EPJ Web of Conferences **214**, 04026 (2019) https://doi.org/10.1051/epjconf/201921404026
CHEP 2018

Using a dynamic data federation for running Belle-II simulation applications in a distributed cloud environment

Marcus Ebert[1,*], *Frank* Berghaus[1], *Kevin* Casteels[1], *Colson* Driemel[1], *Colin* Leavett-Brown[1], *Fernando* Fernandez Galindo[2], *Michael* Paterson[1], *Rolf* Seuster[1], *Randall* Sobie[1], *Reda* Tafirout[2], and *Ryan* Taylor[1]

[1]University of Victoria, 3800 Finnerty Road, Victoria, BC, V8P 5C2, Canada
[2]TRIUMF, 4004 Wesbrook Mall, Vancouver, BC, V6T 2A3,Canada

Abstract. The dynamic data federation software Dynafed, developed by CERN IT, provides a federated storage cluster on demand using the HTTP protocol with WebDAV extensions. Traditional storage sites which support an experiment can be added to Dynafed without requiring any changes to the site. Dynafed also supports direct access to cloud storage such as S3 and Azure. We report on the usage of Dynafed to support Belle-II production jobs running on a distributed cloud system utilizing clouds across North America. Cloudscheduler, developed by the University of Victoria HEP Research Computing group , federates Openstack, OpenNebula, Amazon, Google, and Microsoft cloud compute resources and provides them as a unified Grid site which on average runs about 3500 Belle-II production jobs in parallel. The input data for those jobs is accessible through a single endpoint, our Dynafed instance. This Dynafed instance unifies storage resources provided by Amazon S3, Ceph, and Minio object stores as endpoints, as well as storage provided by traditional DPM and dCache sites. We report on our long term experience with this setup, the implementation of a grid-mapfile based X509 authentication/authorization for Belle-II access, and we show how a federated cluster can be used by Belle-II through gfalFS.

1 Introduction

Traditionally, a Grid site is a unit of compute and storage systems located at a specific location. Compute jobs, which analyze a specific set of data, are often sent to the site that has that data on their storage servers. Back in 2002 when the WLCG was formed, this was needed to increase the reliability of the data access and to reduce the WAN bandwidth needed for data transfers. The compute resources are usually behind a local batch system. Jobs get submitted to the batch system head node and then distributed to worker nodes with free resources.

Today the situation is different. Due to the increased need for compute resources, smaller sites with only very limited storage resources are used. Such small sites may not even have the manpower to maintain an own Grid specific storage system. It would be helpful for such sites if widely used industry standards for storage systems could be used instead of complex WLCG specific storage systems.

*e-mail: mebert@uvic.ca

In addition, the usage of cloud computing increased over the last years to satisfy the increased need for computing resources. Using clouds is an easy way to get access to more computing power and to add it to an experiment's computing resources. Such compute clouds can be research clouds located at a university or computing center, or commercial clouds that can be located anywhere in the world. At the University of Victoria (UVic), we use a system called cloudscheduler [1] to integrate the distributed cloud resources as worker nodes of a HTCondor batch system [2].

This distribution of resources has a number of challenges. Since the Grid system assumes that compute and storage are a unit, it will send the compute jobs to a site with the data. However, now the worker nodes can be located at any cloud that is integrated into the local batch system. Only the batch system head node is located at the site. The storage system however is still located at the site too. This setup can create at least three problems:

1. The storage server can get overloaded due to the increased number of compute jobs accessing a single storage site.

2. The data transfer to/from a worker node can take a long time and is inefficient when the worker node is far away from the storage endpoint.

3. WAN traffic is not as reliable as LAN traffic, due to it being routed through a number of connection points that are outside of the site's control.

Figure 1 shows the general idea of a Grid site and the data access problem when distributed compute resources are used.

To overcome this problem, we use Dynafed, a dynamic data federation software [3], and studied its application for data access in the above described distributed computing environment. The idea behind using Dynafed is to reduce the storage associated with a site to a simple redirector endpoint while the real storage is located elsewhere, similar to the compute where only the head node of the batch system is located at our site and the real compute worker nodes were moved elsewhere. In this model, a traditional Grid site becomes a lightweight site without real storage and compute resources on site, as shown in Figure 2.

