EPJ Web of Conferences **251**, 02005 (2021) https://doi.org/10.1051/epjconf/202125102005
CHEP 2021

Seamless integration of commercial Clouds with ATLAS Distributed Computing

Fernando Barreiro Megino[1], *Harinder Singh* Bawa[2], *Kaushik* De[1], *Johannes* Elmsheuser[3,*], *Alexei* Klimentov[3], *Mario* Lassnig[4], *Cédric* Serfon[3], and *Tobias* Wegner[5]

[1]University of Texas, Arlington, TX, USA
[2]California State University, Fresno, CA, USA
[3]Brookhaven National Laboratory, Upton, NY, USA
[4]CERN, Geneva, Switzerland
[5]Bergische Universität Wuppertal, Germany

Abstract. The CERN ATLAS Experiment successfully uses a worldwide distributed computing Grid infrastructure to support its physics programme at the Large Hadron Collider (LHC). The Grid workflow system PanDA routinely manages up to 700,000 concurrently running production and analysis jobs to process simulation and detector data. In total more than 500 PB of data are distributed over more than 150 sites in the WLCG and handled by the ATLAS data management system Rucio. To prepare for the ever growing data rate in future LHC runs new developments are underway to embrace industry accepted protocols and technologies, and utilize opportunistic resources in a standard way. This paper reviews how the Google and Amazon Cloud computing services have been seamlessly integrated as a Grid site within PanDA and Rucio. Performance and brief cost evaluations will be discussed. Such setups could offer advanced Cloud tool-sets and provide added value for analysis facilities that are under discussions for LHC Run-4.

1 Introduction

The distributed computing system [1] of the ATLAS experiment [2] at the LHC is built around two main components: the workflow management system PanDA [3] and the data management system Rucio [4]. The involved systems manage the computing resources to process the detector data at the Tier-0 at CERN, reprocesses it periodically at the distributed Tier-1 and Tier-2 Worldwide LHC Computing Grid (WLCG) [5] sites, and runs continuous Monte Carlo (MC) simulation and reconstruction. In addition, continuous distributed analyses from several hundred ATLAS users are executed. The resources used are the Tier-0 at CERN and Tier-1/2/3 Grid sites worldwide, opportunistic resources at High Performance Computing (HPC) sites, Cloud computing providers, and volunteer computing resources.

The ever growing needs for CPU and disk resources especially during the challenging LHC Run-4 planned to start in 2027 makes it necessary to explore new technologies and resource providers. The Cloud computing providers Google [6] and Amazon [7] offer the latest

*e-mail: johannes.elmsheuser@cern.ch

technologies in terms of software containerization and workflow scheduling using Kubernetes [8] and large scale storage accessible with industry-standard protocols such as HTTP and S3. These components will be used in the following to setup a transparent integration into PanDA and Rucio for production and user analysis workflows. They can be the basis for an analysis facility in the Cloud and be one ingredient to address the LHC Run-4 data processing challenges.

2 Data management in the Cloud with Rucio

The Rucio system manages ATLAS data all along their life cycle. In particular, Rucio enables files to be uploaded and downloaded from distributed storage systems. Rucio can also initiate transfers between two storage systems directly, a so called *third party copy* transfer, and negotiates across potential network links and transport protocols, such as HTTP, WebDAV, root, GridFTP, or similar. Third party copies are delegated by Rucio to the FTS [9] system, which executes the actual transfers. One of the two involved storage systems becomes the active partner in the transfer, the other becomes the passive partner. The active storage either pushes to, or pulls from, the other storage. It was thus necessary to ensure that these modes of operation are seamlessly integrated in the way Rucio maps the data namespace onto Cloud storage. Importantly, transfers to Cloud storage always require the active storage involved in the transfer to *push* the data using either the HTTP or S3 protocol. There is currently no possibility for Cloud storage to pull data from arbitrary storage with other protocols.

