Integrating Dynafed into the ATLAS and Belle II Data and Workload Management Systems

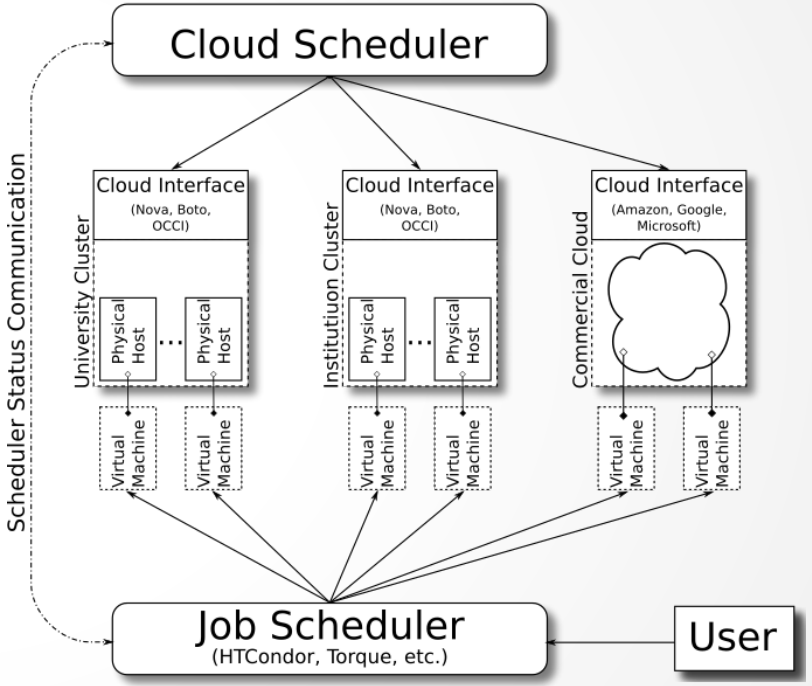
Frank Berghaus

on behalf of

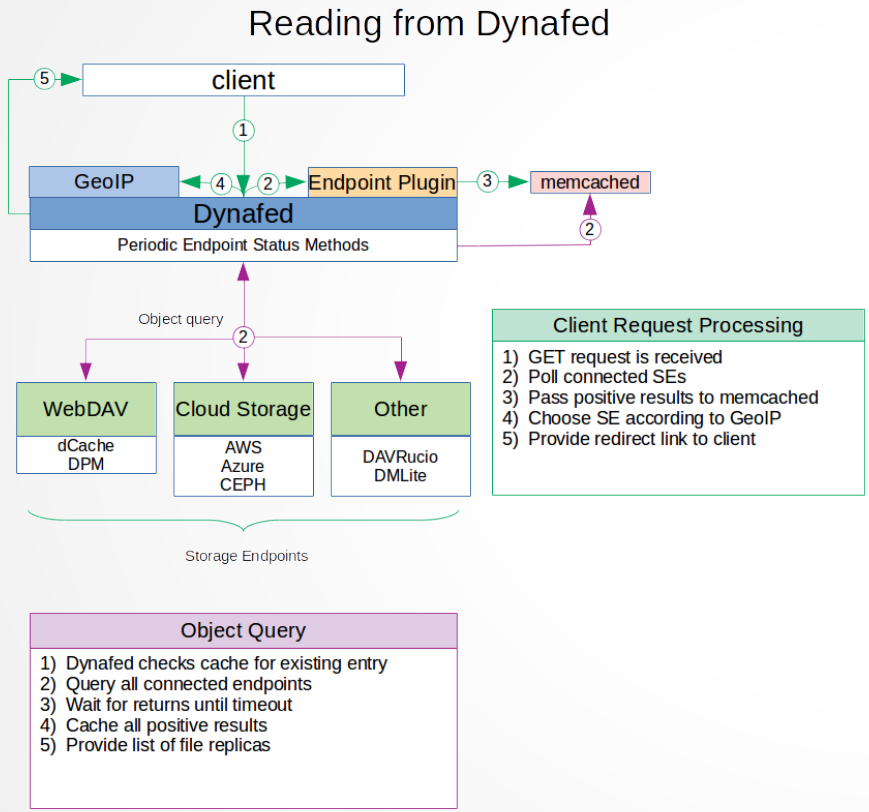
UVic, TRIUMF, CERN-IT, ATLAS, Belle-II

Introduction & Motivation

- Distributed cloud system
  - cloudscheduler.org
  - 8th birthday [arXiv:1007.0050]
  - User: DIRAC (Belle-II) or PanDA (ATLAS)
- Cloud Scheduler at UVic and CERN
- Cloud Resources:
  - In Canada, US, UK, Germany, and at CERN
  - O(10^3) cores – easy to add more
- CE: HTCondor & Cloudscheduler
- SE: dCache (UVic), EOS (CERN)
- Limited by remote access to storage



Dynafed: Redirect To Nearby Storage



- Dynafed redirects to close storage
- Operating three configurations:
  - Belle-II at UVic:
    - R/O access (production)
  - ATLAS at CERN:
    - R/W to cloud storage (dev)
    - R/W to grid storage (dev)
- Instances operated by others:
  - data-bridge at CERN for *@home
  - Dynafed at INFN for Belle-II
- Part of a WLCG Demonstrator

Dynafed and Belle-II

Victoria Dynafed for Belle-II

- Belle-II *is developing* gfal2 support for their DDM and WMS
  - Will allow direct usage of Dynafed as SE in the future
- Workaround:
  - Belle-II allows job configuration to access locally mounted volume
  - gfalFS provides fuse mount within Linux directory tree:

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

  - Jobs access Belle-II data from “local” directory ~/b2data/belle
- gfalFS only needs Dynafed as access point for any cloud
  - Dynafed redirects gfalFS to closest endpoint via GeoIP

Victoria Dynafed for Belle-II

- Endpoints behind Belle-II Dynafed:
  - Compute Canada East (Minio via S3 API)
  - Compute Canada West (Minio via S3 API)
  - Amazon (S3)
  - Chameleon Cloud (Minio via S3 API)
