Extending Rucio with modern cloud storage support

Martin Barisits[1], *Robert* Barnsley[2], *Fernando Harald* Barreiro Megino[3], *Johannes* Elmsheuser[4], *Mario* Lassnig*[1], *Mihai* Patrascoiu[1], *James* Perry[5], *Cedric* Serfon[4], *Alba* Vendrell Moya[1], and *Tobias* Wegner[1]

[1]European Organization for Nuclear Research (CERN), Geneva, Switzerland
[2]Square Kilometre Array Observatory (SKAO), Cheshire, UK
[3]University of Texas at Austin (UTA), Austin TX, USA
[4]Brookhaven National Laboratory (BNL), Upton NY, USA
[5]University of Edinburgh, Edinburgh, UK

Abstract. Rucio is a software framework designed to facilitate scientific collaborations in efficiently organising, managing, and accessing extensive volumes of data through customizable policies. The framework enables data distribution across globally distributed locations and heterogeneous data centres, integrating various storage and network technologies into a unified federated entity. Rucio offers advanced features like distributed data recovery and adaptive replication, and it exhibits high scalability, modularity, and extensibility.
Originally developed to meet the requirements of the high-energy physics experiment ATLAS, Rucio has been continuously expanded to support LHC experiments and diverse scientific communities. Recent R&D projects within these communities have evaluated the integration of both private and commercially-provided cloud storage systems, leading to the development of additional functionalities for seamless integration within Rucio. Furthermore, the underlying systems, FTS and GFAL/Davix, have been extended to cater to specific use cases.
This contribution focuses on the technical aspects of this work, particularly the challenges encountered in building a generic interface for self-hosted cloud storage, such as MinIO or CEPH S3 Gateway, and established providers like Google Cloud Storage and Amazon Simple Storage Service. Additionally, the integration of decentralised clouds like SEAL is explored. Key aspects, including authentication and authorisation, direct and remote access, throughput and cost estimation, are highlighted, along with shared experiences in daily operations.

1 Introduction

In recent years, substantial efforts have been devoted to the integration of the Rucio data management system [1] with different cloud storage solutions. This integration has paved the way for new possibilities and considerations in the realm of scientific cloud computing.

When discussing clouds in this context, it is essential to keep two angles in mind: The *technical angle* encompasses various aspects related to cloud storage, including access

*Contact: Mario.Lassnig@cern.ch

tools, transfer protocols, monitoring mechanisms, authentication and authorisation processes (AAI), accounting and billing procedures, and the underlying storage infrastructure. The *organisational angle* involves considerations such as whether the storage solution is deployed on-site or off-site, whether it follows a centralised or distributed architecture, whether it is built on open-source or closed-source software, and whether it is utilised as a public service within an institute or laboratory, or acquired as a commercial service. Additionally, the method of contribution, either in-kind or as a paid service, plays a role in shaping the overall cloud storage environment.

Navigating the intricacies of cloud storage can be challenging, and various scenarios exemplify the diversity and complexity involved. For example, various plausible scenarios exist already today: a self-hosted MinIO S3 server[2] deployed on a CERN data centre virtual machine (VM) that utilises a centrally managed CephFS volume[3]; a WebDAV portal facilitating access to a self-hosted NextCloud instance[4] hosted by a commercial provider, which, in turn, is connected to free-tier AWS S3 storage[5]; or an experiment collaborates with a commercial cloud provider receiving complimentary storage with S3v4 protocol support[6]. The distinction of *cloud storage* versus *grid storage* thus quickly becomes difficult.

From the perspective of Rucio, cloud storage is then simply defined as storage that necessitates URL-based signatures for access and management, given a predefined signature algorithm and the association secret key sharing mechanism. For exampling, when placing CephFS on top of RADOS (Reliable Autonomic Distributed Object Store)[7], it requires some form of storage system on top, akin to a grid-style storage model. In contrast, adopting the Ceph Object Gateway S3 API [8] on top of RADOS represents a cloud storage configuration, as it is characterised by its compatibility with the S3 protocol and associated necessity to use URL-based signatures.