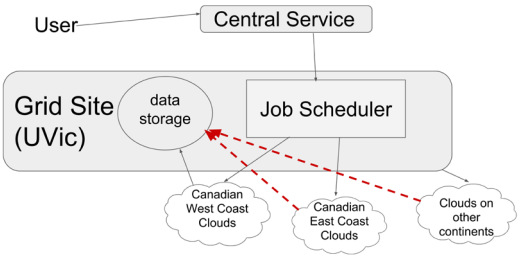

Figure 1: Grid site with distributed compute and traditional local storage, examples for inefficient data access are shown in red

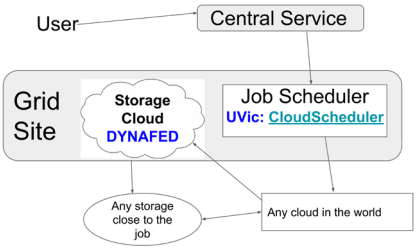

Figure 2: Grid site with distributed compute and storage endpoints located anywhere

2 Dynafed usage for Belle-II production at the University of Victoria

To use Dynafed as a site storage system, the clients on the worker nodes need to access data using the HTTP protocol. Unfortunately, this is not the case for the Belle-II clients, currently. The Belle-II framework only supports remote data access through the SRM interface using

gridftp. However, it supports access to the data when it is accessible within the filesystem of the worker nodes. This allows us to use Dynafed for Belle-II's data access. We use gfalFS [4] to make the data accessible through Dynafed from within the local file system on the worker nodes. With gfalFS, we can mount storage that is accessible through one of the protocols supported by the gfal2 tools. The gfal2 tools support the HTTP protocol and allow us to use our Dynafed instance as the endpoint of the mount. That way, the whole file structure behind Dynafed is accessible through any local file tools and can be accessed in the same way like any other local file, from a client point of view. However, all the advantages of Dynafed still apply. Using gfalFS, the clients will still be redirected to the storage endpoint that is closest to the client. Dynafed also allows the authentication and authorization based on VOMS X.509 proxies and certificates, which we have implemented.

2.1 Using gfalFS to make storage behind Dynafed accessible for Belle-II production jobs

To use gfalFS on the worker nodes, the gfal2 tools and libraries need to be installed. On our worker nodes, which are all Virtual Machines (VMs) on clouds, we use the latest CERNVM 3 image which already has the required software installed [5]. Since we run multi-core VMs while the jobs only use a single core, each job runs as a different local user for better monitoring possibilities. Mounting remote storage using gfalFS is done through fuse in the user space requiring a separate mount for each user. We create the mounts through a script in `/etc/profile.d/` which is executed each time a new job starts as a user process. The steps needed to make the remote data locally available for Belle-II jobs are:

- create a directory where the data will be mounted: `mkdir -p $HOME/b2data/belle`
- set the VOMS proxy variable to be used with gfalFS:
  `X509_USER_PROXY="$workdir/$diracdir/proxy"; export X509_USER_PROXY`
- mount Dynafed using gfalFS:
  `gfalFS -s $HOME/b2data/belle davs://dynafed02.heprc.uvic.ca:8443/belle`
- make local mount point known to the Belle-II framework:
  `export VO_BELLE_DATA_DIR=$HOME/b2data`

The VOMS X.509 proxy is transferred to the worker node together with the job itself and located in the job's work directory. Through the variable `VO_BELLE_DATA_DIR`, jobs know where to find the data in the local file system structure.

2.2 Authentication and authorization used with Dynafed

Dynafed can use external Python scripts to handle the authentication and authorization for data access. We use such a script to identify users based on the VOMS information retrieved from the Belle-II VOMS server. We regularly query the VOMS server and save the user identification together with their role in `/etc/grid-security/grid-mapfile`. If users have more than one role associated with their certificate, then all roles are saved.