Cloud providers use signed URLs to transfer data in and out of their storage and, depending on the protocol, use different mechanisms to generate these URL signatures. A signed URL gives access to the object identified in the URL once for a given operation, and includes additional information, for example, an expiration date and time, that gives more control over this access to the content. Rucio was extended to accept and generate both Amazon and Google style signatures for all its operations. This requires storage of the private key of the Cloud account owner on the Rucio server in a secure environment. Rucio users never see or interact with the private key and this eliminates the need to distribute the private keys. The authorisation module of Rucio was extended to allow users to generate URL signatures selectively; non-privileged users will still see the existence of the data on the Cloud storage, but they won't be able to interact with it since Rucio will not generate a signature for them. Third party copy, on the other hand, requires that signed URLs are handed over from Rucio to FTS. This has two problems: the signature could expire before the transfer is able to leave the queue, and a potential security risk as the signatures could leak through to the monitoring systems. FTS and its underlying libraries GFAL and Davix were thus extended in the same way as Rucio, to accept and sign URLs only at the time of actual third party copy. Similarly, FTS has to store the private key of the Cloud storage owner account. Since even for large collaborations like ATLAS only two to three FTS instances are in place, this is considered an acceptable deployment complication.

An additional challenge was posed by the X.509 certificate infrastructure of the Google Cloud Storage. The trust of the certificates in transfers within WLCG sites is provided by the Inter-operable Global Trust Federation [10], however, it is important that the Cloud storage is also verifiable. The Cloud provider Certificate Authority (CA) has to be part of this federation to allow third party copy transfers. Amazon uses the DigiCert CA which is available in IGTF, however Google does not, as they use their own CA. Up to now there are still ongoing discussions to get the Google CA included in IGTF. There exists a workaround where one can set up a frontend with their own CA, however that incurs additional charges and is thus not recommended.

The storage at the Cloud provider was set up as follows. The Google Cloud Storage (GCS) was setup in a single object storage bucket close to the Kubernetes compute. The data was accessible through signed URLs using HTTPS with the Google signature protocol. The Amazon Web Services (AWS) storage was setup in an object storage bucket and accessible through signed URLs using the HTTPS protocol with the S3v4 signature algorithm.

The ATLAS data files in ROOT format [11] and O(1-10 GB) in size were transferred into the storage buckets using Rucio and processed by either copying them in one-go from the storage or streaming them into the payload using the HTTPS or S3 protocols, respectively.

3 Simulation of Cloud data management

ATLAS storage resources and commercial Cloud resources can be combined using different approaches. An approach that was investigated in more detail is the *Hot/Cold Storage* model. This model categorizes storage into three categories: Hot Storage, a limited storage resource which provides the best performance; Cold Storage which is significantly less limited than hot storage, globally accessible, but not as fast as hot storage; and Archival Storage, which provides the largest storage space but comes with a very high access latency.

In the Hot/Cold Storage model data required by jobs must exist on hot storage. The data are preferably transferred from cold to hot storage. If the data are only available on archival storage they are not only transferred to hot storage but also to cold storage. Data on cold storage are deleted based on a popularity metric, such as access frequency.

The Hot/Cold Storage model can be combined with the data carousel model [12] to the *Hot/Cold Data Carousel* (HCDC) model. The HCDC model targets the bandwidth and access latency bottlenecks of the data carousel model. The HCDC model can be used by the continuous ATLAS derivation production workflow [1]. In this model only tape storage would contain permanent replicas of the input data. The cold storage would be implemented by the commercial Cloud storage. The data are transferred from tape to Cloud storage based on their requirement or popularity metric.

A simulation framework [13] was developed to evaluate storage and network usage of models combining ATLAS resources with commercial Cloud resources. The HCDC model was simulated assuming Google Cloud Storage as cold storage. Three different configurations were simulated. The first configuration (I) did not include Cloud storage and had limitless hot storage available. The results of this configuration should show the best case where all data can be permanently stored on the hot storage. In the second configuration (II) the hot storage limit was changed to 100 TB. The results of the second configuration show how the job throughput is reduced when data have to be transferred from tape, each time they are required. The third configuration (III) keeps the limit on hot storage but includes limitless cold storage. Figure 1 shows for each configuration a histogram of the waiting time for the input data.