This ongoing integration between Rucio and cloud storage technologies is expected to have profound implications for the management and accessibility of data in diverse scientific and computational domains. By facilitating seamless interactions with cloud storage environments, Rucio opens up new avenues for data storage, sharing, and analysis, enhancing the capabilities of research institutions and scientific communities.

2 Rucio credential mechanism

When dealing with namespace operations, such as listing replicas of data stored in Rucio, and storage operations, like uploading and downloading data from storage, the process involves generating URL signatures at the time of executing the command. Each URL signature is one-time use only and allows a particular operation to interact with a particular file. These URL signatures are produced server-side by Rucio, eliminating the need to deploy secrets to the Rucio clients.

To enable this functionality, certain requirements in the Rucio deployment must be met: The associated account must possess schema permission `perm_get_signed_url` and a specific account attribute `sign_url`. The Rucio Storage Element (RSE), which represents the actual storage resource definition, must have several configurations applied: The storage access scheme must be set to `https` to ensure secure communication using this well-known protocol; The protocol implementation utilised should be `rucio.rse.protocols.gfal.NoRename`, since cloud storage typically does not allow file renames which breaks the typical safe atomic upload mechanism from Rucio; Specific attributes must be defined for the RSEs, including `sign_url` with allowed values of `s3`, `gcs`, or `swift`, indicating compatibility with corresponding interfaces; as well as additional mandatory attributes like `verify_checksum=False` since cloud storage typically does not do checksum verification, the `s3_url_style=path` to indicate support for multiple cloud storage endpoints on single nodes, and `skip_upload_stat=True` as well as `strict_copy=True` to skip grid-style storage checks.

For the credential secrets configuration, there are two mechanisms: For S3 and SWIFT-compatible interfaces, e.g., MinIO, Amazon, Ceph S3 Gateway, an entry in the `rse-account.cfg` Rucio configuration file is required to establish the necessary credentials for the server-side generation of the URL signatures. For Google Cloud Storage[9] compatibility, the system necessitates the use of the Google-native JSON credential file obtained from the Google Cloud Console[10]. This file contains the essential credentials required by the Google SDK to calculate the URL signatures.

By adhering to these configurations and ensuring the presence of the appropriate credentials, Rucio can facilitate secure and efficient data operations with various storage interfaces, and is easily extensible for upcoming cloud providers.

3 FTS credential mechanism

When adding Rucio replication rules for Third-Party-Copy (TPC) of files between data centres, the generation of URL signatures must be delegated to the File Transfer Service (FTS)[11]. The duration for which transfer jobs will remain in the FTS queue is uncertain, making it imperative to implement time-limited URL signatures for enhanced security and only generate the signature at the time when it is actually needed.

Furthermore, there is no standardised or universal method for third-party-copying between different cloud storage providers. The TPC process may vary depending on the specific combination of cloud storage services involved, but always requires an active party in the copy process which cannot be done by passive cloud storage implementations.

Regarding credential configuration in FTS, there are three components to configure:

1)*Secrets Configuration:* The credentials required for TPC need to be inserted into the FTS configuration. This typically involves accessing the `fts-host:8446/config/cloud_storage` endpoint and inserting the necessary credentials in a specific format.

2)*GFAL Configuration:* The GFAL (Grid File Access Library)[12] configuration, accessible via `fts-host:8449/fts3/ftsmon/config/gfal2`, cannot be directly edited by users or administrators. Instead, it must be configured and set by the FTS administrators, ensuring secure and proper handling of the GFAL settings.

3)*HTTP Configuration:* Similarly, the HTTP plugin configuration, accessible via `fts-host:8449/fts3/ftsmon/config/http_plugin.so`, cannot be directly modified. The FTS administrators are responsible for configuring this plugin to ensure seamless HTTP-based interactions within the FTS environment.

