EPJ Web of Conferences **214**, 04020 (2019) https://doi.org/10.1051/epjconf/201921404020
CHEP 2018

The Data Ocean Project

An ATLAS and Google R&D collaboration

Martin Barisits[1], *Fernando* Barreiro[2], *Thomas* Beermann[3], *Karan* Bhatia[4], *Kaushik* De[2], *Arnaud* Dubreuil[5], *Johannes* Elmsheuser[6], *Alexei* Klimentov[6], *Mario* Lassnig[1], *Peter* Love[7], *Tadashi* Maeno[6], *Andrea* Manzi[1], *Ruslan* Mashinistov[6], *Andy* Murphy[4], *Paul* Nilsson[6], *Sergey* Panitkin[6], and *Tobias* Wegner[1], on behalf of the ATLAS Collaboration*

[1]European Organisation for Nuclear Research (CERN)
[2]University of Texas at Arlington (UTA)
[3]University of Innsbruck
[4]Google LLC
[5]University of Geneva
[6]Brookhaven National Laboratory (BNL)
[7]University of Lancaster

Abstract. Transparent use of commercial cloud resources for scientific experiments is a hard problem. In this article, we describe the first steps of the *Data Ocean* R&D collaboration between the high-energy physics experiment ATLAS together with Google Cloud Platform, to allow seamless use of Google Compute Engine and Google Cloud Storage for physics analysis. We start by describing the three preliminary use cases that were identified at the beginning of the project. The following sections then detail the work done in the data management system Rucio and the workflow management systems PanDA and Harvester to interface Google Cloud Platform with the ATLAS distributed computing environment, and show the results of the integration tests. Afterwards, we describe the setup and results from a full ATLAS user analysis that was executed natively on Google Cloud Platform, and give estimates on projected costs. We close with a summary and and outlook on future work.

1 Project overview

ATLAS [1] is facing several challenges with respect to its computing requirements for LHC Run-3 in 2021-2023 and HL-LHC runs in 2026-2037 [2]. Many of these challenges are not specific to ATLAS or LHC though, but are common to the scientific computing community in general. Most importantly, storage continues to be the driving cost factor for computing and the projected capacity and throughput growth rates cannot accommodate the volume of data that is expected from future experiments within the foreseen funding constraints. Novel computing models with a more dynamic use of storage and computing resources need to be considered, however the current infrastructure is in production use and cannot be radically changed. Additionally, previous computing system simulation efforts have demonstrated that

due to the complex interactions of the involved software stacks and only fuzzy instrumentation data it is infeasible to arrive at conclusive models about data management [3]. Effective improvements to data management were due to repeated tests on the production infrastructure under different configurations which is time consuming and error-prone. For meaningful tests at the future scale of ATLAS though, additional resources have to be acquired to run such tests. The *Data Ocean* project has been started as an R&D project for evaluating and adopting novel IT technologies for scientific computing in cooperation with the commercial cloud provider Google.

ATLAS and Google have started to integrate storage and computing resources from the Google Cloud Platform (GCP) [4] into the ATLAS distributed computing environment. After a series of teleconferences, a face-to-face brainstorming meeting in Denver, CO at the ACM Supercomputing 2017 conference [5] resulted in the design of a first prototype of the *Data Ocean* project. The first proposed ideas were threefold: (a) to allow ATLAS user analysis to benefit from Google Cloud Platform, (b) to allow ATLAS to explore the use of different computing models to prepare for High-Luminosity LHC, and (c) to provide Google with science use cases to improve their cloud platform offerings.

Specifically, the following three use cases were identified out of which the actual work plan was structured:

1.1 User analysis

When analysts use the distributed analysis services to run on the grid, the outputs are deposited directly to the grid storage. Making 100 percent of those outputs available to the analyst quickly is a difficult problem and remains one of the weak points of distributed analysis. Throughout this R&D project, analysis outputs generated in worker nodes around the world would be directed to Google Cloud Storage (GCS), where they would become uniformly and reliably available to the analyst anywhere in the world. Analysis data products are small and the GCS-resident outputs could be regarded as a cache with a limited lifetime, and thus limited storage footprint, while the value of reliable accessibility of this hot data to analysts is enormous.

