EPJ Web of Conferences **214**, 07009 (2019) https://doi.org/10.1051/epjconf/201921407009
CHEP 2018

Integrating a dynamic data federation into the ATLAS distributed data management system

Frank Berghaus[1,*], *Tobias* Wegner[2], *Mario* Lassnig[2], *Marcus* Ebert[1], *Cedric* Serfon[3], *Fernando* Galindo[4], *Rolf* Seuster[1], *Vincent* Garonne[2], *Reda* Tafirout[4], and *Randall* Sobie[1], on behalf of the ATLAS Collaboration

[1]University of Victoria, Victoria, Canada
[2]CERN, Geneva, Switzerland
[3]University of Oslo, Oslo, Norway
[4]TRIUMF, Vancouver, Canada

Abstract. Input data for applications that run in cloud computing centres can be stored at remote repositories, typically with multiple copies of the most popular data stored at many sites. Locating and retrieving the remote data can be challenging, and we believe that federating the storage can address this problem. In this approach, the closest copy of the data is used based on geographical or other information. Currently, we are using the dynamic data federation, Dynafed, a software solution developed by CERN IT. Dynafed supports several industry standard interfaces, such as Amazon S3, Microsoft Azure and HTTP with WebDAV extensions. Dynafed functions as an abstraction layer under which protocol-dependent authentication details are hidden from the user, requiring the user to only provide an X509 certificate. We have set up an instance of Dynafed and integrated it into the ATLAS distributed data management system, Rucio. We report on the challenges faced during the installation and integration.

1 Introduction

We would like to run data-intensive applications on globally distributed opportunistic resources that have no local storage. The ATLAS [1] experiment leverages a globally distributed system of infrastructure as a service (IaaS) clouds as part of its distributed computing system. These resources are integrated into the ATLAS distributed computing system using the Cloudscheduler [2] technology developed at the University of Victoria. These IaaS resources do not support any local grid infrastructure.

The workflows executed by high energy physics experiments often demand large volumes of input data or produce a significant volume of output data. We want to use a data federation, such as Dynafed [3], to redirect the applications running on opportunistic resources to the optimal storage endpoint to retrieve input or deposit output data. We also aim to integrate storage solutions offered by cloud providers into the ATLAS distributed data management system using Dynafed.

Integrating Dynafed into ATLAS distributed computing presented challenges in two categories: use of the HTTP protocol instead of the standard XRootD protocol [4] on the worker

*e-mail: berghaus@cern.ch

nodes and again for transfers between Dynafed and grid storage sites which use GridFTP [5] for 3rd party copy orchestrated by the file transfer system (FTS) [6].

In this paper we describe a system leveraging Cloudscheduler and Dynafed, which successfully executed functional test jobs and transfers as part of the ATLAS distributed computing and data management systems on the CERN OpenStack [7] cloud resource. Data were read from and written to the Dynafed configurations listed in table 1. The cloud and grid scheme are set up on a single Dynafed instance under two different paths. The read-only scheme is set up under a separate Dynafed instance because Rucio, at the time of writing, only allowed assigning different read and write endpoints if the host name or protocol for those operations were different. Most attention is directed to the cloud configuration over the grid configuration[1]. In the cloud configuration, data were read from and written to an object store implemented using Ceph [8] and exposing an S3 compatible gateway.

Table 1. Three Dynafed endpoints tested for ATLAS.

Endpoint	Storage provider	Storage type
cloud	CERN	Ceph S3
grid	CERN	EOS
	LRZ in Munich	dCache
	ECDF in Edinburgh	DPM
read-only	ATLAS WLCG Sites	EOS, DPM, dCache, StoRM

2 Data access

The ATLAS experiment leverages the resources of the Worldwide LHC Computing Grid (WLCG) [9]. The computing centres that are part of the WLCG and support ATLAS provide a global storage infrastructure for the experiment data and simulated events. They may be accessed using standard protocols, such as HTTP with WebDAV extensions. Dynafed supports storage back-ends that expose these protocols and provides sufficient scalability to create the appearance of a single virtual name space for the entire ATLAS data catalogue. Furthermore, Dynafed allows the usage of cloud storage systems such as S3, SWIFT, and Azure with the same user interface. The virtual data catalogue is presented as a browsable file system. Requests for files redirect the user to the closest copy of the file. Write requests direct the user to the nearest writable storage element.

Three different Dynafed schemes were tested as illustrated in Table 1. The cloud and grid scheme were added to ATLAS distributed data management as storage elements[2] `CERN-EXTENSION_CLOUD` and `CERN-EXTENSION_GRID`. The read-only scheme was added as the read protocol of both storage elements. Users access data using the Rucio client. To allow seamless interaction, part of the checksum verification had to be disabled; more on this in section 2.2.

2.1 Authentication and authorization

Dynafed implements authentication and authorization using a X509 public key infrastructure. It also supports grid security infrastructure extensions of X509 with VOMS attributes [10]. Once the request has been authorized inside Dynafed, the next step depends on the type of

[1] Grid storage means storage endpoints providing their service using the DPM, dCache, EOS, or StoRM software.
[2] or DDMEndpoint

endpoint connected: the client authenticates (a second time), this time against the storage endpoint using their X509 credentials when forwarded to storage supporting this mechanism. For cloud storage Dynafed redirects the authorized client to a signed URL. The signature is generated from the credentials specified in the Dynafed configuration and grants access to the storage endpoint.