When our Dynafed instance is accessed, the received identification string is searched for in the grid-mapfile to see which role a user has within the experiment. The role is then used to authorize or deny specific access to the requested data. The authorization for different parts of the file structure behind our Dynafed instance is defined in an extra file. To allow every member of Belle-II to list the directory structure and to read the file content, the following line is added in this file: `"/belle/MC belle rl"`, where `/belle/MC` is the area that is authorized to be accessed by all Belle-II members (`belle`) for read and list access (`rl`). More information about it can be found in [6].

2.3 Data access and data distribution

At the time when we put our Dynafed instance into production, all Belle-II simulation jobs required some files out of a pool of 85GB. To account for the worker nodes running on clouds in different regions, we created storage endpoints at the different locations. On clouds where no other storage is close by and no native object store is available, we run an extra VM with a large disk area and use Minio [7] to provide an S3 API to access data stored on this VM. The input data file set was manually copied onto the storage endpoints on each cloud and each endpoint was included in our Dynafed configuration. The additional endpoints are also reducing the load on the main Grid site storage element. With 3000 parallel jobs running, we had daily file transfers of about 30TB, which cannot be handled in an efficient way by a single storage endpoint over WAN.

While this worked very well for a small number of input data files, for larger input data sets we have to find a way to distribute files automatically. This is currently in development and the monitoring of our Dynafed instance can help to provide the names of most accessed files together with the client location from where it was accessed the most. If this leads to a large number of file accesses at a single endpoint, the files can be copied over to another endpoint closer to the compute that needs those files.

We have run the described setup using Dynafed and gfalFS successful in production since Summer 2017. It helps to avoid overloading a single site storage system and it makes sure all jobs can access their input data in an efficient way.

2.4 Monitoring

To measure how our Dynafed instance performs over time, an accounting and monitoring system is being developed in collaboration with TRIUMF's Atlas Group. It utilizes the technologies provided by Elastic Stack [8] to send log data (Filebeat), process and enrich the data (Logstash), and to store it in a database (Elasticsearch). Grafana [9] is then used to visualize and monitor this data using custom made dashboards [10].

This allows us to monitor the current state of all endpoints, where requests were made from, which files were requested with popularity statistics, and which endpoints served the requests, as well as the free and used storage capacity of the endpoints. In addition, the above information is available as timelines too.

3 Testing of our Dynafed setup

To test the efficiency of different ways to access the data, we setup the following testing environments:

- use only Grid sites or only object stores behind Dynafed
  - Grid sites: UVic, BNL
  - Object stores: CEPH at UVic, Minio on VMs with external volumes mounted at Compute Canada's (CC) WestCloud and EastCloud, Minio at our group's cloud (UVic cloud) with large local disk
- on worker nodes at different clouds, 500 3GB files were copied to `/dev/shm`
  - clouds with test worker nodes: CC WestCloud and EastCloud [11], UVic cloud

For comparison, different protocols and storage access methods were used:

- direct SRM/gridftp access to UVic's site storage element
- direct HTTP access to UVic's site storage element
- data access through Dynafed with only Grid sites accessible
- data access through Dynafed with only object stores accessible

The results of the tests are shown in the following subsections.