  - Victoria Tier-2 SE (dCache, Victoria SE for Belle-II)
  - Victoria HEPRC Ceph (CephS3)

Dynafed Belle-II Authentication & Authorization

- Based on X.509 VOMS proxy
- Python-module based authentication
  - Implemented grid-mapfile for authentication
  - Plain text file read by python script for authorization, example:

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

- Proxy sent to worker from DIRAC and used for gfalFS mount

Belle-II Experience With gfalFS And Dynafed

- Load is balanced across co-located storage endpoints
  - In recent MC campaign jobs each pulled one (1) of 20 input data sets - each ~5GB
  - 3000 parallel jobs → 30TB per day. Ran smoothly through Dynafed.
- Easy and effective network usage
  - Same configuration for all workers (6 separate clouds are used for Belle-II)
  - With same files used by many jobs network transfers stay local
- Easy addition of new endpoints
  - Added traditional Belle-II SEs while transferring new input data sets to own Endpoints:
    - Instant access to new files without configuration change on jobs/workers
- gfalFS and Dynafed work well for reading input data
  - Output is still written to UVic dCache using SRM
  - Waiting on gfal2 to be added to Belle-II offline computing

Belle-II Experience With gfalFS And Dynafed

- System stability
  - Data is accessed locally with fail-overs observed when local storage becomes unavailable
    - *local* means nearby storage in Dynafed
  - Placing endpoints close to each other effectively balanced load
- **In production for almost 1 year**
- Segfaults of gfalFS (in libcrypto.so) since CernVM update
  - Intermittent issues, thus hard to debug
  - Appeared during Spectre/Meltdown patches
  - Observed when same endpoint is mounted multiple times on one system
  - Workaround: cron jobs remounts failed mounts