By adhering to these credential and configuration guidelines, the FTS system can facilitate secure and efficient third-party data transfers while maintaining the necessary security measures and adhering to the specific requirements of cloud storage providers involved in the TPC process. The involvement of FTS and the delegation of URL signature generation enhance the robustness and security of the overall data transfer workflow.

4 Commercial clouds: Google

Google Cloud Storage has been a subject of long-term research and development (R&D) within the ATLAS[13] project[14], aimed at evaluating the feasibility of using a cloud environment as a grid site. This endeavour involved overcoming various challenges, such as integrating X.509 certificates, the predominant authentication mechanism in grid computing, into commercial cloud services.

To facilitate this integration, local administrators at the grid sites were instrumental in supporting the deployment during the development of the proper solution. In particular, a CERN-provided host certificate had to be injected into a dedicated next-generation Google Load Balancer[15], enabling secure communication and access to resources for the clients.

As part of the effort to accommodate ATLAS' Tier-1 storage requirements, custom proxy rules were devised, supporting the typical `DATADISK` and `SCRATCHDISK` areas. However, this proxy did not perform as expected in the Google Load Balancer, as it was not properly replying with the necessary host certificate. This was necessitating a return to the legacy Google Load Balancer with an even more complex and complicated setup.

Despite the initial challenges, the project has achieved stability since then, with jobs successfully running on Google Compute Engine[16]. This results in that ATLAS computing tasks are effectively being executed within the Google Cloud environment, with both data and compute in the cloud, as well as a similar approach with data on the grid and compute in the cloud, giving flexibility to the experiment.

To optimise space occupancy within the cloud storage system, a greedy deletion model has been adopted. This approach ensures that resources are released efficiently, as it prioritises the removal of data that is no longer needed or has reached its expiration, freeing up space for new data. This is important for cost control, as unused data at rest still incurs charges. This is a crucial difference when compared to typical grid sites.

In conclusion, the ATLAS R&D project has made significant progress in evaluating Google Cloud Storage as a grid site. By collaborating with administrators and addressing technical challenges, the project has succeeded in establishing a stable computing environment on Google Compute Engine. Additionally, the adoption of a greedy deletion model further enhances resource utilisation within the cloud storage infrastructure. This ongoing research and development hold promise for future endeavours of cloud integration, which are currently already in the planning stages.

5 Commercial clouds: SEAL

SEAL Storage Technology[17] is a distributed cloud storage solution that leverages the Interplanetary File System (IPFS)[18] and Filecoin (FIL)[19] to provide reliable and scalable storage capabilities. As part of a long-term R&D project, SEAL has generously offered 10PB of storage to the ATLAS Experiment. To ensure data integrity and long-term archival, SEAL employs a cryptographic *sealing* process that securely packages and stores data, safeguarding it for extended archival on the Filecoin network. Thus Rucio itself does not need to have any integration or interaction with the Filecoin network.

In terms of integration with Rucio, the collaboration has been exceptionally seamless. Rucio's standard URL signature mechanism has been effectively integrated with SEAL's cloud storage, facilitating secure and authenticated access to stored data using the widely-adopted S3 cloud access protocol. Similar to the approach taken with Google Cloud Storage, SEAL administrators injected a CERN-provided host certificate into their load balancer to enable secure interactions between Rucio and the SEAL storage infrastructure.

Recognising the significance of cloud support in Rucio and the success of the integration with SEAL Storage Technology, SEAL is actively investing in further advancements. They are funding a full-time development position dedicated to enhancing cloud support within the Rucio ecosystem via the University of Michigan. This commitment underscores the importance of continuous improvement and the potential for future innovations in data management and commercial cloud partners. The collaboration between Rucio and SEAL Storage

Technology demonstrates promising developments, as this work was specifically designed to provide a robust and generic cloud integration without any vendor lock-in.

6 Commercial clouds: Amazon

In the context of our ongoing discussion about various cloud storage technologies and their integration with the ATLAS Experiment, the experience with Amazon Web Services (AWS) presented unique challenges.

