The Dynafed data federator as a grid site storage element

Marcus Ebert[1,*], *Frank* Berghaus[1], *Kevin* Casteels[1], *Colson* Driemel[1], *Oliver* Keeble[3], *Colin* Leavett-Brown[1], *Fernando* Fernandez Galindo[2], *Fabrizio* Furano[3], *Michael* Paterson[1], *Rolf* Seuster[1], *Randall* Sobie[1], and *Reda* Tafirout[2]

[1]University of Victoria, 3800 Finnerty Road, Victoria, BC, V8P 5C2, Canada
[2]TRIUMF, 4004 Wesbrook Mall, Vancouver, BC, V6T 2A3,Canada
[3]CERN, Esplanade des Particules 1, 1211 Geneva 23

Abstract. The Dynafed data federator is designed to present a dynamic and unified view of a distributed file repository. We describe our use of Dynafed to construct a production-ready WLCG storage element (SE) using existing grid storage endpoints as well as object storage. Dynafed is used as the primary SE for the Canadian distributed computing cloud systems for the Belle-II experiment, where we use it in production for reading input files. We have run up to 6,000 Belle-II jobs simultaneously on distributed cloud resources, requiring the transfer of approximately 60 TB of input data per day. Similarly, we have been using a Dynafed-based SE in pre-production testing for the ATLAS experiment. We will describe the configuration of Dynafed to make it suitable as a WLCG SE. In particular, we will highlight the improvements within Dynafed that make it possible to do checksum based file verification and 3rd-party file copy via WebDAV. We will also report on a new monitoring system and an automated system that collects storage information from all endpoints.

1 Introduction

The Dynafed data federator [1] gives an unified view of a distributed file repository in a dynamic way. Dynafed itself is a lightweight system which redirects file requests via HTTP to the storage endpoints which host those files. In case multiple endpoints host the same file, the request is then redirected to the one closest to the client that made the initial request. The workflow for requesting a file for reading through Dynafed is shown in Fig. 1.

Our main reason for using Dynafed is due to the way we operate the University of Victoria (UVic) grid site. Instead of using static bare-metal worker nodes, our worker nodes are VMs started dynamically on distributed clouds depending on the resource requirements of a job. The management of the VMs and cloud resources is done by Cloudscheduler [2]. Fig. 2 shows in a schematic way how Cloudscheduler works. We can not efficiently use a SE at a single site since we operate on clouds distributed over Northern America as well as Europe. A distributed solution is needed which allows to integrate different storage services located close to the clouds we are running on and which is accessible via a single SE endpoint name. Dynafed is such a solution we are exploring.

We operate three different Dynafed instances. The setup and usage will be discussed in the next section. Our main goal is to operate Dynafed as a production SE for the ATLAS and