1.2 Data placement, replication, and popularity

The final stages of data analysis by users requires access to Petabytes of data products. To ensure high reliability of access, ATLAS replicates multiple copies of this data to worldwide computing resources. The Google Cloud Storage service could be an alternative to these highly used data products since it provides high throughput, high availability, and low latency. We plan to store the end result of a full ATLAS Monte-Carlo simulation and its associated reprocessing data products on Google Cloud Storage. These data products will then be available to users worldwide through Google Compute and ATLAS computing resources in parallel.

1.3 Data streaming

In order to increase flexibility, ATLAS is investigating the use of smaller units of data, called sub-file data products, in the analysis chain. A prototype of this *Event Streaming Service* [6] is currently in development and could benefit from fine-grained cloud storage. This use case evaluates the necessary compute to generate the sub-file data products, commonly referred to as events, from their original files at the scale required by HL-LHC, and the performance gains of highly parallel small size data delivery to the analysis software.

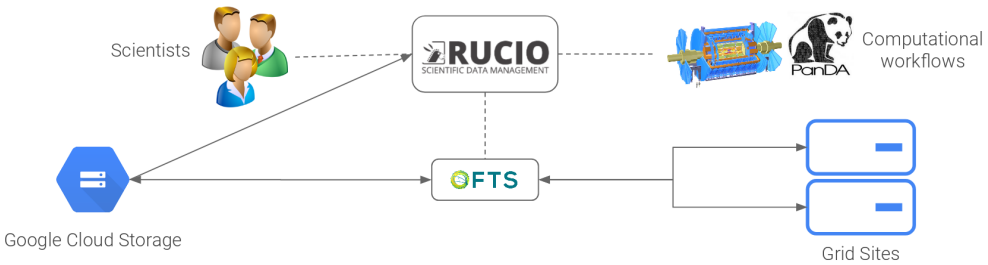

Figure 1. High-level diagram of the integration of ATLAS Distributed Computing and the Google Cloud Platform. Scientists and computational workflows get transparent access to cloud resources.

2 Data management

As the foundation, it was necessary to integrate GCS directly into Rucio [7], the scientific data management system of the experiment. This would allow users as well as automated computational workflows to transparently read and write data to the cloud using direct APIs as well as user-friendly clients. Figure 1 shows the high-level architecture between the main components: the Rucio data management system, the PanDA/Harvester workflow management system [8, 9], the file transfer service FTS [10], as well as Google Cloud Storage and grid storage. Users can directly interact with it using command-line, programmatic APIs, or web interfaces, whereas PanDA/Harvester use the API exclusively. The complex interaction between users and workflows with grid sites has been withheld for clarity.

First, the existing cloud storage access protocol S3 [11] was used to interface Rucio with GCS. S3 is an HTTP verb based protocol and can work in two authentication modes, either with access credentials that have to be distributed among all the clients, or centrally via access signatures. GCS provides S3 as a protocol to ease compatibility with competitor offerings, and was thus the natural first choice to test. The first throughput tests using S3 with access credentials exhibited a hard cap of 100MB/sec to selected European and North America storage regions available on GCS. Next to the problem of distributing the access credentials to users, and risking credential leaks, the S3 protocol has a limit of 5 Gigabyte for file uploads. We could not circumvent this limit without significant development effort therefore we decided to dismiss the use of S3 and instead use the GCS native protocol using HTTP with client-side URL signatures [12].

Several Rucio enhancements were implemented with respect to credential handling, replica lookup, and storage attributes to support transparent signatures. This effort is beneficial for all storage providers which support client-based signatures. Whenever a client requests the location of a file, and a location is found on a storage system that has the appropriate signature flag enabled, the access permissions are checked, i.e., if the user is allowed to expend credits to read or write that data. If yes, a single use time-limited signature is created by Rucio which includes the operation, the location, and other kinds of metadata. It is thus not possible to use the signature for any other operation.

