Federating Distributed Storage For Clouds In ATLAS

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

Frank Berghaus, G07810, CERN, CH-1211 Geneva 23, Switzerland

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] software solution developed by CERN IT. DynaFed supports several industry standards for connection protocols like Amazon's S3, Microsofts Azure, as well as WebDAV and HTTP. Protocol dependent authentication is hidden from the user by using their 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 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 clouds, such as Amazon's Web Services, or the Compute Canada, and CERN OpenStack. These resources are integrated into the ATLAS distributed computing system using two cloud scheduler [2] instances: one at the University of Victoria and one at CERN. 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.

2. Conceptual Design

The ATLAS experiment leverages the resources of the Worldwide LHC Computing Grid, WLCG [3]. The computer centres that are part of the WLCG and support that ATLAS experiment each host some of the experiment data and simulated events. They provide a global storage infrastructure. While the central ATLAS computing infrastructure uses purpose specific protocols to access the content of these grid storage elements, they may be accessed using standard protocols such as WebDAV, HTTP, and NFS. Figure 1 shows how DynaFed
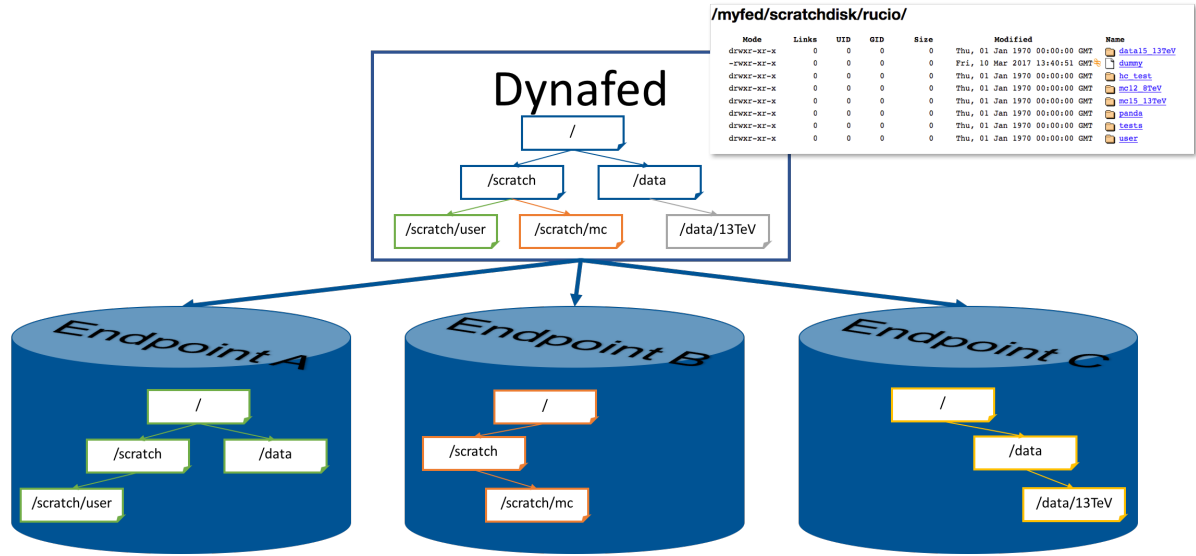
Figure 1. The dynamic federation is cpnnected to multiple endpoints. Each endpoint may be a file system or an object store accessible using the a protocol which allows redirection. The dynamic federation appears to provide a namespace that is a union of all the namespaces of the endpoints. That namespace is presented as a familiar directory structure on the same protocols as exposed by the endpoints.

could appear to present the entire ATLAS data catalog by unifying the namespaces of attached storage elements.

Cloud storage systems are object stores that expose a well defined interface over HTTP and WebDAV. DynaFed also allows the inclusion of these cloud storage into the a grid system. DynaFed implements authentication through public key infrastructure with grid extensions [4] and translates this authentication to authorize access to the cloud storage systems which each have their own authentication and authorization paradigms. Thus a grid user or application authenticates to DynaFed using grid credentials and is forwarded to a pre-signed URL that permits, for a limited time, access to the cloud storage system.

3. Data Access
The DynaFed instance regularly polls all connected endpoints to determine if they are reachable. Should an endpoint become unresponsive no requests will be forwarded to it until it responds again. This dynamically adjusts for storage endpoint failures and should increase the stability of the storage system as a whole.