*e-mail: mebert@uvic.ca

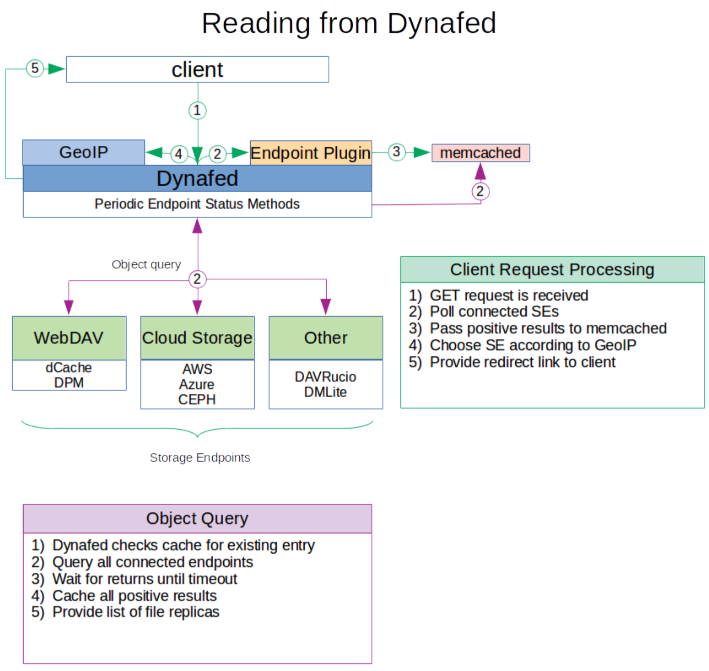

Reading from Dynafed
client
1
5
GeoIP
4
2
Endpoint Plugin
3
memcached
Dynafed
Periodic Endpoint Status Methods
2
Object query
2
WebDAV
dCache
DPM
Cloud Storage
AWS
Azure
CEPH
Other
DAVRucio
DMLite
Storage Endpoints
Client Request Processing
1) GET request is received
2) Poll connected SEs
3) Pass positive results to memcached
4) Choose SE according to GeoIP
5) Provide redirect link to client
Object Query
1) Dynafed checks cache for existing entry
2) Query all connected endpoints
3) Wait for returns until timeout
4) Cache all positive results
5) Provide list of file replicas

Figure 1: Flow chart of a read request send to Dynafed.

Belle-II experiments [3, 4]. Dynafed is also used by CERN for the LHC@home projects of the LHC experiments [5], by RAL for their ECHO cluster [6], and by INFN to allow for an unified view of the Belle-II user area on the Grid [7].

2 Dynafed instances for Belle-II and Atlas

This section describes the general setup of the Dynafed instances that we manage and use for Belle-II and Atlas.

2.1 UVic instance for Belle-II

Belle-II uses the DIRAC management system [8] where a site SE, UVic-SE in this case, can be configured using different protocols. Since most Belle-II storage sites are configured to use only the SRM access [9], all Belle-II storage sites need to support SRM access as well to have a common protocol for general data access to copy data between sites. For file access via SRM, DIRAC utilizes a dCache instance at UVic operated by Compute Canada. The storage space on this dCache instance is about 330TB currently. For the storage access via HTTP/webdav, DIRAC contacts our Belle-II Dynafed instance. The dCache instance is available as endpoint behind Dynafed as well as other grid sites in read-only mode.

In addition, we also operate small object storage systems as Dynafed endpoints. We use MinIO [10] to provide the object storage on single VM instances started on different clouds, each with storage space at the order of 100GB, as well as for a distributed multi-host cluster with a storage capacity of about 45TB. The most used data (in general files needed for most

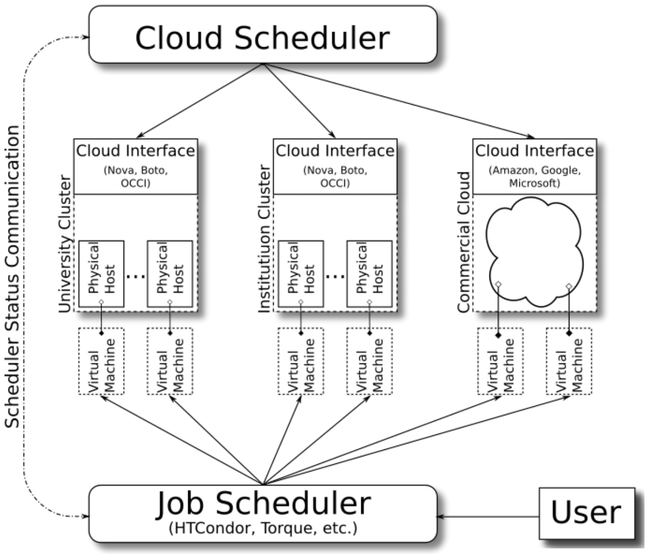


Figure 2: Cloudscheduler schema, Cloudscheduler [2] polls the job scheduler for job requirements and starts/stops VMs on attached clouds when needed depending on those requirements.

simulation jobs) is replicated manually to the different MinIO instances and then instantly available via Dynafed as an additional copy.

Our Belle-II Dynafed instance is used for reading files from storage, as well as for writing the job output to the storage. While reading input files via Dynafed is used successfully since 2017, writing through Dynafed was enabled in 2019 for first tests and was enabled permanently in 2020.