To support storage-to-storage copy, also known as third-party-copy, several changes had to be made. First, we provided the signed URLs directly from Rucio to FTS. FTS would then open a channel to the source storage, open a channel to the signed URL on GCS and stream the results. Figure 2 shows the result of placing 2 Terabytes of data on GCS in parallel, one Terabyte to a European region, and one Terabyte to a North American region. The achieved throughput was roughly 0.6 Terabyte/hour, i.e., 166 MB/sec, with a short tail due to source storage read errors that had to be retried.

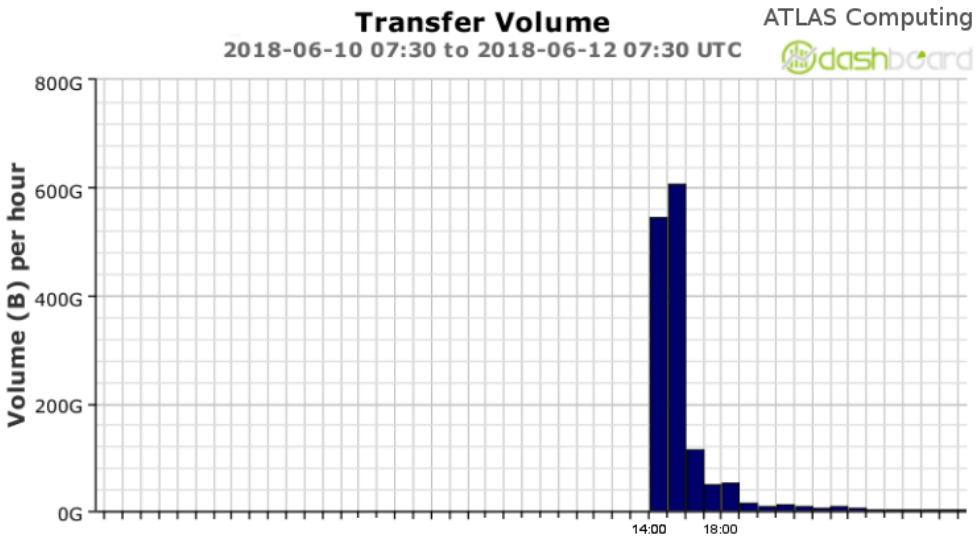

Figure 2. Terabyte-scale transfer test from WLCG to Google Cloud Storage to both European and American GCS endpoints concurrently. Bandwidth was limited by intermediate link capacity.

However, there are still several problems with this approach. In particular, this exposes several security risks, as the signature could be intercepted. FTS was therefore extended to create the client-based signatures internally and only use it for third-party-copy when instructed so by Rucio. Also, the available bandwidth on the source was not exhausted, but was limited by the streaming capacity of the FTS servers. To solve this problem it is necessary that FTS forces the source storage to act as the *active* part in storage-to-storage transfers when the *passive* destination part is cloud storage. Both extensions have been developed in FTS and will be deployed in late 2018, and therefore were not part of this test.

3 Workflow management

ATLAS is relying on the PanDA system to provide workload management on all computing resources. Generic factories submit the PanDA pilot to the batch systems. The pilot [13] occupies the slot, asks the PanDA server for a job, monitors the job through its lifetime, and reports back relevant metrics to the server. This approach was satisfactory on the grid, but as the heterogeneity of resources grew, such as HPC and clouds, the number of specialised factories and different solutions grew over the years. The Harvester component has been designed to address these issues and, to date, has interfaced through a plug-in architecture to cloud resources such as Google Compute Engine (GCE) and OpenStack, and also several US DOE HPCs machines such as Theta, Cori, and Titan [14–16].