Figure 2 illustrated the task division in the dynamic federation handle client requests. When the dynamic federation receives a requests for a while in it's namespace the client is authenticated. Once the request has been authenticated the name is looked up in the cached. If a cached entry exists that entry is returned. If the cache does not contain an entry for the queried file the name of the file is translated to each of the endpoints and the endpoints are queried. The federation waits for a response from all endpoints up to some timeout. The response is cached and the client is redirected to the geographically closest copy of the data. 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 area2c copy tool. Should the client make a request to write dynafed redirects the client to the geographically closest writeable storage element.
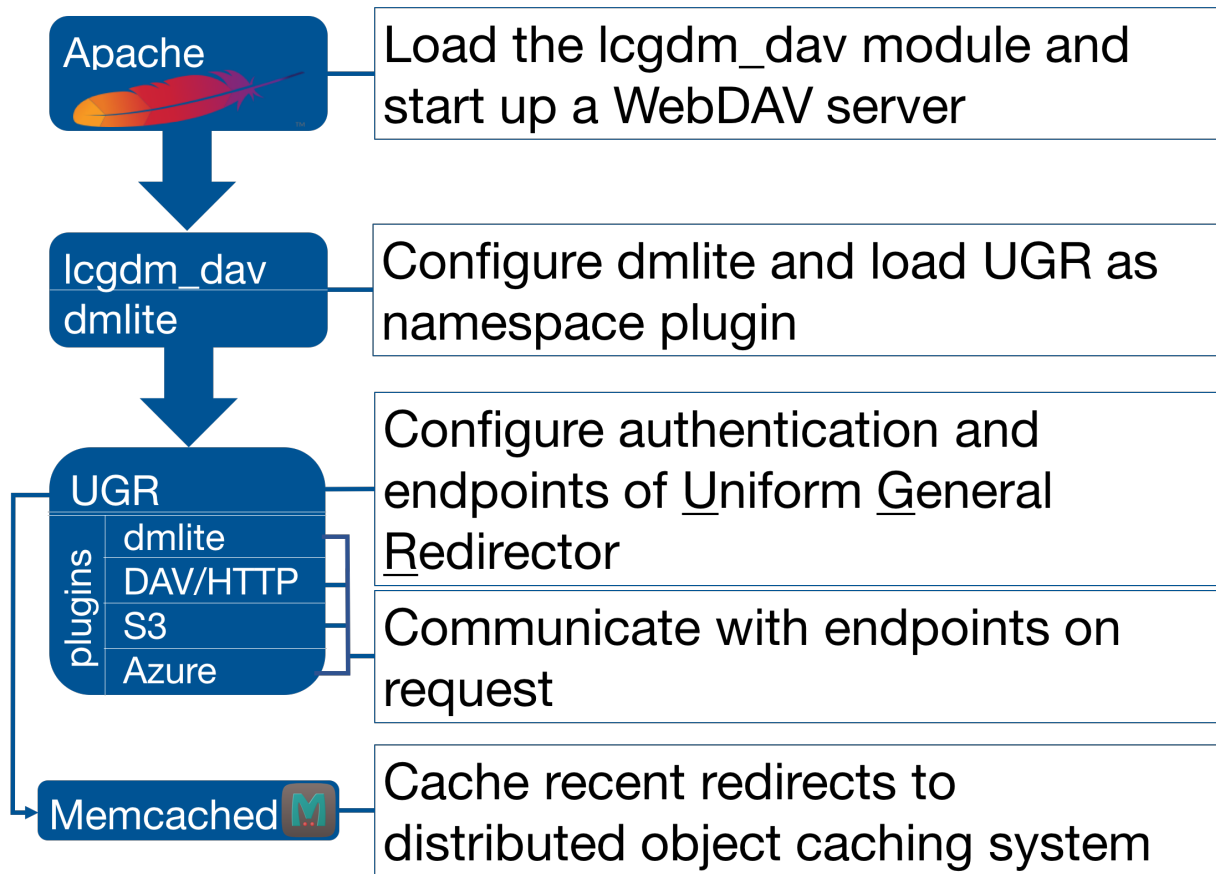
Figure 2. The dynamic web federation is an apache we server running the LCGDM implementation of WebDAV. The namespace generally managed by LCGDM has been replaced by the uniform general redirector which translates the requests to the web file system to the connected endpoints. The endpoint module handle the communicate with the connected 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.

4. Application Workflow

The goal of this project is to integrate the dynamic federation into the ATLAS and Belle-II distributed computing and data management systems. Here we will focus on the ATLAS system since the integration into PanDA [5] and Rucio is more mature.

Currently ATLAS distributed computing takes advantage of volunteer resources [6] and opportunistic cloud resources. The cloud resources are used as the testbed for this work, but the results will be useful for both and hopefully more. Cloud resources are included into ATLAS distributed computing using the cloud scheduler technology as illustrated in figure 3. A cloud scheduler requests resources on a cloud in response to load on a batch queue. The pilot which consumes the ATLAS workloads from PanDA is configured to connect to DynaFed for input and output data.

