Federating Distributed Storage For Clouds In ATLAS

Berghaus F[1], Casteels K[1], Di Girolamo A[2], Driemel C[1], Ebert M[1], Furano F[2], Galindo F[3], Lassnig M[2], Leavett-Brown C[1], Paterson M[1], Serfon C[4], Seuster R[1], Sobie R[1], Tafirout R[3], Taylor R P[1]

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

E-mail: frank.berghaus@cern.ch

Abstract. Input data for applications that run in cloud computing centres can be stored at distant repositories, often with multiple copies of the 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. A federation would locate the closest copy of the data on the basis of GeoIP information. Currently we are using the dynamic data federation DynaFed [1], a software solution developed by CERN IT. DynaFed supports several industry standards for connection protocols like Amazon's S3, Microsoft's Azure, as well as WebDAV and HTTP. 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 setup an instance of DynaFed and integrated it into the ATLAS Data Distribution Management system. We report on the challenges faced during the installation and integration. We have tested ATLAS analysis jobs submitted by the PanDA [2] production system and we report on our first experiences with its operation.

1. Introduction

Our goal is run data-intensive applications on globally distributed opportunistic resources that have no local grid storage. The ATLAS 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 Cloud Scheduler [3] technology developed at the University of Victoria. These IaaS resources are used opportunistically, and 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 aim to use a data federation, such as DynaFed, 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.

In this paper we explain a system leveraging Cloud Scheduler and Dynafed which successfully executed functional test jobs as part of the ATLAS distributed computing system on the CERN OpenStack [4] cloud resource that read their input from and wrote their output to an object store implemented using Ceph [5] and exposing an S3 [6] interface.

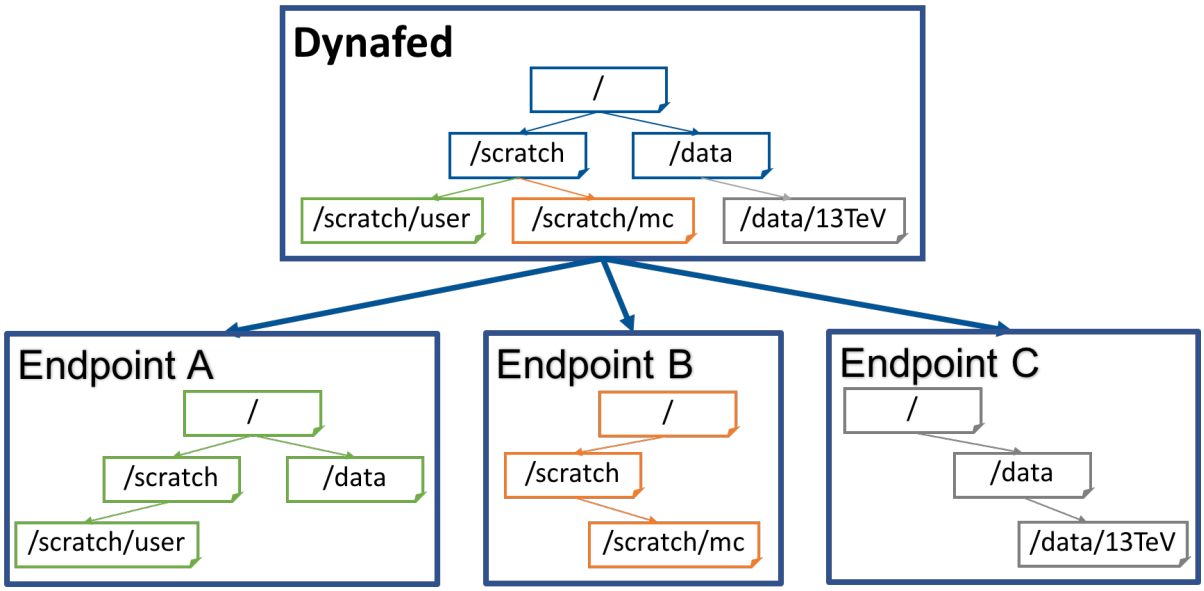

Figure 1. The dynamic federation is connected to multiple endpoints. Each endpoint may be a file system or an object store accessible using a protocol which allows redirection, such as WebDAV or HTTP. The dynamic federation provides a namespace that is a union of all the namespaces of the endpoints. That namespace is presented as a regular directory structure by the Dynafed. The content of a displayed directory is calculated when accessed.

2. Conceptual Design

The ATLAS experiment leverages the resources of the Worldwide LHC Computing Grid, WLCG [7]. The computing centres that are part of the WLCG and support ATLAS provide a global storage infrastructure for the experiment data and simulated events. While the central ATLAS computing infrastructure uses purpose-specific protocols to access the content of these grid storage elements, they can also be accessed using standard protocols, namely HTTP and WebDAV. DynaFed supports storage backends that offer HTTP and WebDAV access and promises sufficient scalability to create the appearance of a single virtual namespace for the entire ATLAS data catalogue. Figure 1 shows how DynaFed could unify the namespaces of attached storage elements into a single namespace.