3.1 Direct SRM/gridftp access to UVic's site SE

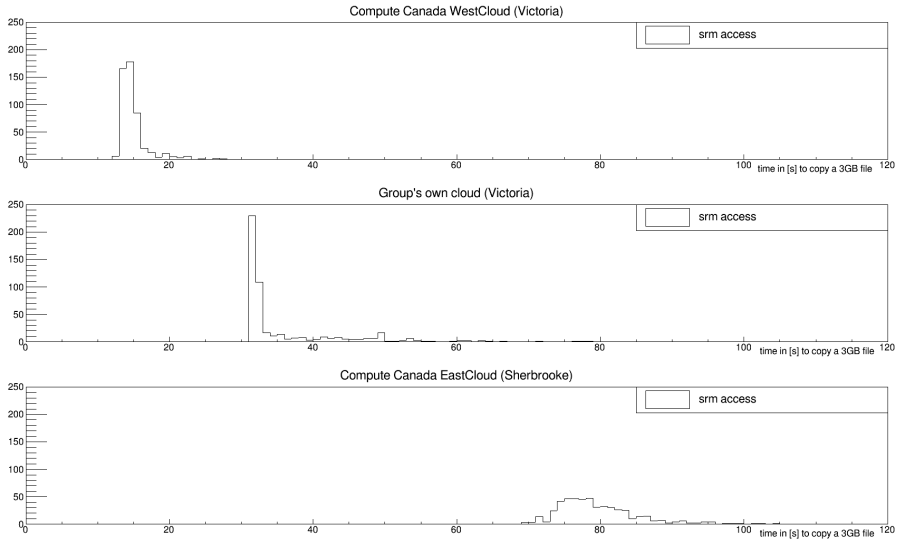

Figure 3: Time distribution for copying 500 3GB files directly from the UVic Grid storage, initiated by clients on different clouds using the SRM/gridftp protocol.

In this test, the SRM/gridftp protocol was used to directly access UVic's site storage element from all three clouds. The CC WestCloud in Victoria has a special connectivity where the VLAN that the VMs are using for their network access is directly routed to the storage element which is also hosted by CC in Victoria. This can be seen in Fig.3 where the peak for the transfer time of a file is at much lower values than for the UVic cloud, which is also in Victoria. Different to the WestCloud, the network traffic of the VMs running on the UVic cloud is routed through the Openstack networking and from there to the storage. This approximately doubles the transfer time for a single file compared to WestCloud. The transfer times to EastCloud, which is located nearly 4000km away from Victoria, is much higher than the transfer rates on the clouds in Victoria which is expected due to the distance and the additional WAN routing.

3.2 Direct HTTP access to UVic's site SE

In this test the same was done like in the previous one, but instead of using the SRM/gridftp protocol the HTTP protocol was used to access the site storage. This is shown in Fig.4. One

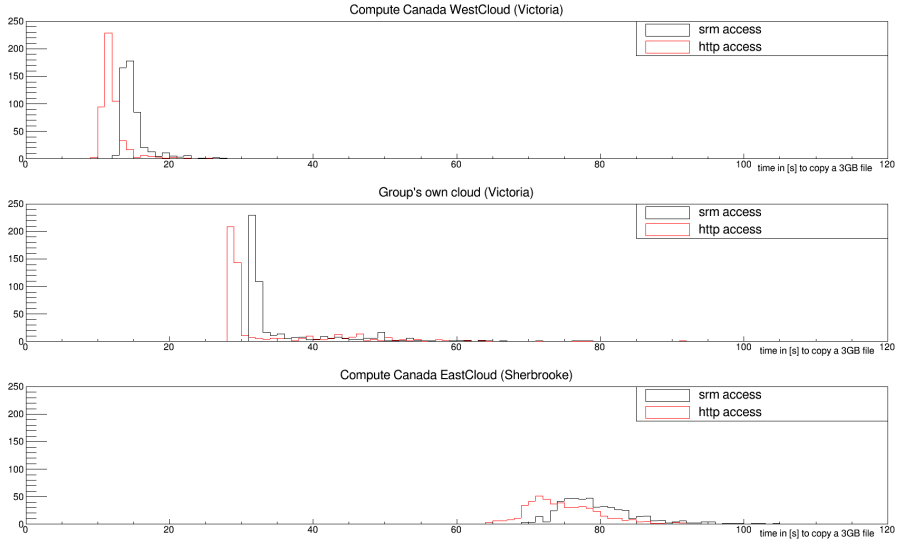

Figure 4: Time distribution for copying 500 3GB files directly from the UVic Grid storage, initiated by clients on different clouds using the HTTP protocol (in red); in addition to previous test distributions.

can see exactly the same differences in the transfer time between all three clouds like in the previous test. However, the transfer times using the HTTP protocol are consistently lower than the transfer times when using the SRM/gridftp protocol. This is consistent across all 3 clouds with their very different network access to the site storage.