Dynafed and ATLAS

Dynafed for ATLAS

- Grid Endpoints:

  dynafed-atlas.cern.ch/data/grid

  - CERN (EOS), LRZ (dCache), ECDF (DPM)
  - CERN-EXTENSION_GRIDDISK
- Cloud Endpoints:

  dynafed-atlas.cern.ch/data/cloud

  - CERN (CephS3)
  - CERN-EXTSION_CLOUDDISK

- Users authenticate to Dynafed via X.509+VOMS:

```
glb.allowgroups[]: "/atlas/*" /data rwl
glb.allowgroups[]: "/atlas/Role=production/*" /data rlwd
```

- Allow ATLAS Users to browse Dynafed by harvesting DNs from VOMS:

```
glb.allowusers[]: "/DC=ch/DC=cern/OU=Organic..." /data rl
…
```

- *Recall*: Rucio supports and SEs expose `HTTP+WebDAV`

Experience With ATLAS and Dynafed

- Early challenges:
  - `MKCOL` and `MOVE` not supported by Dynafed (not guaranteed on object stores), solutions:
    - Rucio protocol that does not rename files after upload
    - Dynafed allows `MKCOL` by creating directories in cache and making appropriate calls on file system endpoints
  - Rucio Replication to `https/davs` endpoints get stuck
    - Manually make calls to FTS and register data
- Early Success:
  - Functional tests run against Dynafed with CephS3 endpoint

Rucio, Dynafed, and Checksums

- Rucio “TODO” fix causes jobs to fail: client can’t get **checksum**
- Mechanism:
  - Grid: User is responsible, `Want-Digest [RFC3230]`
  - Cloud: Provider is responsible, `Content-MD5 [RFC1544]`
- Algorithm
  - Grid: `ADLER32 [RFC1950]`, for many reasons, can support others
  - Cloud: `MD5 [RFC1321]` (because it was there?)
- *Workaround*: Flag for Rucio not to request checksum from Dynafed
- Rucio client now calculates ADLER32 and MD5 on upload, thoughts (?)

Dynafed, FTS, and Proxies

- Rucio used legacy proxies with FTS
  - Some old SEs may not have migrated to RFC
  - Legacy proxy delegation through Dynafed to LRZ dCache were denied
  - Solution: switch to RFC proxy
- `MKCOL` bug:
  - Dynafed's `MKCOL` implementation [slide 12] broken for `davs` endpoints
  - Fixed in Dynafed 1.3.2
- Automated replication by Rucio to/from Dynafed works