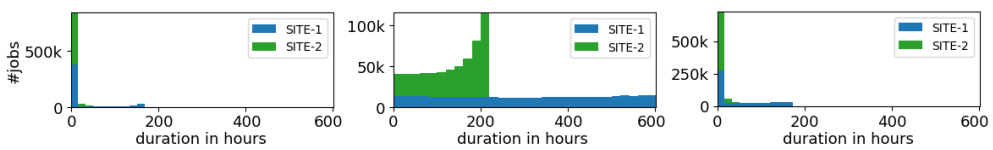


Figure 1: Waiting time distribution of the HCDC model for *configuration I* (left), *II* (middle), and *III* (right). The number of bins in the left and right histogram was reduced from 30 to 10 to improve visibility of the first bin. The simulation is based on the two Grid and Cloud sites SITE-1 and SITE-2 each with different storage element setups.

As expected, jobs in the first configuration have small waiting times, because the data is not deleted on hot storage. The waiting time in the second configuration is higher because data must frequently be transferred from tape storage. The results of the third configuration are similar to the results of the first configuration although the hot storage uses the same limit as in the second configuration. This shows that Cloud storage can improve the job throughput while keeping the hot storage requirements limited.

4 Integration of Kubernetes with the ATLAS workload management system

The workload management system PanDA is used to process, broker and dispatch all ATLAS workloads across the heterogeneous WLCG and opportunistic resources. The interface to the resources is handled by Harvester [14], which was implemented following a flexible plugin approach so that it can be easily expanded for any available resource. During the first proof of concept phase with Google in 2017 [15] the ATLAS payloads on Google Compute Engine (GCE) had to be integrated in the most native and lightweight way possible. At that point Harvester plugins were implemented to manage the life cycle of VMs (Virtual Machines) using the GCE python API. Through contextualization the VMs were starting the PanDA pilot [16] and executing ATLAS payloads. This model worked correctly, but apart from other smaller inconveniences it was not generic enough and could only be applied to GCE.

During that time containerization and Kubernetes were getting increasingly relevant in Cloud environments providing a secure, lightweight, standardized way to package and execute software. Kubernetes was available on major Cloud providers, e.g. Amazon, Azure, Google and Oracle. On the ATLAS side, payloads were being containerized and some collaborating institutes were interested in evaluating Kubernetes as a batch system. Therefore we started a project to implement Kubernetes plugins in Harvester and partnered with a few institutes to evaluate this model [17].

4.1 Harvester-Kubernetes plugins

Figure 2 shows the schema of the PanDA and Harvester setup to use Kubernetes for job execution. Most of the Harvester complexity is implemented in the Core. Kubernetes plugins using the python API will do the following specific integration tasks:

- Credential manager: periodically updates a short lived certificate to the Kubernetes cluster through Kubernetes secrets. This certificate is used by the Pilot to authenticate towards the storage or the PanDA server.
- Submitter: submits Kubernetes job controllers, specifying the memory and CPU needed by the jobs. While various shapes exist, the typical requirements are 1 core and 2 GB of memory, or 8 core and 16 GB of memory. The jobs use Kubernetes affinity so that the single core jobs attract each other, rather than spreading across the cluster and preventing the multi core jobs from being scheduled.
- Monitor: gets the status of the Kubernetes jobs and updates it in the Harvester database. To narrow down the selection of jobs and prevent scalability issues on large clusters, each job is labeled with its ID and the monitor specifies the label selector when listing the jobs.
- Sweeper: performs the clean-up for completed jobs.

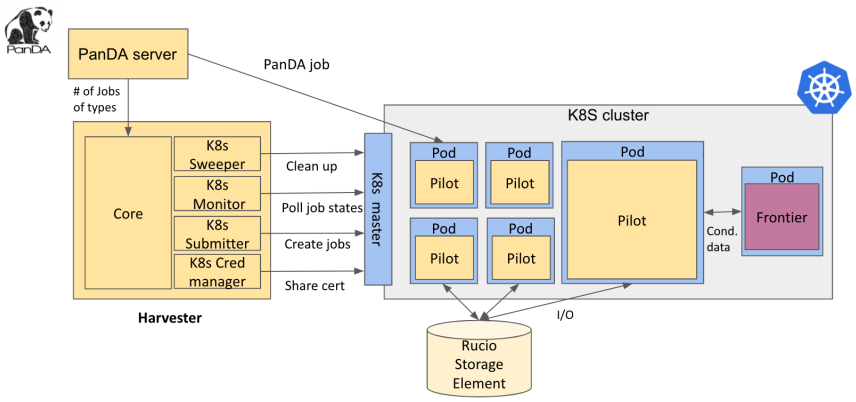


Figure 2: Schema of the PanDA and Harvester setup for job submission to Kubernetes clusters.