2.2 Dynafed instances for ATLAS

The Dynafed instances for ATLAS are setup to test the ATLAS data management system with plain object storage and no other grid interface available than HTTP.

The Dynafed instance at TRIUMF uses a single 30TB Ceph-based [11] endpoint for the data storage and is currently integrated as a Tier-3 site for Atlas. Instead of a single Dynafed installation, three Dynafed systems are operated in an identical configuration behind a HAProxy [12] for load balancing and to improve overall stability of the system. The instances share a common Memcached instance [13]. Such setup was also chosen to study scalability in the future.

The Dynafed instance at CERN also uses a Ceph-based endpoint behind Dynafed with 50TB usable space and is setup for production testing of the storage access. Such tests are ongoing but already improved the ATLAS data management system in handling object storage and using a federation of existing Grid sites for reading.

3 Setup and monitoring of the Dynafed instances

In this section we describe the setup of components needed for a Dynafed instance to be used as SE, like the support of checksum and third party copy requests. In additon, we also introduce a monitoring that we setup.

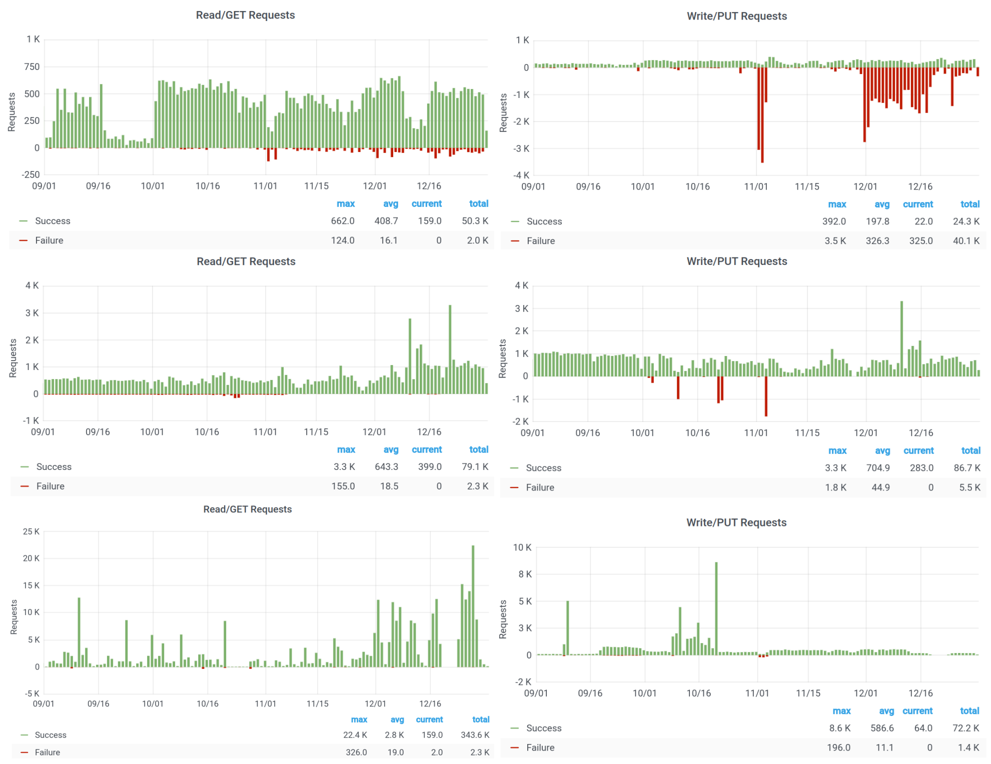


Figure 3: Read (left side plots) and write requests (right side plots) made between September and December 2019 through the CERN Dynafed instance (top), the TRIUMF Dynafed instance (middle), and the UVic Dynafed instance (bottom). The two instances for ATLAS show a more constant access pattern due to long running jobs requesting files, while the Belle-II instance shows large spikes on top of a constant access pattern due to large number of very short running jobs requesting files. Failed requests are mainly due to having a requested file not at any endpoint or clients having the wrong permission for requested file access.

