Dynafed data federator as site SE

Marcus Ebert
mebert@uvic.ca
on behalf of the HEP-RC UVic group:
Frank Berghaus, Kevin Casteels, Colson Driemel, Colin Leavett-Brown, Michael Paterson, Rolf Seuster, Randall Sobie (University of Victoria)
Fernando Fernandez Galindo, Reda Tafirout (TRIUMF)

Why do we want (need) Dynafed

UVic Grid Site:

- run jobs for Atlas and Belle-II
- Cloud computing using distributed cloud systems
  - private and public/commercial clouds
    - Openstack, OpenNebula, Amazon, Microsoft, Google
- Clouds in production are distributed throughout Northern America and Europe currently
- Possibility to integrate other clouds anywhere
  - used resources on an Australian cloud before
- CE: HTCondor + CloudScheduler
  - more about cloudscheduler in Rolf's talk: https://indico.cern.ch/event/765497/contributions/3348868/
- SE: so far traditional dCache site
  - local cluster for distributed compute....

Multi-cloud compute at UVic

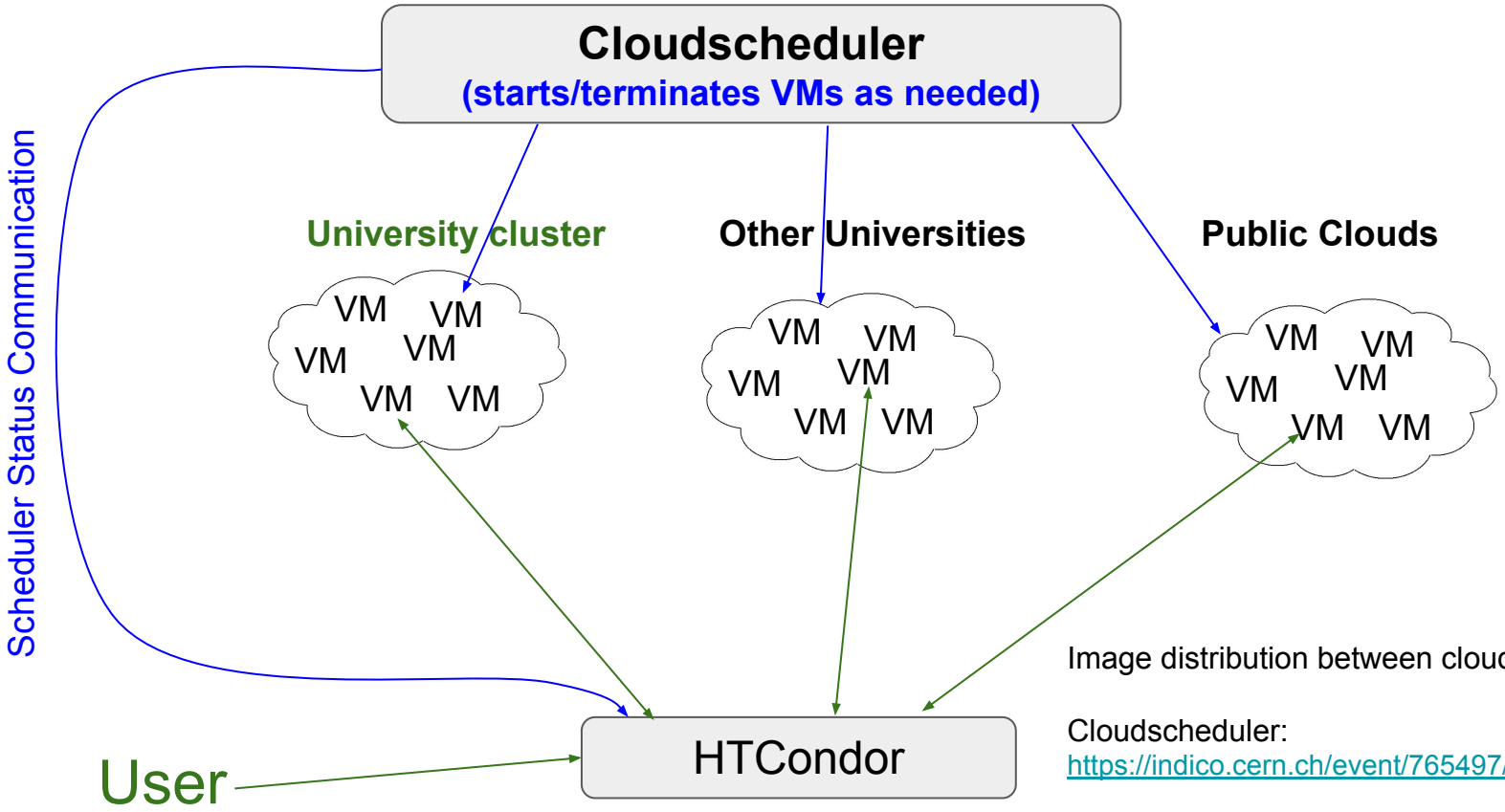
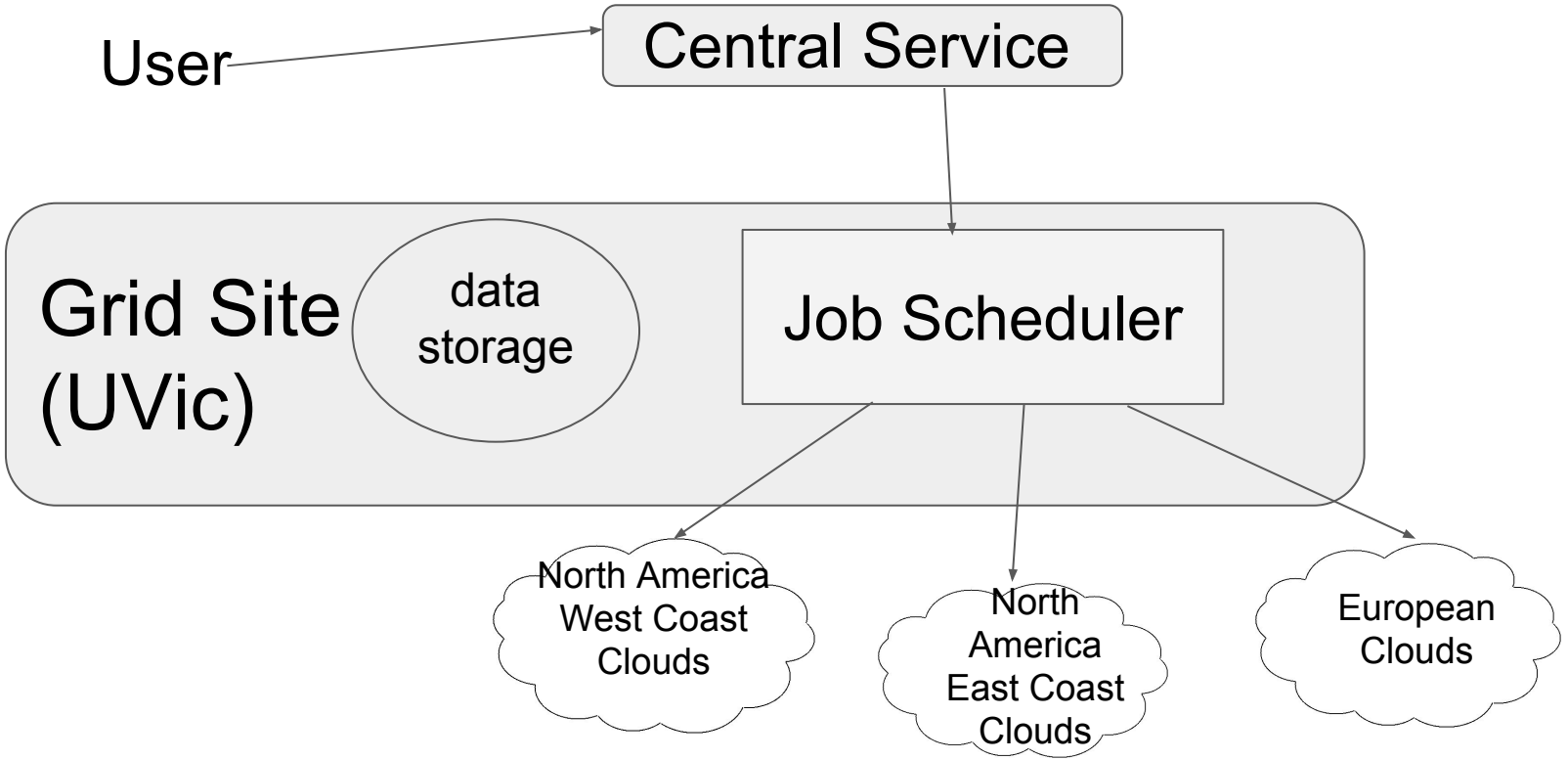