As shown in Figure 3, PanDA can manage the Virtual Machine (VM) life cycle in Google Compute Engine (GCE) through the Harvester resource manager. This is a native PanDA GCE integration with no translation layers, where the plugins interface with GCE via its Python API. The actual messaging between the VMs and Harvester is done via HTTP. The instances use unaltered CernVM4 images [17] and industry-standard *cloud-config* contextualisation. This includes setup and configuration of CVMFS, PanDA resource queues, the

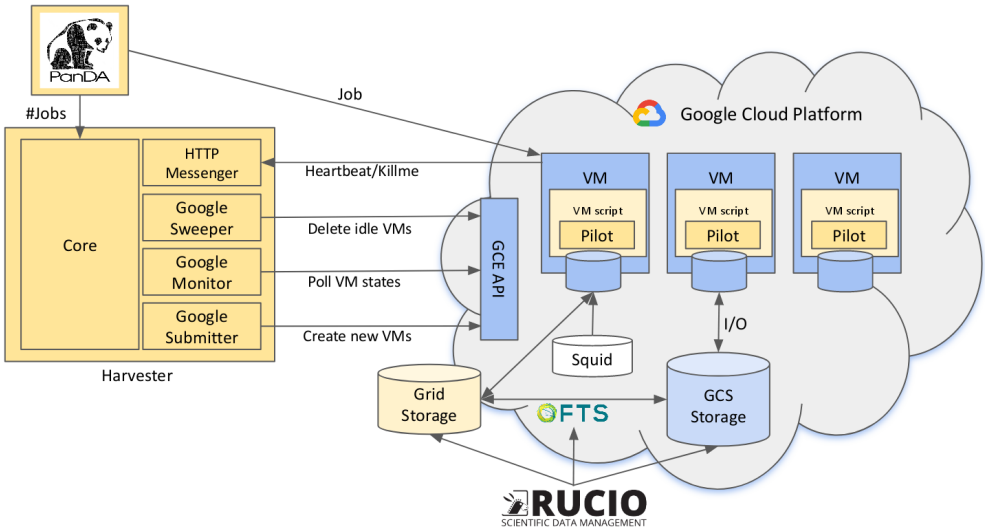

Figure 3. Integration of workflow management system using Harvester with Google Cloud Platform. The existing systems, such as Rucio or the PanDA Pilot, were used in the same way as in the WLCG infrastructure.

startup script, and many more. The startup script is 200 lines of Python code to instantiate the regular PanDA Pilot and continuously transmits and receives *heartbeat* and *killme* signals from Harvester. The VMs are recycled once per day based on the *timefloor* option in the PanDA Pilot. Additionally, Squid proxies [18] are deployed in GCE for caching.

4 Analysis

The simplified outline of the distributed analysis workflow on Google Compute Engine is as follows: create an Rucio-managed storage area on GCS, replicate analysis inputdata to GCS from grid storage using Rucio, submit PanDA jobs to process the data, measure the job performance throughout its runtime, and retrieve the job outputs with the *rucio download* utility. We used regular ATLAS Monte Carlo simulation software, version 20.7.8.4, over extremely skimmed physics data organised in 30 datasets, with a total of 450 GB of data for 1 million events.

The main complication in data management was to have the full-chain from Rucio datasets to the actual analysis software, using the ROOT data I/O framework, understanding the file-level signed URLs on cloud storage. From the computational point of view, several changes had to be applied to the PanDA Pilot and the execution environment to understand the signed URLs. Support in ROOT I/O was developed previous to this project for another integration project.

The first observation was that direct-IO with GCS using the signed URLs caused reading corruptions within ROOT, reproducible across a variety of different software versions. These errors did not happen on grid storage, so must stem from an incompatibility of GCS and the way ROOT expects HTTP streams. Eventually, this was changed to full file stage-in to the job working directory. This worked as expected, however defeated the purpose of only reading what is necessary and not getting the whole file, which incurs significant extra costs, as the network egress could potentially scale up by several orders of magnitude.