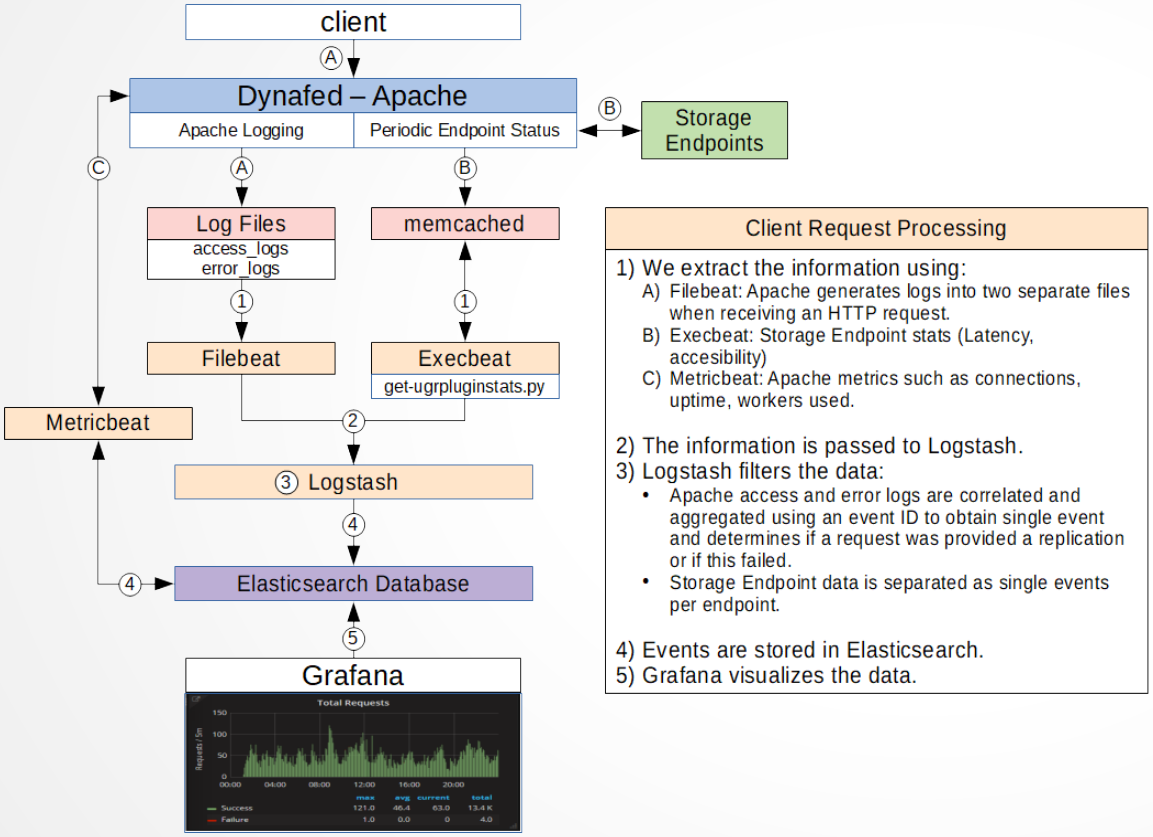
Debug & Development Cycle

- Rucio, Dynafed, and FTS developers are quick to respond and helpful
- conservative deployment policies → long development cycle
- Example: Rucio client & checksums
  - In March implemented behavior for rucio client to respect "`verify_checksum`" flag
  - Debug: caught wrong exception (last week in ATLAS test pilots)
  - This week: implement correct exception
  - Next week: patch released
  - Few weeks: Wait for deployment of patch in cvmfs
  - Test on cloud+dynafed system

Pull Request, Automated Tests, Peer Review, ...