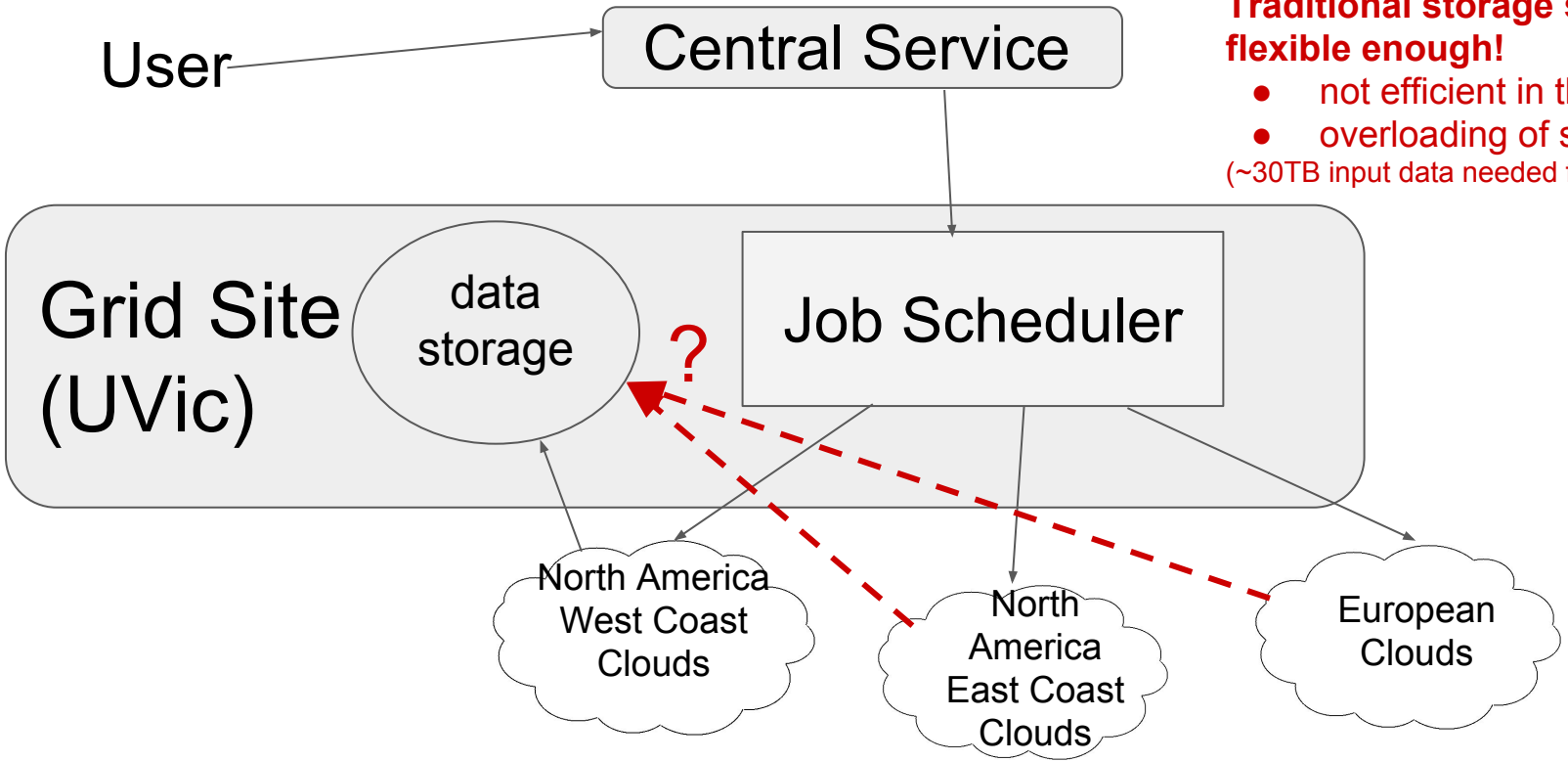
Image distribution between clouds: Glint