2.2 Checksums

The standard work flow to upload data to the ATLAS distributed data management system, Rucio, is: (i) calculate checksum of local file, (ii) upload file, (iii) request checksum from remote storage, (iv) verify that local and remote checksum match, and (v) add file and checksum to Rucio database. An upload is only considered successful if all these steps complete successfully. Similarly, downloading a file involves a verification of the checksum of the file on the remote storage before downloading the file to the local disk.

Unfortunately, Dynafed cannot answer the request for the remote checksum. On the grid, checksums are communicated using the Want-Digest [11] header. Cloud storage supports the Content-MD5 [12] header on upload and an Etag header afterwards. The essentially different nature of communicating checksums means some thoughtful addition to Dynafed, Rucio, and the grid file access libraries must be implemented to support the retrieval of a checksum uniformly, hiding the storage implementation from the user. A candidate solution is to add methods to Dynafed that execute the proper requests to the back-end storage, cache the response, and answer the request by the user. A possible way to allow cloud storage to return checksums other than MD5 is to add those checksums as key-value pairs to the file object.

ATLAS uses the ADLER32 checksum to ensure file integrity. Given that cloud and object stores support only MD5, functionality was added to the Rucio client to calculate the MD5 checksum on upload in addition to the ADLER32. On download, functionality was added that attempts to verify the MD5 checksum if the ADLER32 checksum is not available.

To allow functional tests of the Dynafed storage element they were labelled `verify_checksum = False` in the Rucio database. Functionality was added to the Rucio clients and conveyor daemon[3] to skip checksum verification against any storage element labelled with `verify_checksum = False`.

3 Results

With the additions to the Rucio clients and conveyor, transfers of 1 `MB` files were scheduled at a rate of O(10) per minute to test Dynafed and the underlying storage. Transfers against the cloud setup achieved a success rate of 70%. The reason for transfer failures varied depending on the storage and were adressed in turn:

- The DPM at our disposal was the Edinburgh development setup and failed when the site admins where carrying out tests.
- The EOS had intermittent and frequent issues with their HTTP gateway.
- The dCache had some issues with the legacy proxy Rucio was using to order transfers. This was fixed by migrating Rucio to RFC compliant proxies.

New transfer efficiency measurements are ongoing at the time of writing. In the majority of transfers, the data had to be streamed through the File Transfer System, which Rucio uses to orchestrate transfers between storage elements. While most grid storage software supports

[3]The Rucio conveyor daemon orders transfers between storage elements.

3rd party copy using HTTP, most storage elements in the WLCG are configured not to allow it, or implement a version of the software that does not support it. Support for 3rd party copy in Dynafed is under active development, and a campaign to ensure all grid sites support 3rd party copy for both the XRootD and HTTP protocols is under way as a part of the data organization, management, and access (DOMA) project of the WLCG.

The same cloud resources operated for ATLAS by Cloudscheduler were set to run functional test jobs against the Dynafed storage elements. These jobs were successful when accessing data with the Rucio clients implementing the changes discussed above. Clients retrieved the geographically closest copy of data to them on download, and uploaded data to their closest write-able storage element.

4 Conclusions

Dynafed was integrated into the ATLAS distributed computing and data management architecture. Functional tests of Dynafed are successful. Further development is needed within Dynafed and Rucio to allow running Dynafed in production: support for checksums verification and 3rd party copy. An interesting future avenue of work is the integration of volatile caches with Dynafed.

References

[1] The ATLAS Collaboration, JINST **3**, S08003 (2008)
[2] Cloudscheduler project, ”Cloudscheduler” [software], version 1.12, 2018, Available from http://cloudscheduler.org [accessed 2018-08-16]
[3] Dynafed project, ”Dynafed” [software], version 1.3.1, 2017, Available from http://cern.ch/lcgdm/dynafed-dynamic-federation-project [accessed 2018-08-16]
[4] XRootD project, "XRootD" [protocol], version 3.1.0, Available from http://xrootd.org [accessed 2018-10-10]
[5] Globus, "GridFTP" [software], version 3.17.1, Available from http://toolkit.globus.org/toolkit/docs/latest-stable/gridftp [accessed 2018-10-10]
[6] File Transfer Service project, "File Transfer Service", version 3, Available from http://fts.web.cern.ch [accessed 2018-10-10]
[7] Openstack project, ”OpenStack” [software], various versions, Available from https://www.openstack.org [accessed 2018-08-16]
[8] Ceph project, ”Ceph” [software], various versions, Available from http://ceph.com [accessed 2018-08-16]
[9] Bird I, Ann Rev Nucl Part Sci **61**, 99 (2011)
[10] Foster I, Kesselman C and Tuecke S, IJHPCA **15** 3 200 - 222 (2001)
[11] Mogul J and Van Hoff A, Internet Engineering Taskforce **RFC** 3230 (2002)
[12] Myes J and Rose M, Internet Engineering Taskforce **RFC** 1864 (1995)