3.1 Authentication and authorization

All three Dynafed instances support X.509 authentication with authorization based on VOMS [14] roles. A VOMS server for each supported experiment is queried to get a list of all members together with their certificate identification and role within the experiment. Within each Dynafed instance, access to different paths are authorized for specific requests, like read-only or read-write access requests, depending on the identified user and the role within the experiment. Multiple roles and experiment memberships are supported for a single user. More information about this setup can be found in [15].

3.2 Checksum support

The checksum algorithm supported by most grid experiments is ADLER32 [16]. Such ADLER32 checksum of a file can be queried through RFC3230 [17] using the Want-Digest header. Object storage on the other hand usually uses MD5 [18] and does not support *want-digest* requests.

When the checksum of a file is requested through Dynafed, then Dynafed redirects the checksum request to a grid storage system with native *want-digest* support and runs an external executable for storage endpoints that do not support the *want-digest* request. This external executable can download the file from such endpoint, and then calculate and return the checksum. In addition, it can also store the calculated checksum in the object's metadata on the endpoint and retrieve it from there the next time the checksum is requested, without the need to download the whole file.

This checksum calculation is performed directly on the UVic Dynafed instance, while the checksum is calculated directly on the Ceph Rados gateway for the TRIUMF installation and from there returned to Dynafed. The CERN instance of Dynafed uses load balanced external machines which Dynafed connects to via a web service running on those machines.

3.3 Third Party Copy (TPC)

Traditional grid storage sites support TPC requests which makes it possible to have a file copied directly between two sites without going through the client that requested the copy. The object storage implementations used however do not support TPC requests. To still allow TPC requests through Dynafed, the request is forwarded to the endpoints which support those and an external executable is called for endpoints which do not support TPC requests. This executable then handles the transfer of a file between two endpoints. Client's X.509 credentials used to initate a transfer can be delegated to the actual endpoints using gridsite delegation [19].

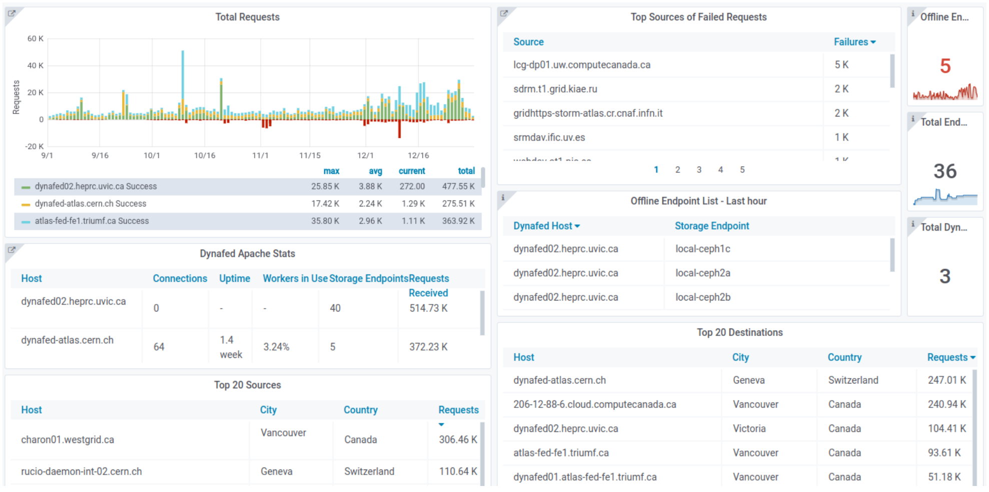
This external executable runs directly on the UVic Dynafed instance, and on the Ceph Rados gateway for the TRIUMF instance. At CERN, the executable runs via SSH on a set of load balanced servers to take the load off the Dynafed instance itself and to test scalability which is still ongoing.