4.2 CVMFS

The only requirement on the cluster is to have CVMFS [18] installed, since CVMFS is used to distribute all ATLAS software. On static clusters the most stable option is to install CVMFS directly through the package manager. On auto-scaled clusters or clusters where nodes can be exchanged dynamically, for example Google pre-emptible instances and Amazon Spot instances, direct installation is not possible and CVMFS needs to be installed as a daemonset. There are different solutions that other projects in the High Energy Physics (HEP) environments have developed: CVMFS CSI driver [19] and the CVMFS OSG driver [20]. Both options have a very similar daemonset concept with a privileged CVMFS pod running on each node and sharing a volume to the other pods in the node.

- The CSI driver shares the volume by implementing the CSI standard functions in Golang.
- The OSG driver shares the volume through a local volume and is much simpler to operate.

Slight adaptations of the OSG driver were made and liveness probes were added to verify all CVMFS mounts are present. Independently of the driver, it is of critical importance to specify adequate resource specifications (memory and CPU requests) for the pod, since otherwise it will get killed when the node is out of memory or be extremely slow when the node is out of CPU. With these considerations the OSG driver has worked very stably for us.

4.3 Frontier squid

ATLAS uses a Frontier squid [21] hierarchy to distribute detector conditions data that describes the environment under which the LHC collision events were taken. The squid can also be used to cache CVMFS for the cluster. If a Grid site is not connected to a squid cache, the requests from all individual nodes are directed to the Tier-1 launchpads at the top of the hierarchy. The Tier-1 launchpads are a critical service and it's important to protect them. Therefore a squid cache has to be setup in a Cloud VM next to the Kubernetes processing cluster or directly in the cluster as a pod. The frontier pod is exposed as a Kubernetes service controller so that the internal IP does not change even when the pod is moved in the cluster.

4.4 Service accounts

Harvester does not require administrator privileges on the clusters, but only a restricted set of actions to interact with jobs, pods and secrets. The actions can be limited to a certain namespace. The usual procedure is to generate a service account that defines the scope of action for Harvester. Depending on the Cloud provider it is possible to generate service accounts directly through the admin interface, or otherwise it is possible to generate them directly in the Kubernetes cluster. These settings are extracted to a Kubeconfig file per cluster that contain information about the user, the cluster endpoint and the certificate. They are placed in Harvester to communicate with the cluster.

4.5 Infrastructure choices in the Cloud

Commercial Cloud providers offer a wide range of options for virtual machine sizing and performance. During the current tests at Google and Amazon we did not attempt to do a complete study or benchmarking of all available options, but did gain some experience in the choices to make, depending on the different job requirements. During our exercises we typically run two types of jobs:

- Monte Carlo simulation jobs: multi core jobs that are very CPU intensive and have very low I/O demands. Given the low I/O requirements the worker node can download the input from, and upload the output to, a remote storage without heavily penalizing the CPU efficiency or congesting the network. For these jobs we usually select the most affordable virtual machines with 8vCPU, 16GB of RAM (or the default RAM selection) and the lowest-end storage (i.e. spinning network disk) for the creation of our Kubernetes cluster. In the case of Amazon it is not recommended to use "Burstable" instances, since the jobs will exceed the CPU baseline and incur in additional costs. We did not compare the processing times of different CPU families in the same Cloud, but it would be interesting to see which CPU family offers the best price/performance relation for these jobs.
- User analysis: single core jobs with frequently very high I/O requirements. Preferably, these jobs are executed close to the storage endpoint. Partially because of our "in vitro" conditions and the short expiration times of the signed URLs provided by Rucio to interact with the storage, our analysis exercises required a high throughput to the VM disk. It is important to pay attention to the throughput rates of the devices you choose in your setup. Cloud providers can have throttle policies to protect their network and share it across the tenants. In the case of the Google Cloud storage, there are different strategies you can consider in order to increase the disk throughput, such as over allocating disk (throughput can be a function of the disk size) or taking faster devices like SSDs (see Figure 3). In the case of local SSDs, the fastest option, you will need to see the availability and conditions. The output of these jobs is usually much smaller in size compared to the inputs and chosen to be uploaded to the Cloud storage using Rucio. The signed URLs for the output files on the Cloud storage are generated during the stage-out process.