3.3 Data access through Dynafed with only Grid sites accessible

For this test, we access the storage through Dynafed with only the two Grid sites, our site storage element and the BNL site storage, as storage endpoints setup behind Dynafed. For the two clouds in Victoria, Fig. 5 shows that there is basically no difference between the transfer times when accessing our site storage directly through HTTP and when it is accessed through Dynafed. However, on EastCloud one can see that the transfer times go down significantly when using Dynafed. The reason is that through Dynafed a file request gets redirected to the nearest storage, which is the BNL site storage for file requests from EastCloud.

3.4 Data access through Dynafed with only object stores accessible

For this test we access the files only from object stores through Dynafed. We use Minio test installations on VMs in each cloud to provide an S3 like API to the storage, as well as a Ceph storage cluster that runs in Victoria outside of any cloud. All VMs with Minio installations have a large second image as storage space available. On the UVic cloud this storage space is physically located on a local disk of the hypervisor that runs the VM. On the CC clouds, the image that provides the storage space is located on physical storage that is accessed over the network by the hypervisor.

This test shows that the transfer times increase by about a factor of two on WestCloud when accessing the files not on the Grid site storage element (Fig.6). The reason is that

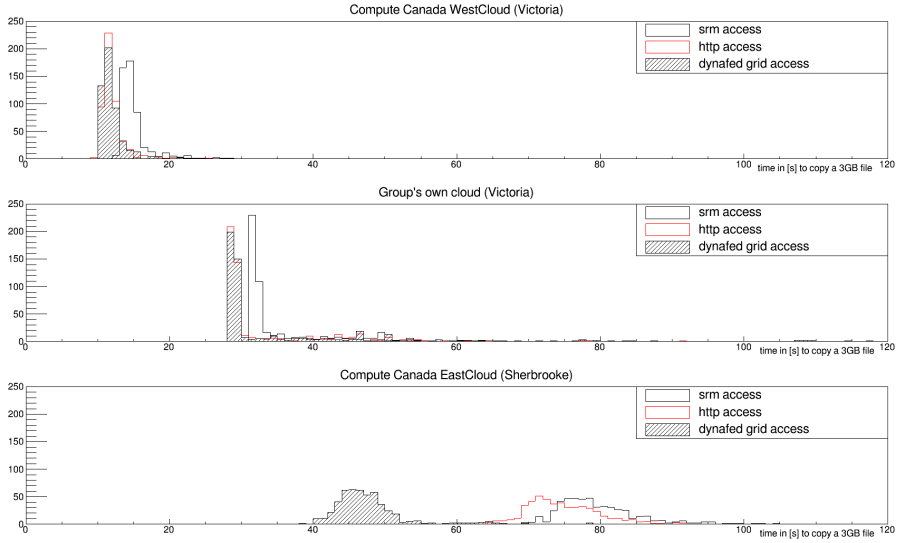

Figure 5: Time distribution for copying 500 3GB files with access through Dynafed using Grid sites as endpoints behind Dynafed (black striped histogram); initiated by clients on different clouds and in addition to previous test distributions.

the VMs on WestCloud have lost the benefit of direct access to the storage element that hosts the files. To access it on Ceph, it needs to go through the normal Openstack network routing to access anything on the outside. The access to files on the Minio instance on the cloud shows similar transfer times due to the additional network transfers involved. This results in similar transfer rates like on the UVic cloud when accessing files on the Grid site storage (Fig.5). This also shows that the transfer rates itself are similar for Grid storage and Minio installations when using a similar network access pattern. On EastCloud, the transfer times are reduced compared to using Grid storage behind Dynafed. This is expected since accessing local storage should always give better results. The reason that better transfer times are achieved on EastCloud than on WestCloud is most likely that on WestCloud many more VMs are running in parallel using the network and the network based storage. The best results in this test are achieved on the UVic cloud because here the files are located on an image that is local to the hypervisor and access does not involve any additional network traffic. In addition, since the VM on which Minio runs is within the Openstack installation, all file transfers happen within the local network avoiding the Openstack routing to access any outside networks.