5. Summary
Acknowledgments
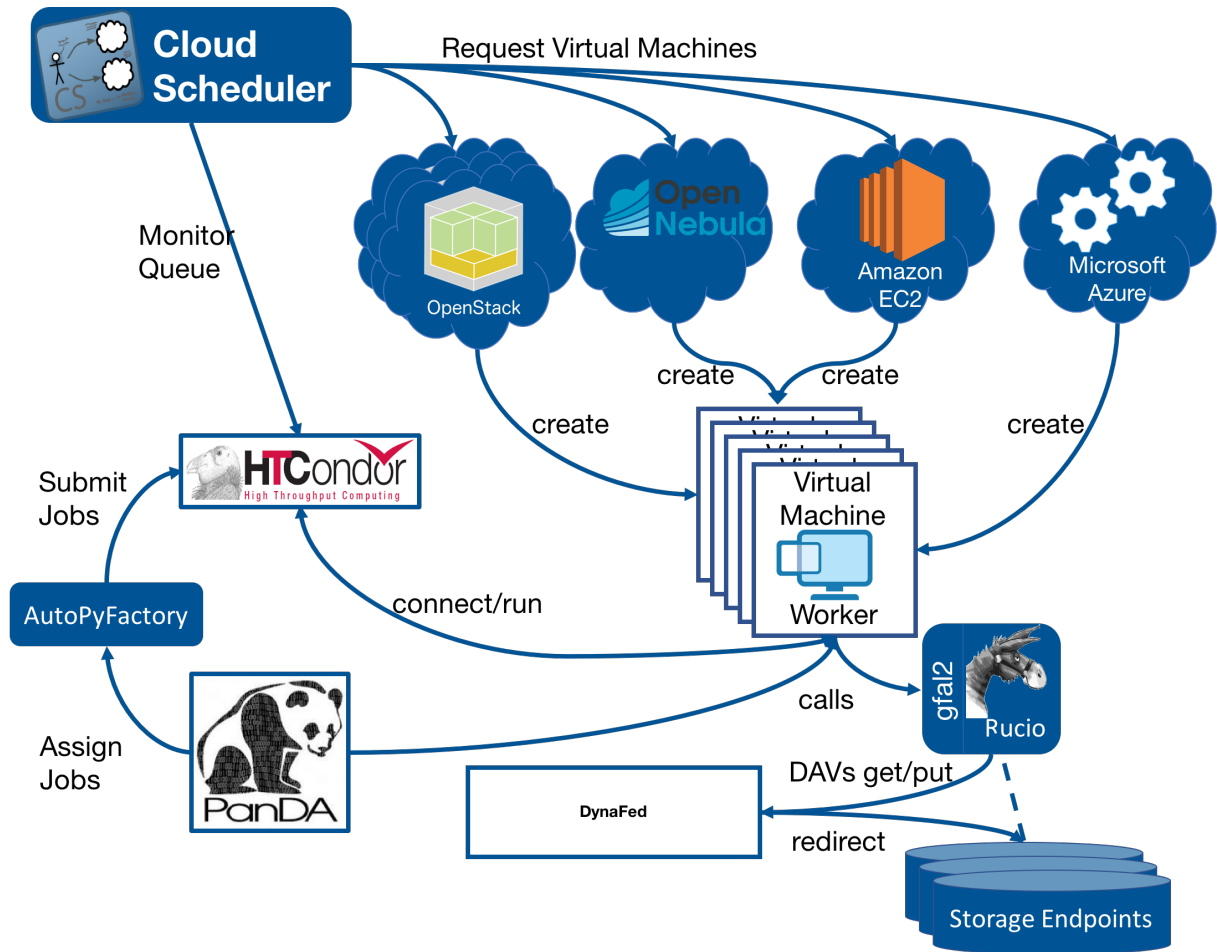Should I put people here instead of the long list of authors.

Figure 3. A client (AutoPyFactory or Harvester) submit 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 load on the queue. 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 the condor 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 via WebDAV. The federation forwards Rucio to the closest available storage.

Where does our grant funding come from
ATLAS DDM, ADC, CERN IT/DPM, who else?

References
[1] Furano F *et al* 2017 Dynafed `http://cern.ch/lcgdm/dynafed-dynamic-federation-project`
[2] Gable I *et al* 2017 Cloud Scheduler `http://cloudscheduler.org`
[3] Bird I 2011 *Ann. Rev. Nucl. Part. Sci.* **61** 99
[4] Foster I, Kesselman C, Tuecke S 2001 *International Journal of Supercomputer Applications* `http://www.globus.org/alliance/publications/papers/anatomy.pdf`
[5] T.Maeno et al 2017 PanDA `http://www.pandawms.org`
[6] Cameron D, Wu W 2017 LHC@home `http://lhcathome.web.cern.ch`