Initially, the integration with AWS seemed straightforward, and it functioned smoothly for a significant period, thanks to the accidental use of DigiCert[20] host certificates by AWS. However, the situation took a complicated turn when Amazon transitioned to using its custom Certificate Authority (CA) for managing security certificates.

Within the ATLAS Experiment, the FRESNO US Tier-3 had invested significantly in an R&D project on the AWS infrastructure. Nevertheless, setting up the integration with the new custom CA proved to be a daunting task, marked by numerous trials and errors.

The process of achieving a successful integration with the updated AWS environment spanned over six months of dedicated effort. As a result of this persistent endeavour, a concise and valuable document was produced, summarising the key insights, findings, and solutions discovered during this challenging period.

This experience highlights the complexities involved in integrating cloud storage services, particularly when changes in security mechanisms and certificate management are introduced by cloud providers like Amazon. It also underscores the importance of thorough testing, troubleshooting, and documentation to ensure the successful integration and continued functionality of cloud storage solutions for Rucio. Such valuable lessons serve as a basis for enhancing future cloud storage integration efforts.

7 ROOT IO

In scenarios involving interactive analysis and other stream processing cases, remote reads play a crucial role in accessing data stored in cloud storage systems. When using cloud storage, the path returned from the `rucio list-replicas` operation can often be directly fed into the widely used `TFile::Open()` C++ function from the ROOT IO toolkit[21] in physics analysis software, simplifying the process of opening and accessing remote files.

However, it's important to be aware that the S3 protocol, used by Amazon Web Services (AWS) and many others, does not support multi-range byte requests which are necessary for efficient `TFile` operations. To overcome this limitation, AWS requires the use of their CloudFront Content Delivery Network (CDN), which provides a translation layer capable of handling multi-range requests.

On the other hand, cloud storage providers like Google Cloud Storage or MinIO do not have such a built-in translation layer for multi-range requests. To work around this absence, a simple solution is to emulate multi-range requests through the Davix library. This can be achieved by appending specific URL options to the `TFile::Open()` function call, emulating the behaviour of actual multi-range requests, by serialising parallel accesses and thus sacrificing IO throughput. The options appended include `multirange=false&nconnections=XX`, where the number of connections should be suitable scaled to the client usage. This workaround may vary depending on the client used for data access, as different clients may have different configurations and requirements. Therefore, a one-size-fits-all approach is not applicable in this context.

To address this issue more systematically, it was worth considering modifications to Rucio's behaviour. One potential approach is to have Rucio include the necessary URL options when replying with the list of replicas. This could involve providing a hint to the list-replicas operation, such as `"-nr-connections-for-direct-io=30"` or a similar solution, allowing Rucio to specify the appropriate configuration at runtime to use during remote reads. This was preferable to some communities using Rucio. For ATLAS, a server-side approach was chosen, to limit the possibilities of clients causing disturbing scenarios on the storage and network, and to reduce the potential cost implications of repeated failed reads.

By investigating and implementing such specific cloud enhancements, Rucio can facilitate a more seamless and efficient data access experience in interactive analysis and stream processing scenarios, regardless of the cloud storage provider being used.

8 Future work

The current configuration and setup of cloud storage integration in the project have evolved organically through ongoing Cloud R&D projects and requires refactoring. While the existing setup has served as a foundation, it is apparent that a complete overhaul is needed, particularly with regard to the naming of attributes and the overall complexity of the implementation.

In preparation for a production-level integration, several essential features have been identified as necessary improvements. Firstly, the access control mechanism requires refinement to allow for more fine-grained control, as the current setup only supports an all-or-nothing approach. Secondly, a smarter peering mechanism is needed, considering the trade-offs between static multi-hop distance configurations and dynamic cloud regions. Introducing the concept of cloud regions is deemed crucial for enhanced flexibility and resource optimisation, also including different regional cost.

Security considerations have also come to the forefront. The current dependency on X.509 certificates with the DNS-injection workaround is insufficient for comprehensive security. To improve security practices, leveraging cloud providers' support for OpenID/OAuth2 flows is deemed advantageous and can be supported by general token migration efforts in grid computing. Additionally, implementing mechanisms for throughput and cost control is necessary to prevent unrestricted access to cloud resources. As of now, the design phase for these has only just started.