3.4 Accounting and reporting

The storage space available through Dynafed is published via a JSON file that follows the WLCG Storage Resource Reporting format [20]. The experiment's data management systems can download this file through Dynafed. This JSON file can be generated automatically by calling the Dynafed storagestats utility [21]. While connecting to all endpoints to get their used and available space, the storagestats utility also adds the free space information into Memcached so Dynafed can use it to decide where to write files. When the data about free space on endpoints is available, Dynafed can redirect write requests only to endpoints who have sufficient free space available. In addition, the storagestats utility can also produce a list of files accessible through a Dynafed instance (content dump). Experiments usually manage an own database where they record which files are stored on which SE. Such content dump helps experiments to check if all data at an SE is accounted for correctly in their database.

3.5 Monitoring

We keep monitoring all activities on all three Dynafed instances based on the information in the Apache log file, customized for Dynafed. This information is available in near-realtime. For selected properties, such as the number of read or write requests, historic data is also available as shown in Fig. 3. To analyze the Dynafed information, we make use of different tools. On the server side we run Logstash [22] and Elasticsearch [23], and on the Dynafed instances Filebeat [24] and Execbeat [25]. Filebeat is used to parse the Apache log files for entries related to Dynafed and sends those information to Logstash. Execbeat reads the



Total Requests

	max	avg	current	total
dynafed02.heprc.uvic.ca Success	25.85 K	3.88 K	272.00	477.55 K
dynafed-atlas.cern.ch Success	17.42 K	2.24 K	1.29 K	275.51 K
atlas-fed-fe1.triumf.ca Success	35.80 K	2.96 K	1.11 K	363.92 K

Dynafed Apache Stats

Host	Connections	Uptime	Workers in Use	Storage Endpoints	Requests Received
dynafed02.heprc.uvic.ca	0	-	-	40	514.73 K
dynafed-atlas.cern.ch	64	1.4 week	3.24%	5	372.23 K

Top 20 Sources

Host	City	Country	Requests
charon01.westgrid.ca	Vancouver	Canada	306.46 K
rucio-daemon-int-02.cern.ch	Geneva	Switzerland	110.64 K

Top Sources of Failed Requests

Source	Failures
lcg-dp01.uw.computecanada.ca	5 K
sdrm.t1.grid.kiae.ru	2 K
gridhttps-storm-atlas.cr.cnaf.infn.it	2 K
srmdav.ific.uv.es	1 K

Offline Endpoint List - Last hour

Dynafed Host	Storage Endpoint
dynafed02.heprc.uvic.ca	local-ceph1c
dynafed02.heprc.uvic.ca	local-ceph2a
dynafed02.heprc.uvic.ca	local-ceph2b

Top 20 Destinations

Host	City	Country	Requests
dynafed-atlas.cern.ch	Geneva	Switzerland	247.01 K
206-12-88-6.cloud.computecanada.ca	Vancouver	Canada	240.94 K
dynafed02.heprc.uvic.ca	Victoria	Canada	104.41 K
atlas-fed-fe1.triumf.ca	Vancouver	Canada	93.61 K
dynafed01.atlas-fed-fe1.triumf.ca	Vancouver	Canada	51.18 K

Figure 4: Overview page for our Dynafed monitoring.

information in Memcached related to storage information on the endpoints and sends those also to Logstash. Logstash processes all the events by aggregating those with the same ID and adding DNS name and location information based on the geographic location of the IP addresses. Elasticsearch stores all the information and we use customized dashboards to display all information relevant to us. The information is:

- file information for each read/write/delete/copy request together with information about from where the request came and to which endpoint it was redirected
- available and used space for each endpoint
- most requested files
- top 20 source and destination server addresses including location

In addition, we also keep track of some Apache metrics like the uptime, number of connections, and how many worker processes were started. More detailed information can be found in [26] and an overview of the monitoring itself at [27]. Fig. 3 shows an example of a read/write request timeline we are able to extract from our monitoring. The ATLAS job management sends production jobs at a mostly constant level which can be seen in the timeline for both ATLAS queues. Belle-II did not have many production jobs during the time period shown, but users often submitted large number of analysis jobs. Those jobs run for a much shorter time than production jobs and appear as large spikes in the timeline plot for the UVic Dynafed instance. Failed requests for all Dynafed instances shown are mostly a result of requesting files that are not at one of the endpoints or due to requests made by clients which do not have the right permissions to access the requested file. The general overview page of the monitoring can be seen in Fig. 4.