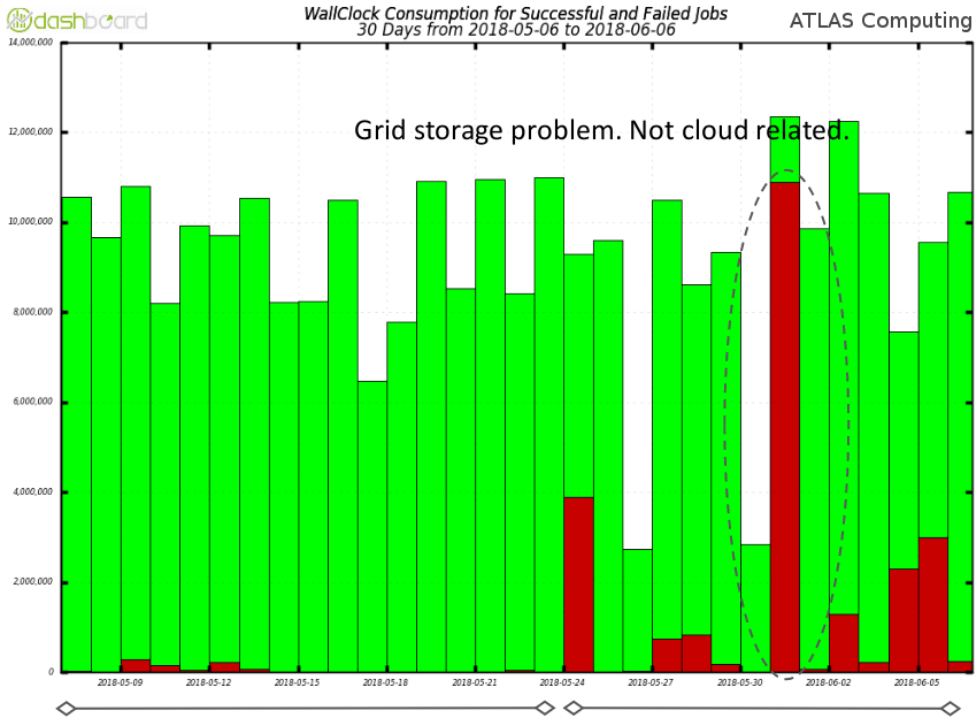

Figure 4. One month of running ATLAS jobs on GCP. First half running with dedicated VMs, second half running with preemptible VMs exhibiting slightly higher error rate.

The number of virtual machines on GCE is not bound by technical reasons and could be scaled up significantly. As shown in Figure 4, we evaluated both normal and preemptible VMs, going from close to 0% failure rate to almost 20% respectively. Considering that the cost of preemptible VMs is 80% lower, the cost/event ratio makes this an attractive option, however some parts of the ATLAS analysis software are not yet fully prepared for such cycle stealing and exhibit errors which potentially lose the whole job. Event level workflows would allow a reduction in the lost wallclock time by only losing the small event range currently being processed.

5 Costing

There was no API available to calculate the price for a finished job. The manual cost estimate thus includes the following items based on the standard GCP price template [19]:

- CPU: n1-standard-1, 1 CPU, 3.75GB RAM, $0.0475/hour.
- Storage: $0.026 GB/month multi-regional storage.
- Network: Ingress free, Egress $0.12 per GB/month.
- Access: 1 million storage operations total $10/month.

This would sum to $26,000 for storing 1 Petabyte/month of raw data without usage. ATLAS analysis usage would have to be added on top of that, where processing 50 TB of stored data with 5 TB job output in a month costs $143 per day, i.e., $4000 in total per month.

This includes storage, network, and 100 VMs with 24 cores each, providing 73k hours of CPU hours. On average, 100 cores out of the 2400 total cores would be idling though, due to the I/O characteristics of the ATLAS physics software. Cheaper $0.01/hour preemptible VMs exist, where the CPU can be stolen for other clients on GCP during such idle phases, however first tests showed that this is unstable with ATLAS analysis and causes job failures.