Cloudscheduler:
https://indico.cern.ch/event/765497/contributions/3348868/

Distributed cloud computing for the GRID



Distributed cloud computing for the GRID



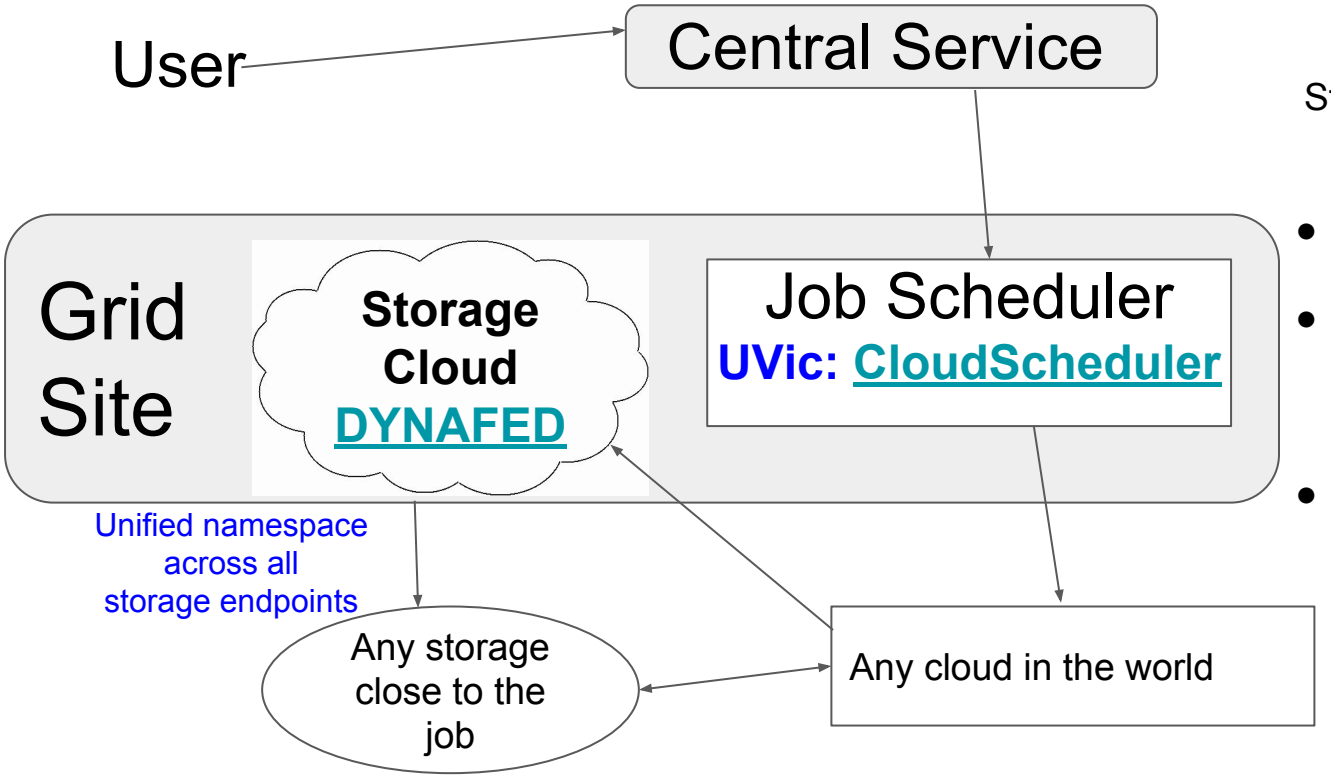
Problem:

Traditional storage systems are not flexible enough!

- not efficient in that situation
- overloading of single SE

(~30TB input data needed for Belle-II jobs daily)

Distributed cloud-storage and cloud-compute for the GRID



Storage cloud not only useful for cloud jobs but also for:

- compute-only sites,
- any group that needs distributed storage with unified namespace,
- and for adding storage to the GRID without using/knowing/needing Grid specific storage systems!

What is Dynafed

- redirector for a dynamic data federation, developed by CERN-IT (fabrizio.furano@cern.ch)
  - for data transfers, client is redirected to a storage element with the data
  - this can be done depending on geographic location (GeoIP)
    - storage elements closer to the job are preferred

- access through http(s)/webdav(s)

- can federate existing sites without configuration changes at sites
  - storage needs to be accessible through http(s)/dav(s)
  - world wide distributed data can be made accessible under common name space and through a single endpoint

- can also **directly access S3/Ceph and Azure based storage**
  - no credentials visible to the client
  - preauthorized URL with limited lifetime is used to access files on the storage

- X509/VOMS based authentication/access authorization can be used with dynafed
  - **http://HEPrc.blogspot.com** for grid-mapfile based authentication/authorization
    - different posts have also links to dynafed installation instructions in our TWiki

Dynafed@Victoria HEPRC group

- different installations at CERN, TRIUMF, and UVic
  - CERN: Atlas testing
  - Victoria: Belle-II production (r/o) and Atlas testing
  - TRIUMF: testing for monitoring and storage reporting development

- different storage solutions behind dynafed (for the instance at UVic):
  - existing GRID sites: our dCache and other Belle-II sites
  - CEPH storage: 50TB in our group CEPH, some TB at CERN
  - Amazon S3 (some hundreds of GB up to some tens of TB)
  - Minio in VMs on different clouds (some hundred GB)
    - https://www.minio.io/ , used for testing

Our Dynafed usage for Belle-II

- Belle-II does not support gfal2 so far
  - only srm access for remote storage access possible
  - in addition, can also use data in the local file system (!)
    - unfortunately only for read access, not write access

- use gfalFS to mount everything behind Dynafed in the local file system tree
  - fuse mount

```
gfalFS -s ${HOME}/b2data/belle davs://dynafed02.heprc.uvic.ca:8443/belle
```

- used successful in production since end of 2017

- gfal2 and http support will come to Belle-II framework very soon
  - in testing now
  - will enable dynafed usage for writing and removes the need of gfalFS