The integration process is further expected to benefit from bucket-copy transfer tools, which would facilitate direct transfers between cloud-based storage buckets without the need to involve the File Transfer Service (FTS). A potential cloud boosting option was also proposed, allowing dynamic allocation of currency by involved institutes or science groups for extra throughput or storage as required, providing greater flexibility in resource utilisation and accelerated scientific results.

Consideration of data lifetime and the associated cloud Quality of Service (QoS) costs are crucial factors for efficient cloud storage management. Studying cloud storage caching through theoretical R&D simulations, as demonstrated in Tobias Wegner's PhD work[22], offers valuable insights into optimising data access and retrieval processes. As Wegner has shown, temporary cloud bursting presents a promising approach to improve typical science workflows that require tape recalls. By achieving more than 15% reduction in job times, this approach showcases the potential of cloud bursting to enhance overall performance in data-intensive operations.

In conclusion, while the existing cloud storage integration has laid the groundwork, a comprehensive revision and enhancement of the configuration and features are required for a seamless and efficient production-level integration. Addressing issues related to access control, security, throughput, cost control, and data lifetime considerations will contribute to optimising cloud storage utilisation and resource management within Rucio. Theoretical R&D studies and the exploration of innovative approaches like temporary cloud bursting provide valuable insights for further advancing the project's scientific data management capabilities in the cloud.

References

[1] M. Barisits *et al.*, *Rucio - Scientific data management*, Comput. Softw. Big Sci. **3** (2019) no.1, 11

[2] MinIO: High Performance Object Storage, https://min.io/

[3] CephFS: Ceph File System, https://docs.ceph.com/en/latest/cephfs/

[4] NextCloud: Online collaboration platform, https://nextcloud.com/

[5] Amazon Web Services, https://aws.amazon.com/

[6] S3 Protocol Version 4, https://docs.aws.amazon.com/AmazonS3/latest/API/

[7] Reliable Autonomic Distributed Object Store (RADOS), https://docs.ceph.com/en/reef/rados/api/librados-intro/

[8] Ceph Object Gateway S3 API, https://docs.ceph.com/en/latest/radosgw/s3/

[9] Google Cloud Storage, https://cloud.google.com/storage

[10] Google Cloud Console, https://console.cloud.google.com/

[11] E. Karavakis *et al.*, *FTS improvements for LHC Run-3 and beyond*, EPJ Web Conf. **245** 04016 (2020)

[12] Grid File Access Library, https://gitlab.cern.ch/dmc/gfal2

[13] ATLAS Collaboration, *The ATLAS Experiment at the CERN Large Hadron Collider,* JINST **3** S08003 (2008)

[14] F. Barreiro *et al.*, *Accelerating science: the usage of commercial clouds in ATLAS distributed computing* Proc. CHEP Conf. (2023) - in these proceedings

[15] Google Cloud Load Balancer, https://cloud.google.com/load-balancing

[16] Google Compute Engine, https://cloud.google.com/compute

[17] SEAL Storage Technology, Decentralized Cloud Storage, https://www.sealstorage.io/

[18] D. Trautwein *et al.*, *Design and evaluation of IPFS: a storage layer for the decentralized web*, SIGCOMM '22: Proceedings of the ACM SIGCOMM 2022 ConferenceAugust 2022

[19] Protocol Labs, *Filecoin: A decentralized storage network*, White paper, 2017

[20] DigiCert Certification Authority, https://www.digicert.com/

[21] I. Antcheva *et al.*, *ROOT: A C++ framework for petabyte data storage, statistical analysis and visualization*, Comput.Phys.Commun. **182** (2011) 1384-1385

[22] T. Wegner *et al.*, *Simulation and Evaluation of Cloud Storage Caching for Data Intensive Science*, Comput.Softw.Big Sci. **6** (2022) 1, 5