Cloud storage systems are object stores that expose a well-defined interface such as S3, Swift, or Azure. DynaFed supports these protocols as storage backends. On the user-facing side DynaFed still presents an HTTP or WebDAV interface implementing authentication and authorization through public key infrastructure with grid extensions [8]. When DynaFed forwards clients to cloud storage systems, it translates their grid credentials to pre-signed URL that permit, for a limited time, access to the cloud storage system.

3. Data Access

The dynamic federation used for this work was configured to use three endpoints: one at CERN, one at TRIUMF, and one at the University of Victoria. Each endpoint was a CephS3 object store. Figure 1 illustrates the task division within the dynamic federation to handle client requests.

To access data through DynaFed a client makes a request for a file using WebDAV or HTTP. WebDAV is the appropriate protocol for interacting with files and file systems, so we will focus on it from here on. In our configuration the DynaFed only allows access to members of the

Table 1. The dynamic web federation is an Apache running the LCGDM implementation of WebDAV. The namespace usually managed by LCGDM has been replaced by the uniform general redirector (UGR) which translates the requests to the web file system to the connected endpoints. The endpoint modules handle the communication with the configured endpoints. All requests are cached in memory on the server as well as in a second-level cache which may be shared across multiple load-balanced servers.

Component		Purpose
Apache		Load the `lcgdm_dav` module and start up a WebDAV server
`lcgdm_dav` dmlite		Configure dmlite and load the uniform general redirector as namespace plugin
UGR		Configure authentication and endpoints
Plugins	dmlite WebDAV/HTTP S3 Azure	Communicate with endpoints on request
Memcached		Cache recent redirects to distributed object caching system

ATLAS Virtual Organization. That means the client must provide either a certificate and key combination or a proxy credential with the request. The credential must be signed by a trusted certificate authority. If the credential contains VOMS extensions certifying the user to be a member of the ATLAS collaboration, access is granted. Should the client provide an X509 credential without VOMS extensions the DynaFed checks that credential against all current members of the ATLAS collaboration and grants access if a match is found. Finally, there are privileged accounts for administrators and data management services. Authorized clients are redirected to a signed URL on the closest CephS3 endpoint. Authorization is granted explicitly for reading, writing, listing, and/or deleting operations.

When a file is requested the dynamic federation checks whether the locations of the file are already in its cache. If so the cached entry is used, otherwise each endpoint is queried for the file after name translation to that endpoint. DynaFed waits for responses up to some given timeout[1]. Responses are collected and cached. The resulting endpoints are evaluated for proximity to the client and the client is redirected to the closest copy. The DynaFed regularly polls all connected endpoints to determine if they are reachable. Should an endpoint be unresponsive, requests will not be forwarded to it. This should increase the stability of the storage system as a whole by dynamically adapting to storage endpoint failures.

It is also possible to query a metalink which returns a XML list of all copies of the queried file. In the future we wish to implement chunked downloads from multiple locations using this metalink and the aria2c copy tool. Should the client make a request to write, Dynafed redirects the client to the geographically closest writeable storage element.

4. Application Workflow

In order to integrate the dynamic federation into the ATLAS distributed computing and data management system, it was defined as a storage element associated with the CERN-EXTENSION grid site. It was configured to be accessible using WebDAV and flagged as special in the ATLAS grid information system to allow Rucio to select a n implementation of a copy tool that does not move or rename files.

The input datasets for analysis and production functional tests were transferred to the dynamic federation using the file transfer service [9] at CERN. Once the transfers completed