Testing gfalFS/dynafed

CHEP2018: https://indico.cern.ch/event/587955/contributions/2936834/

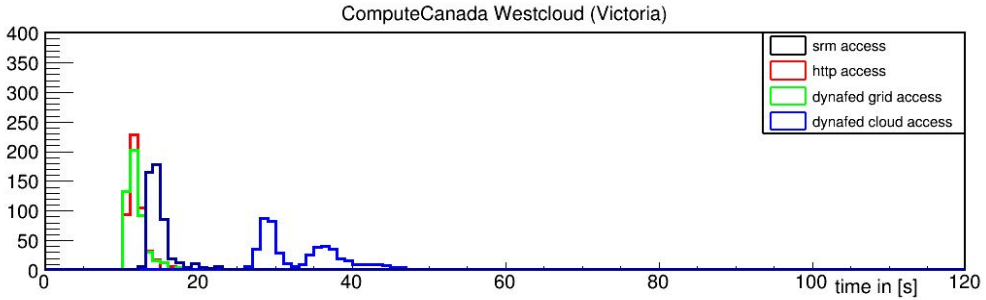
Victoria - Sherbrooke: 3853km

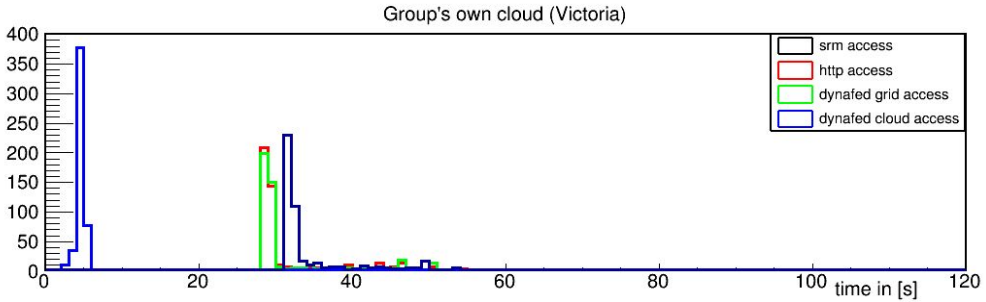
Sherbrooke - BNL: 515km

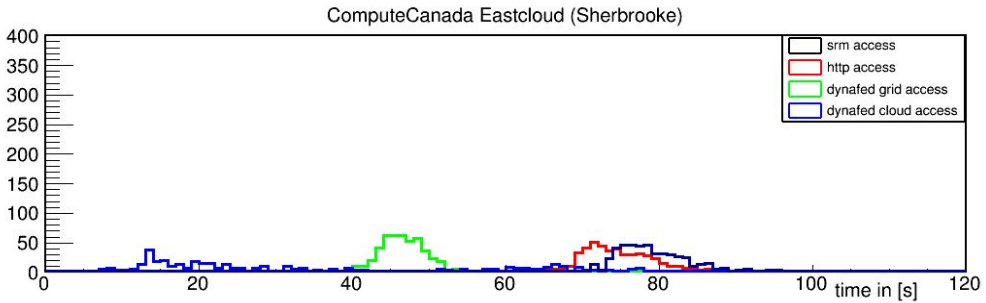
Victoria : 1xminio, 1xown Ceph, 1 Belle-II SE, 1x shared Openstack, 1x own Openstack
Sherbrooke : 1x minio, 1x shared Openstack
BNL : 1x Belle-II SE

Data access on clouds

CHEP2018: https://indico.cern.ch/event/587955/contributions/2936834/







- black: access to site SE (Victoria) using srm
- red: access to site SE (Victoria) using http
- green: access to site SEs using dynafed
- blue: access to object stores using dynafed (minio or Ceph)

- 500 3GB files copied to a worker node VM
  - copied to /dev/shm to not rely on virtual disk access
- minio “storage disk” is on network storage for Eastcloud, on local disk for our own cloud
- on Westcloud, minio and Ceph are accessible with minio on our own cloud
- Westcloud VMs have direct access to the site SE (don’t need to go through Openstack routing)

Our Dynafed usage for Atlas

- in testing currently
- much progress in Rucio to support Dynafed
- functional tests work
  - redirection
  - reading
  - writing
  - deleting
- some things are missing for production
  - checksum support
  - third party copy
  - space awareness, reporting and accounting

Our Dynafed usage for Atlas

- in testing currently
- much progress in Rucio to support Dynafed
- functional tests work
  - redirection
  - reading
  - writing
  - deleting
- some things are missing for production
  - checksum support
  - third party copy
  - space awareness, reporting and accounting

Problems Atlas/Rucio faces will also be problems for Belle-II and need to be solved before Dynafed can replace traditional Grid SEs.

Dynafed development

checksum:

- problem is difference in how checksums are handled on the Grid and on object stores
- Grid:
  - client calculates checksum and transfers file
  - client ask storage to get the checksum
  - client compares checksums and makes decision about file (ok or delete)
  - experiments decided to use adler32

- object stores:
  - calculate checksum and transfer file together with the checksum to the storage
  - storage system saves file and keeps it if local checksum and transferred checksum are identical, otherwise gives failure for file transfer
  - uses md5

- Dynafed developers are working on general checksum support
- DPM developers also work on supporting both methods
  - dCache has it already implemented (v5.x)

Dynafed development

third party copy (TPC):

- implemented in latest release 1.4.x

- implementation is script based
  - different scripts for pull/push
  - on request executes specified script
  - you can do whatever you prefer in the script to do the copy
    - a way to support checksums during file transfers
    - transfers between endpoints using different protocols possible
      - e.g. between dynafed and plain xrootd site
  - if endpoint supports redirection, then dynafed redirects copy directly to endpoints
    - otherwise dynafed acts as client for TPC

- participate in WLCG TPC DOMA (http)
  - testing sites to have http/webdav based TPC working between all sites
  - we should have it working very soon for our Dynafed instance
    - just matter of correct configuration now

Dynafed development

space awareness:

- Dynafed can be used for writing
  - same logic like for reading: use nearest site based on GeoIP db
  - problem: endpoint is used even if not enough space available

- solution: we developed a system that queries all endpoints for free/used storage and stores information in memcache until next query (https://github.com/hep-gc/dynafed_storagestats)
  - external scripts executed via crond

- supports Azure, S3, WebDav (RFC4331)
  - DPM and dCache supporting RFC4331 conform queries
    - only in latest releases, many sites still run on old versions....
  - ceph-admin and AWS cloudwatch based queries supported too

- at write:
  - create list of write-enabled endpoints sorted by GeoIP
  - get file size from write request
  - remove endpoints that do not have enough free storage space available
  - redirect write request according to remaining list of endpoints

Dynafed development

space awareness:

- Dynafed can be used for writing
  - same logic like for reading: use nearest site based on GeoIP db
  - problem: endpoint is used even if not enough space available

- solution: we developed a system that queries all endpoints for free/used storage and stores information in memcache until next query (https://github.com/hep-gc/dynafed_storagestats)
  - external scripts executed via crond

- supports Azure, S3, WebDav (RFC4331)
  - DPM and dCache supporting RFC4331 conform queries
    - only in latest releases, many sites still run on old versions....
  - ceph-admin and AWS cloudwatch based queries supported too

- at write: **Implemented in latest testing version of Dynafed, to be released in stable soon!**
  - create list of write-enabled endpoints sorted by GeoIP
  - get file size from write request
  - remove endpoints that do not have enough free storage space available
  - redirect write request according to remaining list of endpoints

Dynafed development

space reporting and accounting:

- json based
  - have json file in specific format and location on the storage
  - lists VO, VO disk allocation, used space, free space,....
  - needs to be created automatically

- same system that does the free/used space collection can also create such json file automatically
  - can also write information out in different formats to be processed externally for more complicated setups (e.g. having multiple copies of files) or endpoint specific storage statistics

