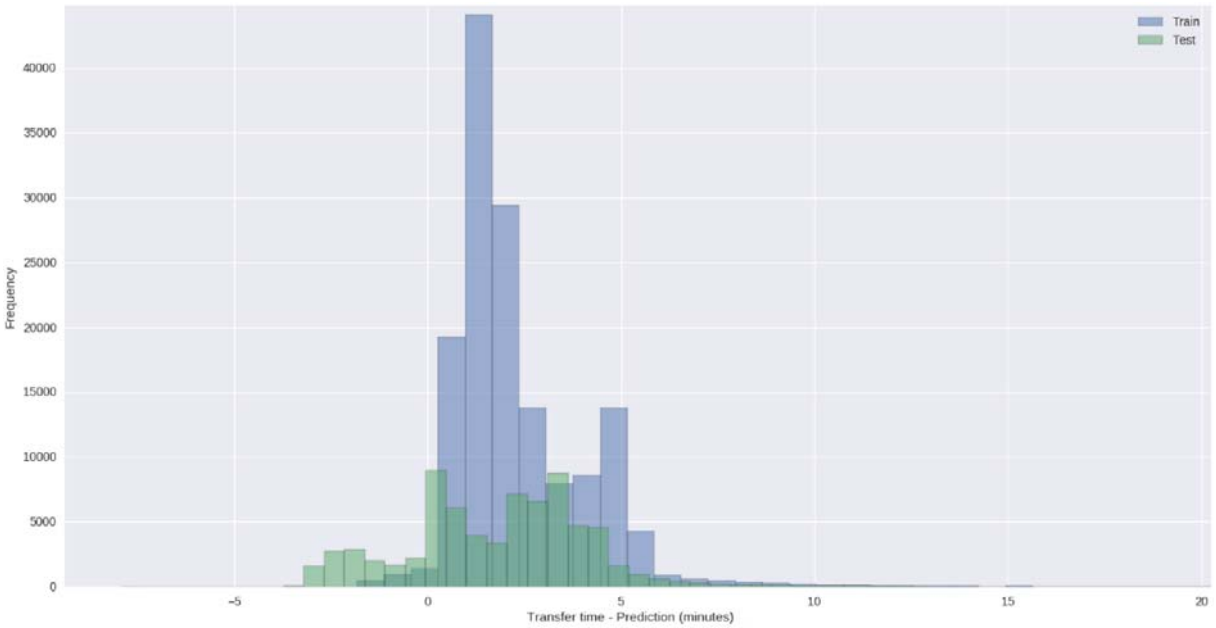

Figure 3. Single layer LSTM model absolute error across all activities and labels.

5. Beyond ATLAS

In addition to ATLAS, two other experiments are already running Rucio in production. The Alpha Magnetic Spectrometer (AMS) and Xenon1t experiments have expressed interested in both the namespace and transfer capabilities of Rucio, and subsequently installed a separate Rucio instance each. The most striking difference to ATLAS is the usage of a different database backend, namely MariaDB instead of Oracle, demonstrating the versatility of the Rucio persistence layer. Shared installations of the underlying File Transfer Service (FTS) instances at CERN and BNL are used by all experiments. Current developments in Rucio target an even easier deployment scheme for additional experiments using docker containers, new development workflows using *github.com* to facilitate non-ATLAS contributions, as well as monitoring improvements for experiments who cannot benefit from CERN-provided monitoring.

6. Conclusion

Rucio has demonstrated excellent performance since the beginning of LHC Run 2. In this paper we presented some of the recent developments that were done in order to reduce the operational burden as well as to exploit new storage technologies. Research activities based on machine learning will help us to further improve the performance and scalability of the system. Developed initially for the ATLAS experiment, Rucio is now used by two other non-LHC experiment and is being evaluated by others.

References

[1] Garonne V, Vigne R, Stewart G, Barisits M, Beermann T, Lassnig M, Serfon C, Goossens L, Nairz A, *Rucio - The next generation of large scale distributed system for ATLAS Data Management*. Journal of Physics: Conference Series 2014
[2] ATLAS Collaboration, *ATLAS Collaboration*. JINST **3** (2008) S08003
[3] Garonne V, Guan W, *Object-based storage integration within the ATLAS DDM system*. to appear in Journal of Physics Conference Series 2017
[4] Beermann T, Lassnig M, Barisits M, Garonne V, Serfon C, *C3PO - A Dynamic Data Placement Agent for ATLAS Distributed Data Management*. to appear in Journal of Physics Conference Series 2017

[5] Barisits M, Serfon C, Garonne V, Lassnig M, Beermann T, Javurek T, *Automatic rebalancing of data in ATLAS distributed data management*. to appear in Journal of Physics Conference Series 2017
[6] Hochreiter and Schmidhuber, J 1997 Neural Computation 9(8):1735-1780
[7] Lassnig M, Toler W, Vamosi R, Bogado J, *Predictive analytics for ATLAS Data Management using Machine Learning methods*. to appear in Journal of Physics Conference Series 2017