4.6 Kubernetes mini-grid

The Harvester-Kubernetes model has proven a very interesting solution. A few ATLAS sites have shown interest to run (a part of) their batch services as a Kubernetes cluster. Regarding commercial Clouds, the integration has been demonstrated at different scales on Google, Amazon and Oracle (the latter only to run some test jobs on a trial account). During 2020 we have run a 1000-2000 core Kubernetes mini-grid (see Figure 4) with several participants

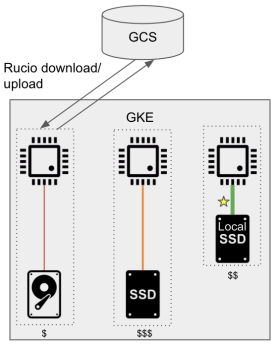


Figure 3: Options in Google Cloud Storage (GCS) for increasing throughput to disk.

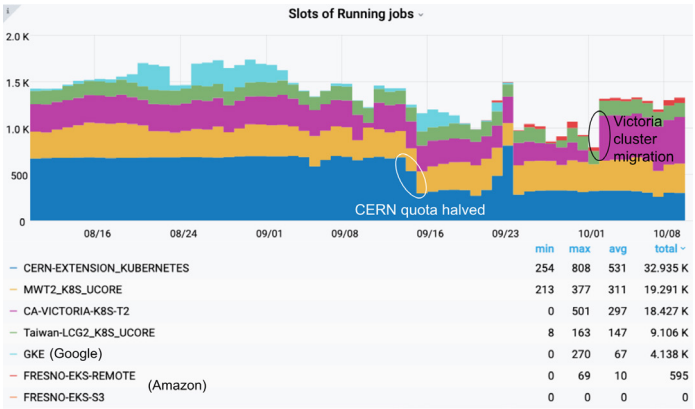


Figure 4: Number of running job slots using Harvester and Kubernetes at various Grid and Cloud sites in August-October 2020. The successful wallclock consumption was 91 per cent during this period.

and dedicated significant effort to stabilize operations. We managed to bring down the failure rate to levels comparable to the Grid. Eventually the failure rate depends on a combination of issues at each site, the associated storage and also other SW or central infrastructure issues.

5 Production and Analysis jobs performance

Using the compute and storage setup described in the previous sections, we first demonstrated that ATLAS production input data can be automatically transferred in and out of the Cloud storage using Rucio. Figure 5 shows the throughput rate in GB/s to the Google Cloud storage from various Grid sites. Data transfers are possible at the same or even higher rates than between currently used Grid sites. One aspect to consider is that the Grid sites are largely connected through the general purpose network to Cloud providers and are not using dedicated links on the LHCOPN/LHCONE networks. As a second step we demonstrated that continuous and stable production and user analysis using the previously described Harvester and Kubernetes workflow path can be successfully used. Figure 6 (a) shows the number of running job slots for production and user analysis on Google Kubernetes Engine (GKE). The