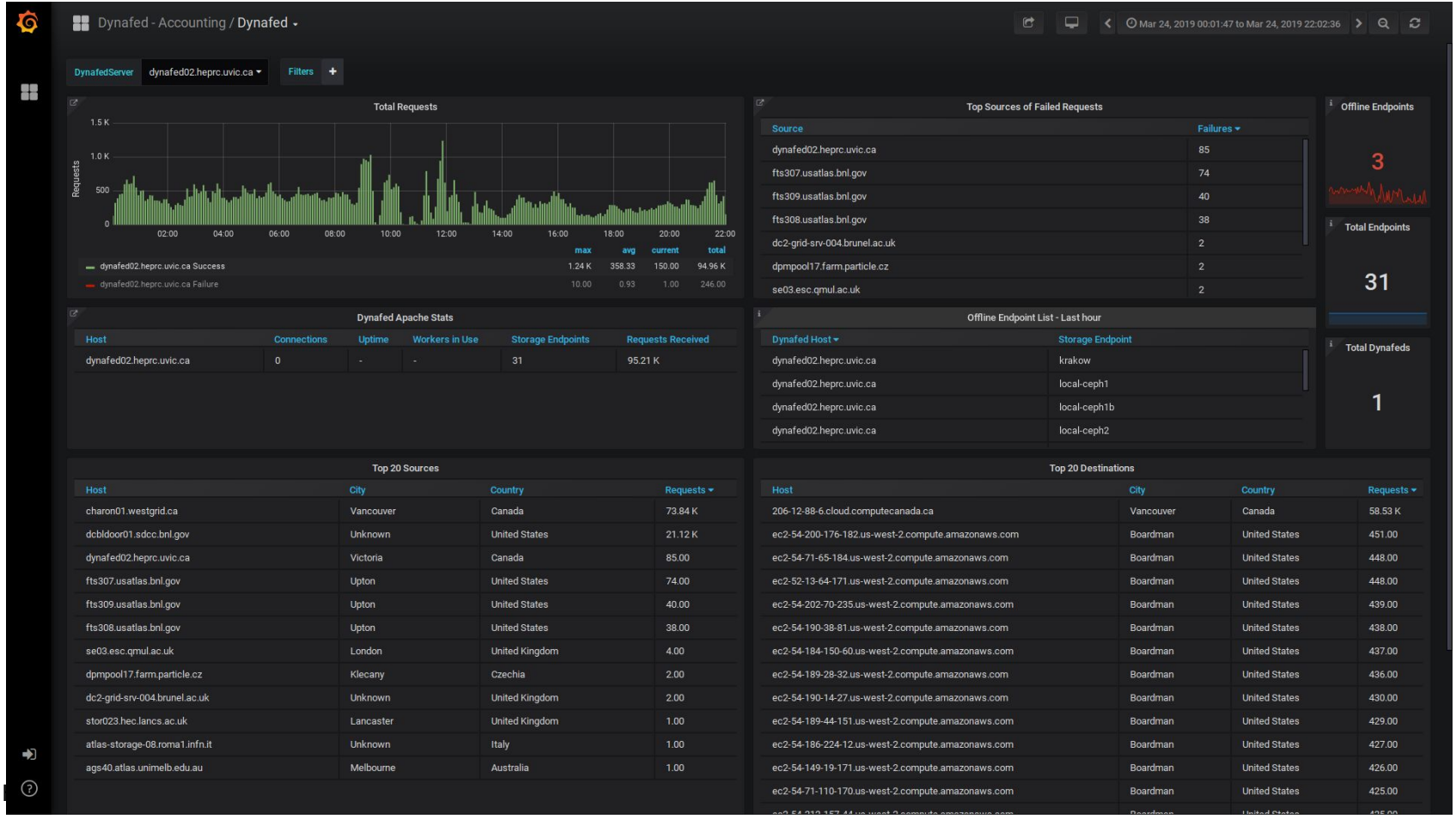
Other development

monitoring:

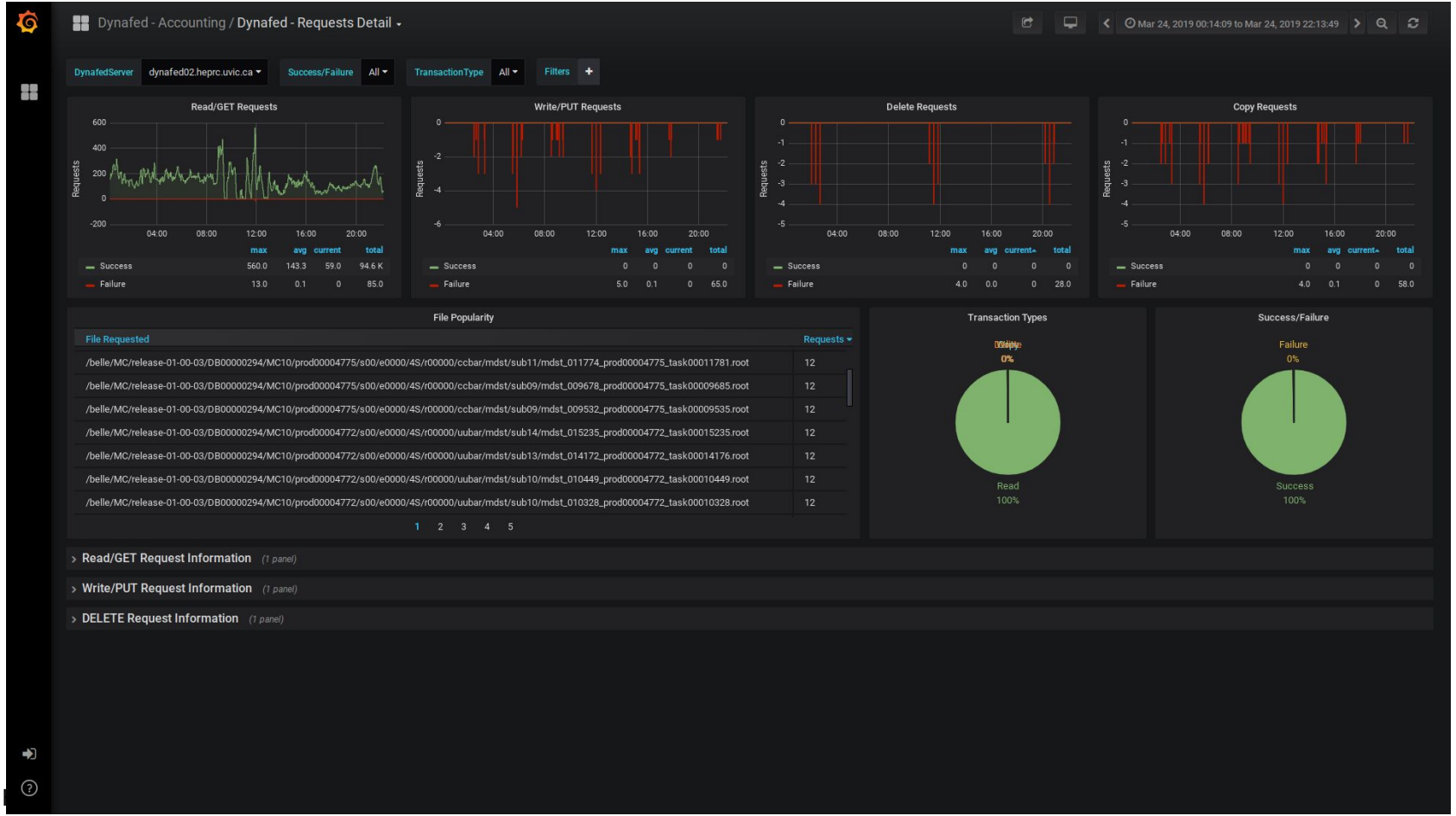
- we developed own monitoring
  - source of file request
  - redirection endpoint
  - files that got requested and how often
  - endpoint storage information (free space, used space, latency, quota,...)
  - timeline for file requests
  - separated for read/write/delete/copy requests