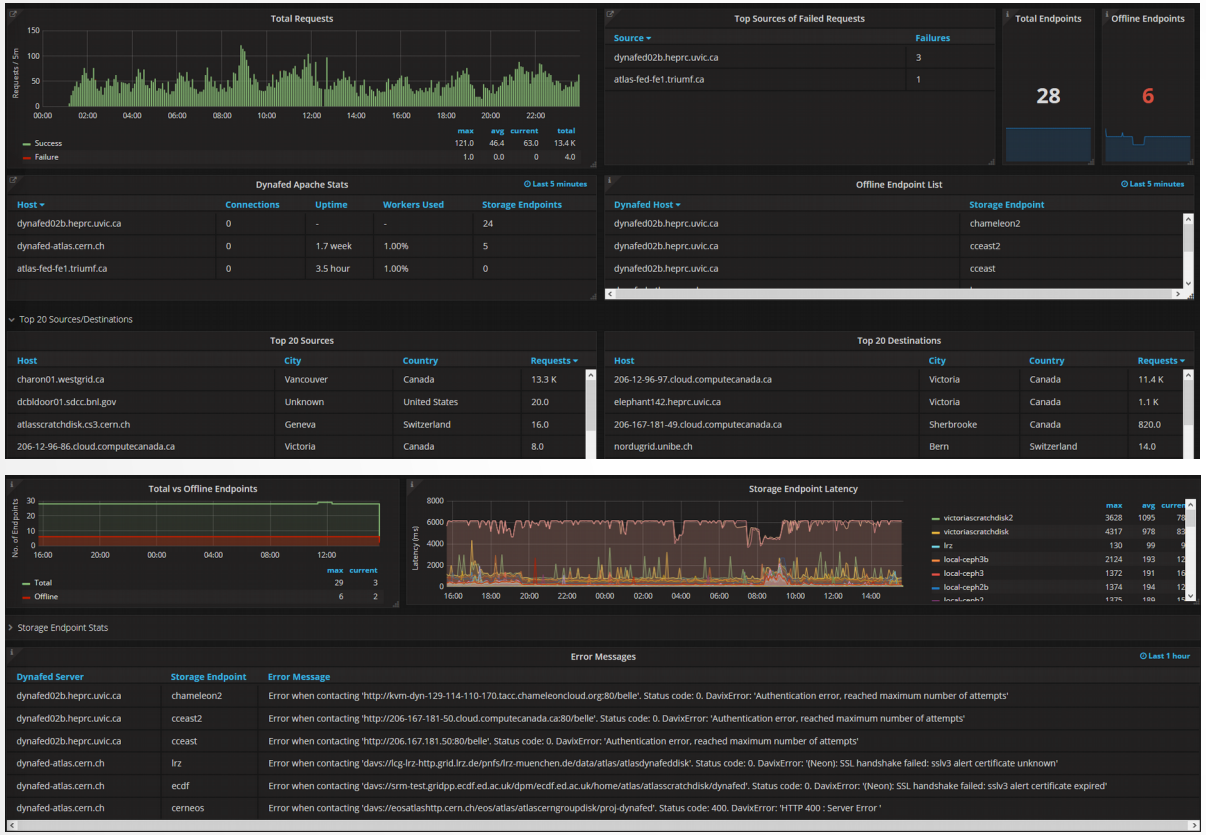
Monitoring Dynafed

Elasticsearch, Logstash, and Grafana



- Filebeat: Apache access and error log files
- Excecbeat: Python script to extract stats from memcached
- Metricbeat: Apache metrics
- Grafana display for:
  - Load, Endpoint Status, Requests, and Redirects

Elasticsearch, Logstash, and Grafana

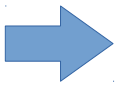


- Main Dashboard
  - Apache Metrics Stats
  - Online/Offline SE's
  - Top Sources and Destinations
  - Total Success/Fail Requests
- Storage Status Dashboard
  - Error messages
  - Online/Offline SE's
  - SE's Latency to Dynafed.