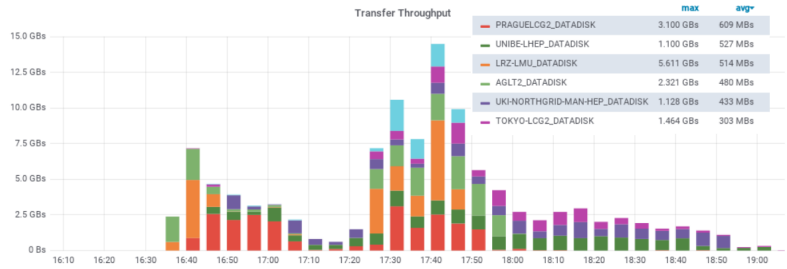


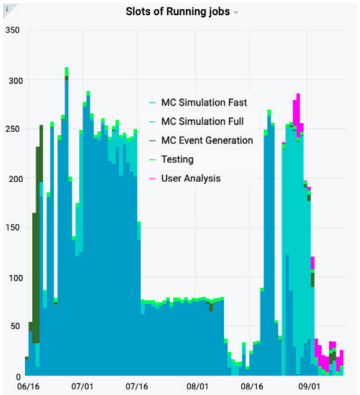
Figure 5: Rucio throughput to GCS during a test transfer of about 50 Terabytes split by source Grid sites. These Grid sites store a fraction of the ATLAS data files on Rucio storage elements called DATADISK.

number of slots is not always flat, because we were actively working on different activities and redistributing the resources according to internal priority. Pre-emptible worker nodes have been used for production jobs in order to save costs. Pre-emptible nodes are made available by Google at a significantly reduced cost with the limitation that they only can live up to 24 hours, but could be evicted earlier if there is a fully paying Cloud tenant. Pre-emptible nodes are required to find a balance between maximal job duration and acceptable failed walltime. For example jobs with a duration over 24 hours will never succeed on a pre-emptible node. Our choice was to only broker jobs with a duration under 5 hours, which lead to a failed walltime under 10%. This loss was largely outweighed by the cost savings. Failed jobs are automatically retried by PanDA and in reasonable conditions do not require additional operational load.

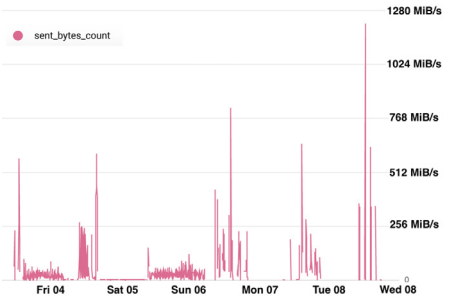
User analysis jobs have been scheduled on powerful standard (non pre-emptible) worker nodes to avoid our users having to wait for some of the failed jobs to be automatically retried. Figure 6 (b) shows the Google Cloud Storage throughput rate in MB/s that is mainly dominated by analysis job input file reading and demonstrates that the large I/O demands from user analysis can be met by the Cloud storage. As described in Section 2 input files are either downloaded once before the user payload is started on the worker node or streamed into the running user payload. Both modes work fine with AWS storage while with GCS only the first mode works at scale right now. There had been so far only a very small number of analysis users and the dataset size permanently stored and analysed was in the O(TB). Non-standard ATLAS workflows like using GPU in addition to CPU processing could be easily added.

The list-price costs running with a GKE cluster of O(200) CPUs and a small local GCS storage in the O(TB) and remote stage-out of the production data was about 2.3k USD/month (77 USD/day) in July and 1.7k USD/month (54 USD/day) in August 2020 including all costs and in particular network egress costs. Depending on the Cloud service choices the Cloud CPU and storage prices can be more than a factor of 5 higher applying the list prices in comparison to a Tier2 Grid site. This applies especially to continuous operations. But Cloud computing resources can very flexible requested on-demand in case of short notice resource needs. The price difference will be lower if staffing costs and costs for power and housing are considered. Pre-emptible CPU slots or academic discounts will reduce the costs for Cloud computing usage further. One non-negligible factor remains the network egress costs in Cloud computing which constrains some of the workflows in terms of costs.

We have also operated PanDA queues on Amazon Elastic Kubernetes Service (EKS). Amazon offers their unused capacity in the Spot market and users can bid for those resources. When providing a large enough bid it is possible to obtain virtual machines without a prede-



(a)



(b)

Figure 6: (a) The number of running jobs slots for production and user analysis on Google Kubernetes Engine. (b) The Google Cloud Storage throughput rate in MB/s mainly dominated by analysis job input file reading.