All information available on command line too (via curl).

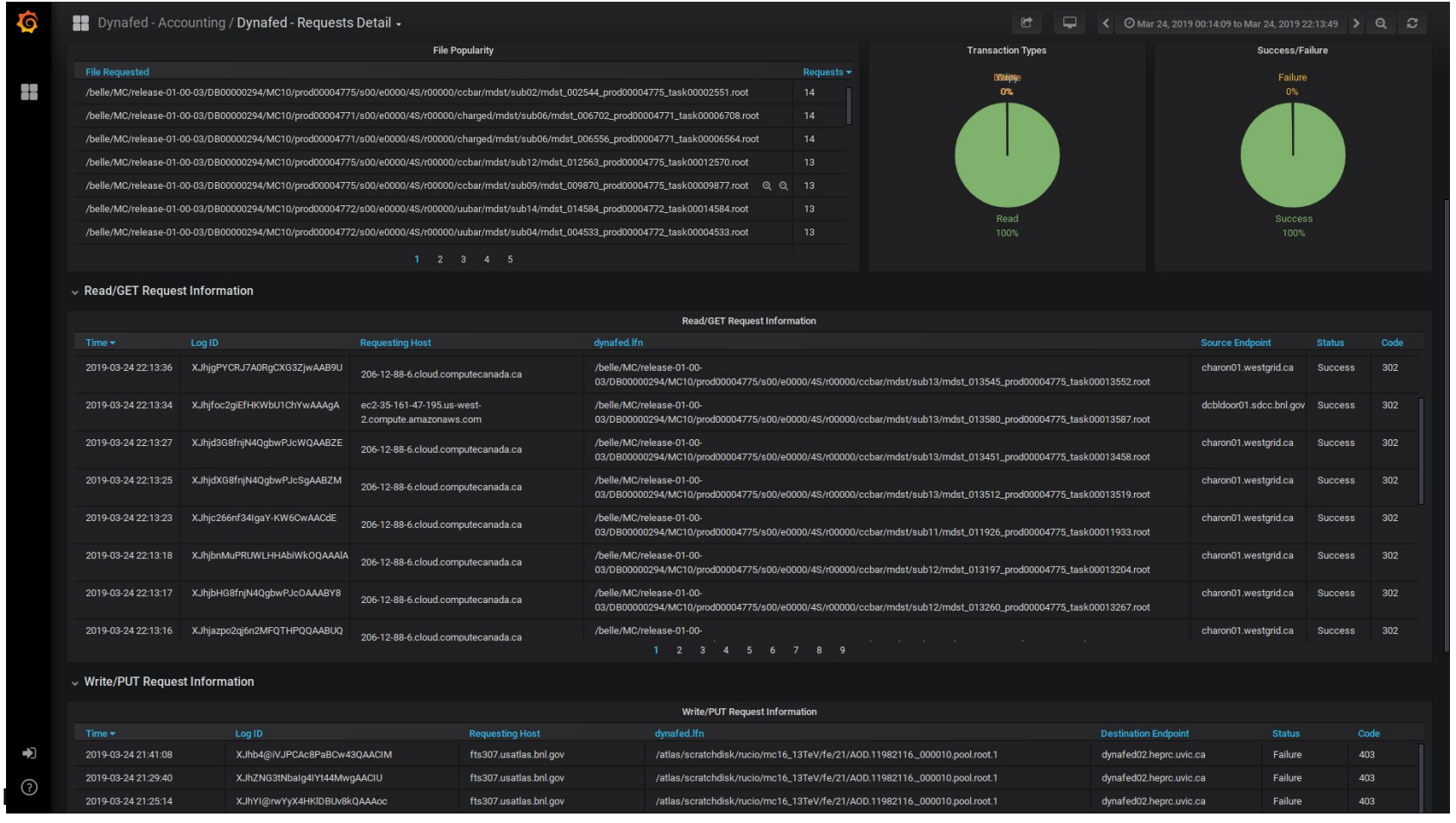
Dynafed monitoring



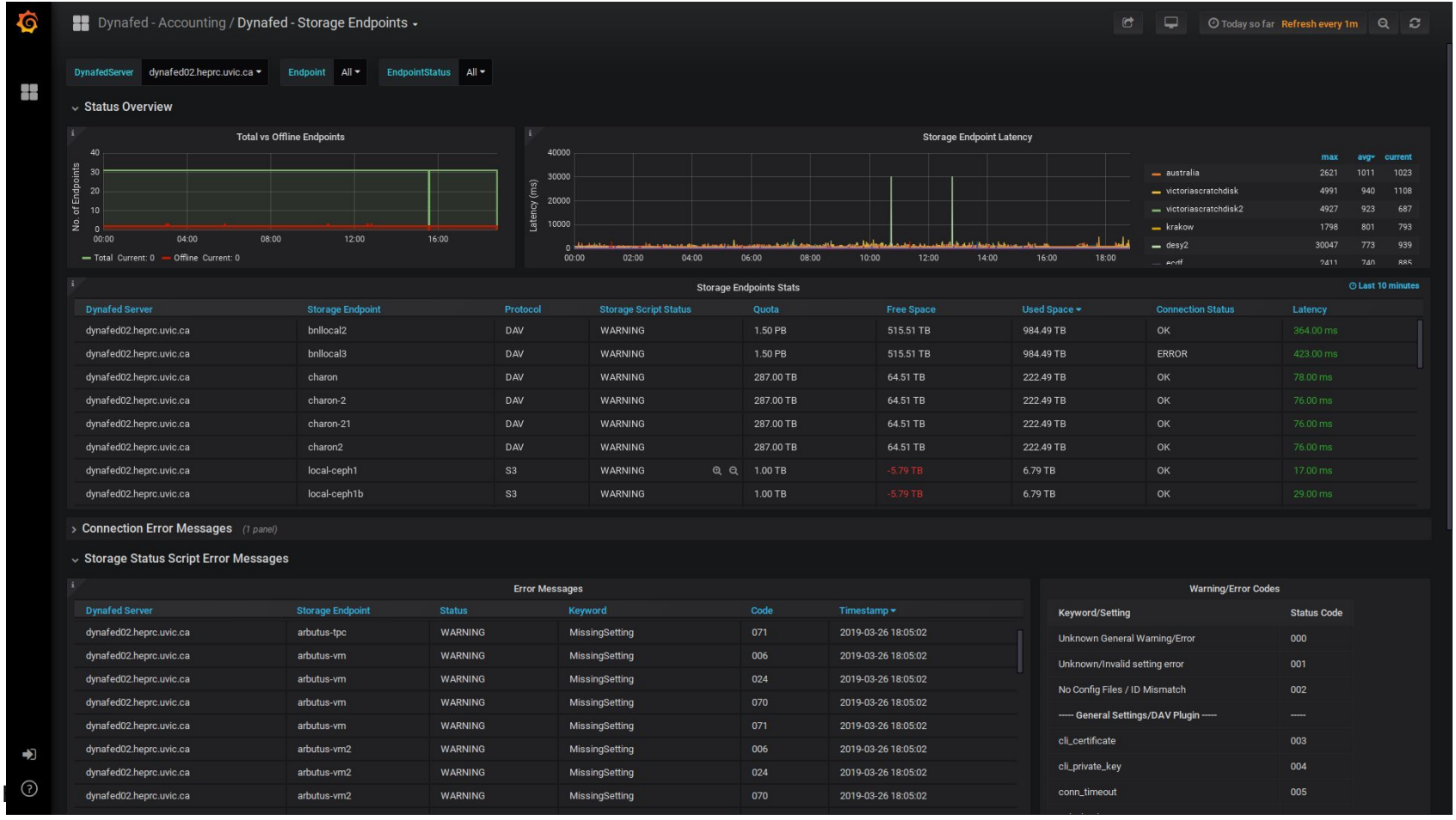
Dynafed monitoring



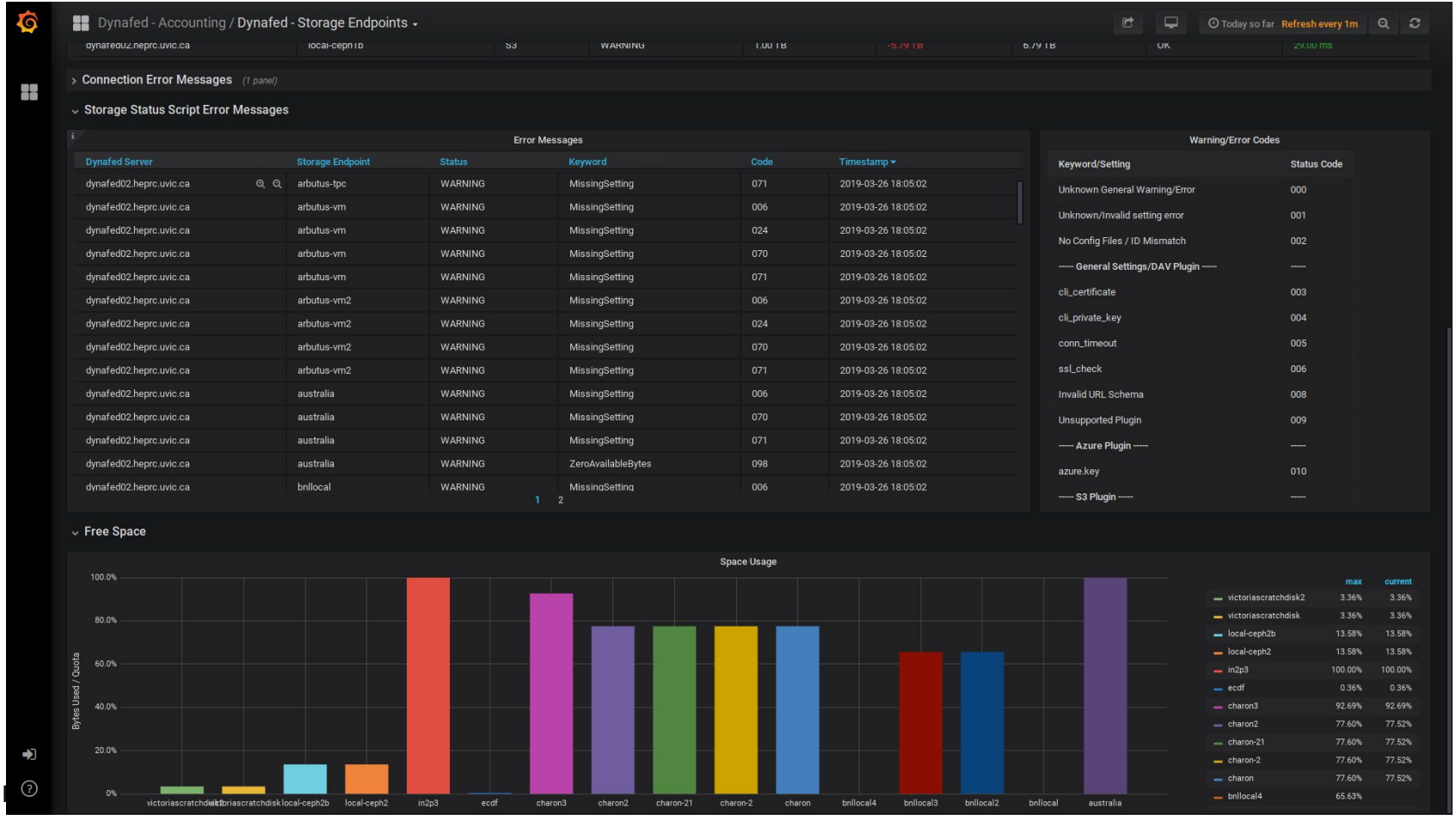
Dynafed monitoring



Dynafed monitoring



Dynafed monitoring



Other development

to do:

- using file popularity to automatically redistribute most often used files
  - we know how often a file is accessed in the last hours/days/weeks
  - we know from where it is accessed
  - we know which storage is close by and how much space is available on it
  - dynafed has all information to access the endpoint for write operations

Other development

to do:

- using file popularity to automatically redistribute most often used files
  - we know how often a file is accessed in the last hours/days/weeks
  - we know from where it is accessed
  - we know which storage is close by and how much space is available on it
  - dynafed has all information to access the endpoint for write operations

Just need to put some logic in a script and it should work fine....