[1] Set to 3 seconds here

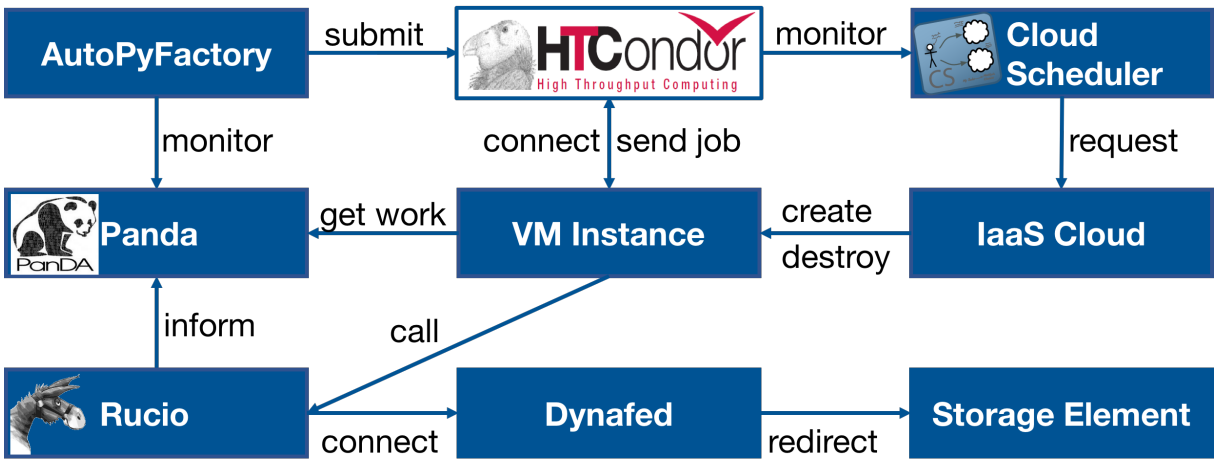

Figure 2. A client (AutoPyFactory or Harvester) submits pilot wrapper scripts to an HTCondor queue. The queue is monitored by a Cloud Scheduler. The Cloud Scheduler makes requests to connected cloud interfaces in response to queued jobs. The cloud infrastructures create virtual machine instances and provide user-data to cloud-init running in the virtual machines for configuration. The VM instances are configured to connect to HTCondor and start consuming jobs from the queue. The pilot wrapper scripts run on the virtual machines and download the pilot which drains tasks from a PanDA queue. The pilot uses Rucio to download input data and upload results. Rucio is configured to contact the dynamic federation using the WebDAV protocol. The federation forwards Rucio to the closest available storage element.

successfully the data was registered manually in the Rucio data catalogue.

With the input data registered in the Rucio data catalogue it was possible to run grid jobs against the data in the dynamic federation. The jobs were executed on virtual machines hosted on the CERN OpenStack using the Cloud Scheduler technology system illustrated in figure 2. The resulting data and logs were uploaded to the CephS3 storage via DynaFed upon job completion.

Some additional development is required for full integration of cloud storage into the ATLAS distributed data management: bulk transfers negotiated between storage endpoints using the HTTP or WebDAV protocols must be fully supported, and the data management system must be able to parse the checksums of files on cloud storage. The first point requires upgrades to the logic of Rucio's conveyer mechanism. The second point requires Rucio to fully support MD5 [10] checksums, which are commonly used in cloud storage systems, alongside ADLER32 [11] checksums which are the standard on the worldwide LHC computing grid. This will likely involve some additional logic to the grid file access libraries since these checksums are also communicated in a different fashion on cloud storage as opposed to grid storage [12, 13].

5. Summary

It was shown that ATLAS jobs can retrieve and deposit their data on a cloud storage system accessed via a dynamic federation using the WebDAV protocol. The jobs ran on virtual machine instances in a cloud and could be scheduled anywhere in the distributed cloud system currently running as part of the ATLAS production system. Further development is necessary to allow the execution of production or analysis jobs against the dynamic federation. Work is ongoing to integrate the dynamic federation with the Belle-II experiment and thereby the DIRAC workload management system. While this development is being pursued against opportunistic cloud

resources it would also be very useful in the context of volunteer resources [14].

References

[1] Dynafed [Compute Software] 2017 `http://cern.ch/lcgdm/dynafed-dynamic-federation-project`
[2] Maeno T 2008 PanDA : Distributed Production and Distributed Analysis System for ATLAS *Journal for Physics* **119** 6
[3] Cloud Scheduler [Computer Software] 2017 `http://cloudscheduler.org`
[4] OpenStack [Computer Software] 2017 `https://www.openstack.org`
[5] Ceph [Computer Software] 2017 `http://ceph.com`
[6] Amazon Simple Storage Service [Computer Software] 2006 `https://aws.amazon.com/s3/`
[7] Bird I 2011 Computing for the Large Hadron Collider *Ann. Rev. Nucl. Part. Sci.* **61** 99
[8] Foster I, Kesselman C and Tuecke S 2001 The Anatomy of the Grid: Enabling Scalable Virtual Organizations *International Journal of Supercomputer Applications* **15** 3 200 - 222
[9] CERN IT-ST 2017 FTS3 `http://fts3-service.web.cern.ch/`
[10] Rivest R 1992 The MD5 Message-Digest Algorithm *Internet Engineering Task Force* RFC1321
[11] Deutsch P and Gailly J-L 1996 ZLIB Compressed Data Format Specification Version 3.3 *Internet Engineering Task Force* RFC1950
[12] Myers J 1995 The Content-MD5 Header Field *Internet Engineering Task Force* RFC1864
[13] Mogul J 2002 Instance Digests in HTTP *Internet Engineering Task Force* RFC3230
[14] Cameron D and Wu W 2017 LHC@home `http://lhcathome.web.cern.ch`