Figure 7: The number of running job slots on Amazon Elastic Kubernetes Service.

fined deadline. In practice we have used a bid of 0.16 USD/hour (list price is 0.384 USD) for m5.2xlarge machines (8 vCPU and 32GB RAM). With this bid we have been running a stable, uninterrupted 160 core cluster since September 2020 (see Figure 7). We have encountered one important issue, where the EKS nodes stop receiving jobs after one or two weeks. This is caused by a bug in an old systemd version used in the Amazon Machine Image (AMI), which does not clean up correctly transient mounts and makes the node unschedulable. This explains the periodic drops in Figure 7. Currently we are terminating those nodes manually, but a permanent solution needs to be found. We didn't evaluate jobs with heavy I/O requirements.

6 Summary and Conclusions

Google and Amazon Cloud computing services have been successfully integrated within the ATLAS workflow and data management systems PanDA and Rucio. State-of-the-art tech-

nologies like Kubernetes and Cloud storage are used and work at the scale of today's medium sized Grid sites. The Cloud provider list prices are still higher than currently operating Grid sites. The discussed solutions are not ATLAS specific and can be easily adapted by other HEP or non-HEP communities, and may be considered for example in the implementation of analysis facilities for LHC Run-4, with all the added values and services offered by the commercial Cloud providers.

References

[1] J. Elmsheuser et al. [ATLAS Collaboration], *Overview of the ATLAS distributed computing system,* EPJ Web Conf. **214** 03010 (2019).

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

[3] F. H. Barreiro Megino *et al.*, *PanDA for ATLAS distributed computing in the next decade*, J. Phys. Conf. Ser. **898** (2017) no.5, 052002

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

[5] Worldwide LHC Computing Grid, URL http://cern.ch/lcg [accessed 2021-01-06]

[6] Google Cloud Services, URL https://cloud.google.com [accessed 2021-05-26]

[7] Amazon Web Services Services, URL https://aws.amazon.com [accessed 2021-05-26]

[8] Kubernetes, URL https://kubernetes.io/docs/home/ [accessed 2021-01-06]

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

[10] IGTF: Interoperable Global Trust Federation, URL https://www.igtf.net/ [accessed 2021-01-06]

[11] ROOT, URL https://root.cern/about/ [accessed 2021-01-06]

[12] M. Barisits *et al.*, *ATLAS Data Carousel*, EPJ Web Conf. **245** (2020), 04035

[13] T. Wegner *et al.*, *Simulation and evaluation of cloud storage caching for data intensive science*, arXiv:2105.03201

[14] F. H. Barreiro Megino *et al.*, *Managing the ATLAS Grid through Harvester*, EPJ Web Conf. **245** (2020), 03010

[15] M. Barisits *et al.*, *The Data Ocean project: An ATLAS and Google R&D collaboration*, EPJ Web Conf. **214** (2019), 04020

[16] P. Nilsson *et al.*, *The next generation PanDA Pilot for and beyond the ATLAS experiment*, EPJ Web Conf. **214** (2019), 03054

[17] F. H. Barreiro Megino *et al.*, *Using Kubernetes as an ATLAS computing site*, EPJ Web Conf. **245** (2020), 07025

[18] J. Blomer *et al.*, *The CernVM File System*, https://doi.org/10.5281/zenodo.4114078

[19] CVMFS CSI, URL https://github.com/cernops/cvmfs-csi, [accessed 2021-01-06]

[20] CVMFS OSG, URL https://github.com/opensciencegrid/osg-k8s-cvmfs, [accessed 2021-01-06]

[21] Frontier system, URL http://frontier.cern.ch, [accessed 2021-05-25]

We are grateful to Eric Schanet (LMU Munich) for thoroughly testing the analysis workflows. We are also grateful to Ema Kaminskaya, Usman Qureshi, Karan Bhatia, Ross Thomson, Kevin Kissell, Miles Euell and Dom Zippilli from Google for the fruitful discussions and the excellent support. We thank Edward Karavakis and Mihai Patrascoiu from the CERN FTS team, Tadashi Maeno (BNL) and Fahui Lin (UTA) from the PanDA team, Mandy Yang from ASGC, Misha Borodin (IU) from the ATLAS production system team, Andrew Hanushevsky (SLAC) and Douglas Benjamin (ANL).