Summary

- Dynafed is a great tool for us to
  - use distributed storage endpoints for our distributed cloud based Grid computing
  - easily add new “site” storage without most of the GRID storage knowledge needed for traditional Grid storage systems
  - easily integrate object stores to the Grid (Ceph/S3)
    - can easily be used by everyone, in our dynafed instance for Belle/Atlas with own accounting if needed or in an own installation

- nearly at the point where we can replace a traditional Grid SE with Dynafed
  - at least for on-disk storage

- TPC to be tested and configured
- storage space information usable for write decision
- full checksum support only open issue left

- we have very good monitoring available
  - development ongoing for other features we want, e.g. memcache information, open connections, number of http workers,....
- work on automated redistribution between endpoints depending on file popularity

Thank you!

additional information:
HEP-RC blog
Dynafed@CERN
CERN dynafed users forum
CERN dynafed support archive

What is Dynafed

- redirector for a dynamic data federation, developed by CERN-IT (fabrizio.furano@cern.ch)
  - for data transfers, client is redirected to a storage element with the data
  - this can be done depending on geographic location
    - storage elements closer to the job are preferred

Example:

file: https://dynafed02.heprc.uvic.ca:8443/belle/MC/release-00-09-00/DB00000265/BG15th/phase3/set14/BHWide.tgz
meta data: curl -k https://dynafed02.heprc.uvic.ca:8443/belle/MC/release-00-09-00/DB00000265/BG15th/phase3/set14/BHWide.tgz?metalink

on cc-east cloud (Sherbrooke):
http://206.167.180.208:80/belle/MC/...
https://gridftp02.clumeq.mcgill.ca:8443/webdav/belle/DATA/belle/MC/...
http://129.114.33.181:80/belle/MC/...
https://s3-uvic.dev.computecanada.ca/rjsBucket/belle/MC/...
http://elephant132.heprc.uvic.ca/mebucket/belle/MC/...

on cc-west cloud (Victoria):
https://s3-uvic.dev.computecanada.ca/rjsBucket/belle/MC/...
http://129.114.33.181:80/belle/MC/...
https://gridftp02.clumeq.mcgill.ca:8443/webdav/belle/DATA/belle/MC/...
http://206.167.180.208:80/belle/MC/...

Advantages of using S3 based storage

- easy to manage
  - no extra servers needed, no need for the whole Grid infrastructure on site (DPM, mysql, apache, gridftp, xrootd, VOMS information, grid-mapfile, accounting, ...)
  - just use private/public access key in central Dynafed installation

- no need for extra manpower to manage a grid storage site
  - small group with budget to provide storage but no manpower for it: Just buy S3 based xTB for y years and put the information into dynafed ---> instantly available to the Grid, no need to buy/manage/update extra hardware
  - if university/lab has already large Ceph installation --> just ask for/create a bucket, and put credentials in dynafed

- industry standard
  - adapted from Amazon by Open Source and commercial cloud and storage solutions
    - HPC, Openstack, Ceph, Google, Rackspace cloud storage, NetApp, IBM,...

- scalable
  - traditional local file storage servers based on traditional filesystems will become harder to manage/use with growing capacity needs, same for other “bundle” solutions (DPM,...)
  - raid5 dead, raid6 basically dead too, ZFS will get problems with network performance

Access to S3 based storage

glb.locplugin[]: libugrlocplugin_s3.so localceph2 2 http://elephant132.heprc.uvic.ca/mebucket/belle
locplugin.localceph2.xlatepfx: /belle /
locplugin.localceph2.s3.pub_key: <PUPLIC-KEY>
locplugin.localceph2.s3.priv_key: <PRIVATE-KEY>
locplugin.localceph2.writable: true
locplugin.localceph2.s3.signaturevalidity: 3600
locplugin.localceph2.s3.region: us-east-1
locplugin.localceph2.s3.alternate: true

---

gfal-copy davs://dynafed02.heprc.uvic.ca:8443/belle/datadisk/space-usage.json local-file.json

Process: Contact Dynafed --> Dynafed looks where file is ---> Dynafed gets authorized link -->
---> Dynafed redirects client to this link ---> Client access file direct on S3 through that link

http://elephant132.heprc.uvic.ca/mebucket/belle/datadisk/space-usage.json?X-Amz-Signature=2d8fc601379eb9e43dc16219a9da11452e8b7c0a22ad98186aaa4fe841b97e53&X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Credential=TJHJA902TSSJZ659E9D5%2F20171016%2Fus-east-1%2Fs3%2Faws4_request&X-Amz-Date=20171016T061053Z&X-Amz-Expires=3600&X-Amz-SignedHeaders=host

Authentication/Authorization in Dynafed

- built-in VOMS proxy based authentication
  - in ugr.conf: *glb.allowgroups[]: belle /belle/ rl*
  - includes authentication and authorization
  - does not work in web browsers

- python-module based authentication possible
  - can be used for anything you want to use
  - exit code 0 = access granted
  - exit code 1 = access denied

- implemented usage of grid-mapfile for authentication
  - more info in our blogpost
  - plain text file used for authorization, read by the python module

Authentication/Authorization in Dynafed

- built-in VOMS proxy based authentication
  - in ugr.conf: *glb.allowgroups[]: belle /belle/ rl*
  - includes authentication and authorization
  - does not work in web browsers

- python-module based authentication possible
  - can be used for anything you want to use
  - exit code 0 = access granted
  - exit code 1 = access denied

- implemented usage of grid-mapfile for authentication
  - more info in our blogpost
  - plain text file used for authorization, read by the python module

Example of access file:

/atlas atlas rlwd
/belle belle rlwd
/minio admin rldw
/localCeph admin rlw

Authentication/Authorization in Dynafed

- built-in VOMS proxy based authentication
  - in ugr.conf: *glb.allowgroups[]: belle /belle/ rl*
  - includes authentication and authorization
  - does not work in web browsers

- python-module based authentication possible
  - can be used for anything you want to use
  - exit code 0 = access granted
  - exit code 1 = access denied

- implemented usage of grid-mapfile for authentication
  - more info in our blogpost
  - plain text file used for authorization, read by the python module

- also implemented alternative version of grid-mapfile usage without the need of a python module
  - you can read about it here

Redirect and direct access with Dynafed

- client tools can get new redirect to another site if anything happens with an already established connection
  - site outage, network problems at a site,....

- root based tools can speak webdav and access data over network using dynafed
  - *TFile *f=TFile::Open("davs://dynafed.server:PORT/belle/path/to/file/file.root")*
  - uses external davix libraries

Possibility to open root files over the network with redirect to closest storage (on-site) and seamless switchover to other storage endpoints in case of problems!