A second user workflow was investigated to see if and how cost can be controlled, using a lepton isolation analysis. The total input data was 92 TB, split into 40% Monte Carlo simulation data and 60% detector data. Rough processing time estimations at event sizes from 60kB to 120kB equal 2800 CPU hours, equal to 4 VMs for a full month. This could be reasonably processed by 100 VMs in a day, incurring the cost mentioned previously. When compared to grid-style analysis, the average user reads 900 Terabytes per month, and writes 6 Terabytes, though heavily dependent on the type of analysis.

6 Summary and future work

We integrated Google Cloud Storage into the ATLAS data management system Rucio, added Google Compute Engine into the ATLAS workflow management system PanDA using Harvester. We demonstrated that we can run regular ATLAS distributed analysis workload in file stage-in mode.

The missing direct-IO support is a showstopper for large-scale performance tests, and would also be necessary to better understand the latencies in wide-area reading. The job output stage-out was only supported to the CERN data centre, there needs to be future development on the Rucio side to support direct stage-out to GCS. The full third-party-copy transfers also have to be commissioned in production now that they have been implemented.

As an estimation of cost, usually analysis start with DAODs with 100-500 TB of Monte Carlo simulated data and detector data, this quickly becomes expensive in terms of storage. Further studies to understand the best price model for optimal storage and CPU usage need to be developed. Finally, we also foresee containerisation of user code instead of using grid-style tarballs or CernVMs.

References

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

[2] L. Evans and P. Bryant, *LHC Machine* JINST **3** S08001 (2008)

[3] M. Barisits, *Hybrid simulation models for data-intensive systems*, PhD Thesis, Technical University of Vienna (2017)

[4] Google Cloud Platform, https://cloud.google.com [accessed 2018-09-03]

[5] ACM/IEEE Supercomputing, *The International Conference for High Performance Computing, Networking, Storage and Analysis*, Denver, CO (2017)

[6] P. Calafiura, K. De, W. Guan et al., *The ATLAS Event Service: A new approach to event processing*, Journal of Physics: Conference Series **664** 062065 (2015)

[7] M. Barisits et al., *Rucio – Scientific Data Management*, Preprint, https://cds.cern.ch/record/2637240 [accessed 2018-09-03]

[8] T. Maeno, P. Nilsson, K. De et al., *PanDA Production and Analysis backend*, Journal of Physics: Conference Series, **219** (2009)

[9] T. Maeno et al., *Harvester: An edge service harvesting heterogeneous resources for ATLAS*, Preprint, https://cds.cern.ch/record/2625435 [accessed 2018-09-03]

[10] A. Ayllon, M. Salichos, M. Simon et al., *FTS3: New Data Movement Service For WLCG*, Journal of Physics: Conference Series **513** (2014)
[11] Amazon S3 Cloud Object Storage, https://aws.amazon.com/s3/ [accessed 2018-09-03]
[12] Google Cloud Storage – Signed URLs, https://cloud.google.com/storage/docs/access-control/signed-urls [accessed 2018-09-03]
[13] P. Nilsson et al., *Next Generation PanDA Pilot for ATLAS and Other Experiments*, Journal of Physics: Conference Series **513** (2014)
[14] US DoE LCF Theta, http://www.alcf.anl.gov/ [accessed 2018-09-03]
[15] US DoE LCF Cori, https://www.nersc.gov/ [accessed 2018-09-03]
[16] US DoE LCF Titan, https://www.olcf.ornl.gov/titan/ [accessed 2018-09-03]
[17] J. Blomer et al., *Delivering LHC Software to HPC Compute Elements with CernVM-FS*, LNCS **10524** (2017)
[18] Squid: optimising web delivery [software], Available from: http://www.squid-cache.org/ [accessed 2018-09-03]
[19] Google Cloud Platform Cost and Pricing, https://cloud.google.com/pricing/ [accessed 2018-09-03]