4 Conclusion

We are able to use Dynafed successfully as a production SE within the WLCG for ATLAS and Belle-II. We operate three Dynafed instances at CERN, TRIUMF, and UVic for the ATLAS and Belle-II experiments, all with a different setup. Those different setups help us to learn about the behavior of Dynafed in different situations, for example how to handle large number of checksum requests or TPC requests. All instances support the usual authentication and authorization mechanism commonly used within the WLCG.

In addition, our Dynafed instances are able to handle all typical requests needed within the WLCG. Checksum and TPC requests are handled by external executables for endpoints which do not support it naively, on the Dynafed instance itself or via access to other machines. We also wrote tools which allow to report the storage behind Dynafed in a WLCG conformant way and make those storage information available to Dynafed for write decisions.

Acknowledgements We would like to thank the support of the Natural Sciences and Engineering Research Council of Canada and the Canadian Foundation of Innovation. In addition, the resources and generous support provided by Compute Canada, the Chameleon Project, the Edinburgh Compute and Data Facility (ECDF), and Amazon are acknowledged.

References

[1] F Furano *et al.*, http://lcgdm.web.cern.ch/dynafeds-text-documentation-white-paper
[2] B Berghaus *et al.*, "High-Throughput Cloud Computing with the Cloudscheduler VM Provisioning Service", Computing and Software for Big Science **4**, 4 (2020)
[3] https://atlas.cern/
[4] https://www.belle2.org/
[5] LHC@HOME, http://lhcathome.web.cern.ch/
[6] A Dewhurst *et al.*, "Using WLCG data management software to support other communities", https://indico.cern.ch/event/773049/contributions/3474406/
[7] A Doria *et al.*, "HTTPD federations and caching", https://agenda.infn.it/event/17957/contributions/83876/
[8] DIRAC Consortium, "DIRAC" [software], http://diracgrid.org/
[9] Storage Resource Management Working group, https://sdm.lbl.gov/srm-wg/index.html
[10] MINIO, "minio" [software], version 2017-04-29, https://www.minio.io/
[11] "Ceph" [software], https://ceph.io/
[12] "HAProxy" [software], http://www.haproxy.org/
[13] "Memcached" [software], https://memcached.org
[14] "Virtual Organization Membership Service", https://italiangrid.github.io/voms/
[15] UVic HEP-RC, https://heprc.blogspot.com/2017/06/grid-mapfile-based-authentication-for.html
[16] RFC-1950, "ZLIB Compressed Data Format Specification version 3.3"
[17] RFC-3230, "Instance Digests in HTTP"
[18] RFC-1321, "The MD5 Message-Digest Algorithm"
[19] http://gridsite.org/ and https://github.com/CESNET/gridsite
[20] https://twiki.cern.ch/twiki/bin/view/EGEE/WLCGISEvolution#Latest_docs
[21] UVic HEP-RC, "dynafed_storagestats" [software], https://github.com/hep-gc/dynafed_storagestats
[22] Elasticsearch B.V., "Logstash" [software], https://www.elastic.co/logstash
[23] Elasticsearch B.V., "Elasticsearch" [software], https://www.elastic.co/elasticsearch
[24] Elasticsearch B.V., "Filebeat" [software], https://www.elastic.co/beats/filebeat
[25] Ch Galsterer, "Execbeat" [software], https://github.com/christiangalsterer/execbeat
[26] UVic HEP-RC, https://heprc.blogspot.com/2019/06/monitoring-dynafed-with-elk.html
[27] UVic HEP-RC, https://elk2.heprc.uvic.ca/d/000000128/dynafed-overview