4 Conclusion

In conclusion, we found that Dynafed is an ideal tool to access storage through a single endpoint when using distributed computing resources and we have used Dynafed and gfalFS for Belle-II grid jobs successful in production since Summer 2017. For best results, all possible storage endpoints an experiment can access should be used with Dynafed. As expected, storage local to the client shows the best performance while it does not matter for that purpose if it is a full Grid storage installation or a simple Minio data VM. The slowest transfer rates have been seen when accessing data on far away storage elements using WAN access. Since

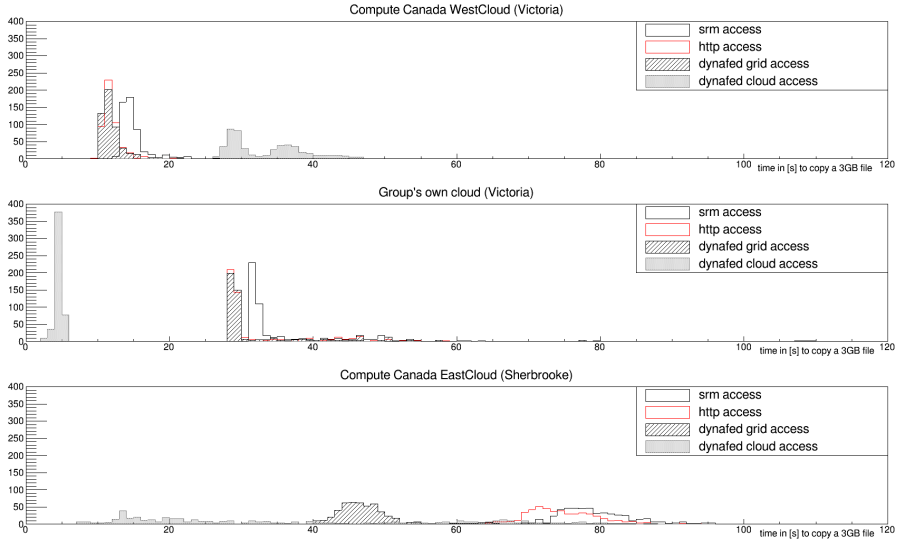

Figure 6: Time distribution for copying 500 3GB files with access through Dynafed using object stores as endpoints behind Dynafed (gray dotted histogram); initiated by clients on different clouds and in addition to previous test distributions.

Dynafed redirects to an endpoint that is nearest to the client, it is an ideal tool to avoid such inefficient long distance transfers. Using gfalFS makes it possible to use the advantages of Dynafed when only local file access is usable or preferred. We found that there is no change in performance when using Dynafed compared to direct HTTP access to a storage element. We also found that accessing files through the HTTP protocol shows better performance than the SRM/gridftp protocol.

We would like to thank Compute Canada and the Chameleon Project for giving us access to their compute clouds. In addition, this work was supported by the Canada Foundation for Innovation, the Natural Sciences and Engineering Research Council and the AWS Credits for Research Program. The assistance of the CERN Dynafed development team is greatly appreciated.

References

[1] R Seuster *et al.* J. Phys.: Conf. Ser. **898**, 052039 (2017)
[2] HTCondor project, "HTCondor" [software], https://research.cs.wisc.edu/htcondor/index.html
[3] F Furano *et al.*, http://lcgdm.web.cern.ch/dynafeds-text-documentation-white-paper
[4] CERN IT, "gfalFS" [software], version 1.5.2, https://dmc.web.cern.ch/projects/gfalfs
[5] J Blomer *et al.*, J. Phys.: Conf. Ser. **513**, 032009 (2014)
[6] https://heprc.blogspot.com/2017/06/grid-mapfile-based-authentication-for.html
[7] MINIO, "minio" [software], version 2017-04-29, https://www.minio.io/
[8] Elasticsearch B.V., "Elastic stack" [software], version 6, http://www.elastic.co
[9] Grafana Labs, "Grafana" [software], version 5, https://github.com/grafana/grafana
[10] https://atlas-fed-metrics.triumf.ca
[11] Compute Canada,
https://www.computecanada.ca/research-portal/national-services/compute-canada-cloud/