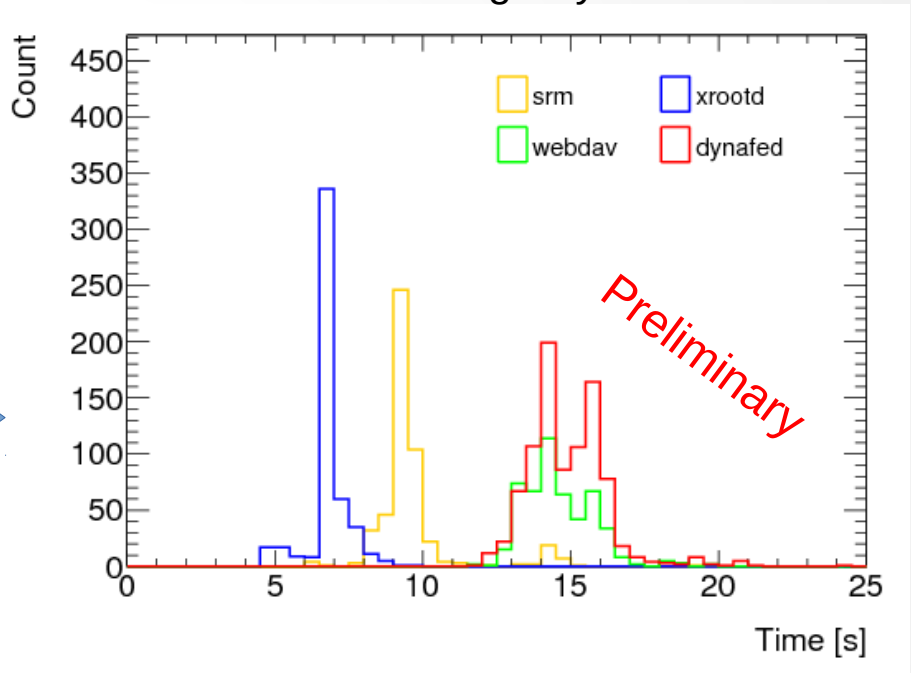
Dynafed: Next Steps

Next Steps with Dynafed

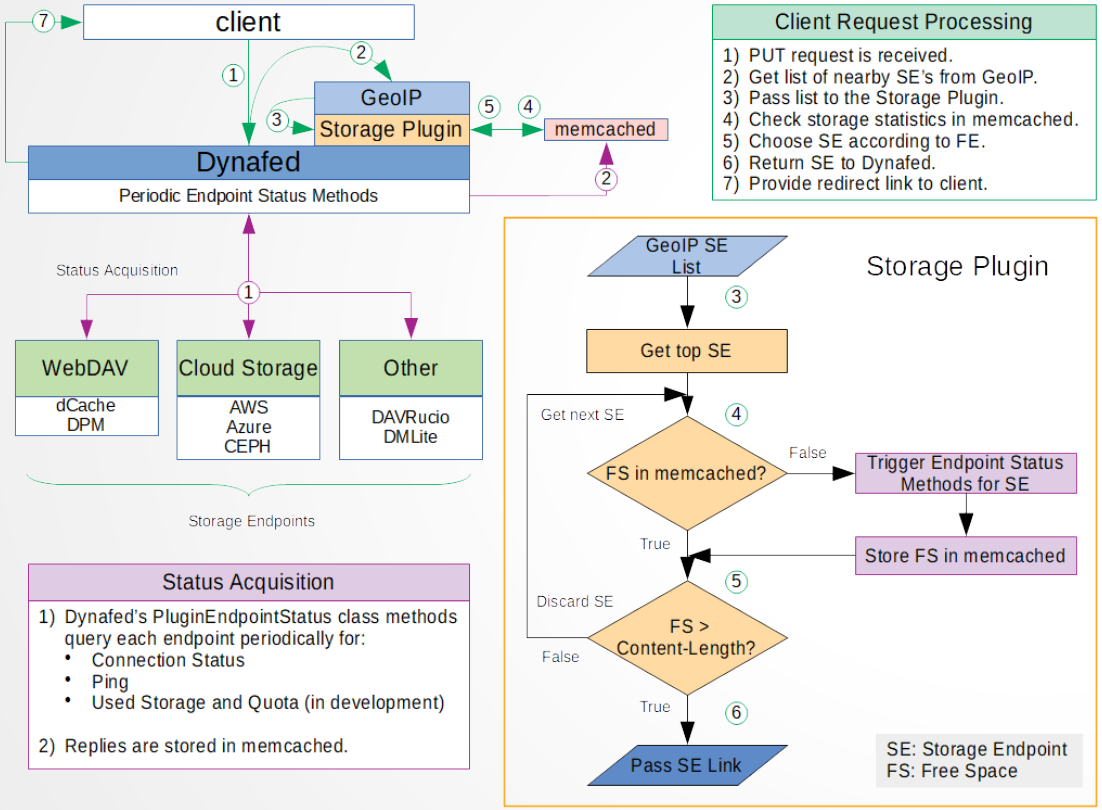
- Integrate gfal2 into Belle-II
  - Next MC campaign needs 9TB, with 2GB per job
  - Full dataset cloud will be more challenging
- Run ATLAS functional tests (again)
- Benchmark Dynafed:
  - `HTTP+WebDAV` vs `xrootd/srm`
  - `HTTP+WebDAV` with Dynafed vs direct
  - Qualification task for Benjamin Rottler
    - Marcus Schumacher's group at Uni Freiburg
- Rucio Functional Tests for xrootd and WebDAV
- Run ATLAS production
- Get cloud storage at other sites



Local Test Streaming Physics Data



Next Steps: Dynafed Storage Plugin



- Issue with writing to Dynafed
  - Free space on endpoints unknown
- Need method to query usage and quota from endpoints
- Need common protocol for this information
- Commercial providers don't provide quota: we choose?

Thank You!