Dynamic Federations

Seamless aggregation of open-protocol-based storage endpoints

Fabrizio Furano (presenter)
Patrick Fuhrmann
Paul Millar
Daniel Becker
Adrien Devresse
Oliver Keeble
Ricardo Brito da Rocha
Alejandro Alvarez

Credits to ShuTing Liao (ASGC)
Dyn Federations in a nutshell

- Goals:
  - Browse/access a huge repository made of many sites without requiring a static index
    - No “registration”, no maintenance of catalogues
  - Redirect intelligently clients asking for replicas
  - Automatically detect and avoid sites that go offline
  - Accommodate both algorithmic name translations and catalog-based ones, at the same time
  - Correctly map on the fly existing SRM TURLS to HTTP URLs
  - Flexibility: accommodate almost any kind of endpoint
  - High robustness (=correctly treats failures)
  - High performance (=many requests per second per frontend)
  - Allow interactivity (achievable only with quick systems)
  - Scalability (=frontends can be replicated indefinitely)
  - Geographical scalability (=can have many frontends in different places)
The basic idea

Storage/MD endpoint 1

.../dir1/file1
.../dir1/file2

Storage/MD endpoint 2

.../dir1/file2
.../dir1/file3
The basic idea

This is what we want to see as users:

Sites remain independent and participate to a global view.

All the metadata interactions are hidden and done on the fly.

NO persistency needed here, just efficiency and parallelism.

Aggregation:

```
/dir1
/dir1/file1
/dir1/file2
/dir1/
```

Storage/MD endpoint 1:

```
.../dir1/file1
.../dir1/file2
```

Storage/MD endpoint 2:

```
.../dir1/file2
.../dir1/file3
```

With 2 replicas:

30 Oct 2012
Dynamic Federations

• Technically “loosely coupled storage systems”
• Idea: an unified entry point for a *federation of endpoints*
  – “lonely” storage clusters (e.g. dCache, DPM, plain HTTP servers)
  – even catalogues (e.g. LFCs) indexing their storage elements
• Idea: *make it dynamic*
  – The task of federating is done *on the fly*, just communicating with the endpoints
• This entry point knows its endpoints, can redirect clients to them, it can merge and present their metadata to browsing clients
• Many interesting possibilities
  – Federate third party outsourced HTTP/DAV servers (also clouds)
  – Federate the content of fast changing things, like SQUID caches
  – Federate them together with the information of some experiment’s DB
  – When requesting a file we would get it from an endpoint that is good for us, it could be a cache or a non-cache one.
  – Automatically detect if the endpoint is working
  – See as one two experiment’s DBs (e.g. two LFCs or LFC plus Rucio), plus whatever other Cloud-like storage endpoint
    • Transparent, direct access to the official replicas AND the external ones as well
Dynamic Federations

• Federating storage means giving seamless access with thin, standard clients
  – without the need of switching communication protocol
• We know the use cases
  – Easy, direct job/user data access, WAN friendly
  – Failover is part of the concept
  – Friend sites can share storage and bandwidth
  – Easy access to shared things like conditions
  – Support for advanced site choices, e.g. self healing
• We like HTTP and DAV
• We want to take the concept to the next level
Dynamic Federations

• What’s the goal?
  – Make different storage clusters be seen as one
  – Make global file-based data access seamless

• How should this be done?
  – No strange APIs, everything looks “banal”
  – Participate in the HTTP HEP/HPC ecosystem that is growing now
  – Use dynamic systems that are easy to setup/maintain:
    • no complex metadata persistency
    • no DB babysitting (keep it for the experiment’s metadata)
    • no replica catalogue inconsistencies
  – Head to high performance

• Local SE as a preference, give the freedom to point to an efficient and reliable global federation
  – Optimize redirections based on on-the-fly client-data proximity
  – Avoid inconsistencies, just looking at where the files are now, and at the real status of the endpoint (online/offline)
  – Limit complexity: read only (by now), as usually writes happen to well-known, close islands
How it works

Frontend (Apache2+DMLite)

Federator

Metadata cache

Plugin  Plugin  Plugin  Plugin  Plugin  Plugin  Plugin

Catalog e.g. LFC/Rucio

Catalog e.g. LFC

SE  SE  SE  SE
How it works

Frontend (Apache2+DMLite)

Federator

Metadata cache

Plugin Plugin Plugin Plugin Plugin Plugin Plugin

Catalog e.g. LFC/Rucio

Catalog e.g. LFC

SE SE SE

Europe

Asia-Pacific
How it works

Catalog e.g. LFC

SE

SE

SE

Metadata cache

(Apache+DMLite)

Plugin

Plugin

Plugin

Plugin

Plugin

Plugin

Plugin

Catalog e.g. LFC/Rucio

SE

SE

SE

Europe

Asia-pacific

Middle East

How it works
How it works:

- **Frontend (Apache2+DMLite)**
- **Federator**
- **Metadata cache**

The cache remembers what happened:

The next metadata interactions will very likely be cached.

Diagram:
- **Catalog e.g. LFC/Rucio**
- **SE**
- **Federator**
- **Plugin**
- **Metadata cache**
- **Catalog e.g. LFC**
- **SE**
Demo

• We have a stable demo testbed, using HTTP/DAV http://federation.desy.de/myfed

• It is actually 2 demos in one
  – An ATLAS demo, federating 8 sites, plus LFC as name translator
    • Note that this is not the full ATLAS repo, it’s just 8 sites.
    • DESY, KIT, SARA, WUPPERTAL, NDGF, Muenchen, Prague, ASGC
  – A fully dynamic catalogue-free demo with the EMI testbed
    • Federating three endpoints.
      – a DPM instance at CERN
      – a dCache instance in DESY
      – one endpoint in LBNL

• What does it do
  – If you request a file, the best replica (for your location) is returned from one of the known storage elements
  – In no case it will return a SE that is not available
  – In addition, it allows you to browse the namespace of the ATLAS LFC (working to federate Rucio together with it)

• The feeling it gives is surprising
  – Metadata performance is in avg much higher than contacting the endpoints

• We see the directories as merged, as if it were only one system

• 10K files are interleaved in a 4-levels deep directory /myfed/dteam/ugrtest/interleaved
  – Oddly-numbered files are at CERN, evenly-numbered files are at Desy

• 10K files have 2 replicas in DESY and CERN: /myfed/dteam/ugrtest/interleaved
Dynamic Federations

• Currently available!

• Technically TODAY we can dynamically aggregate:
  – dCache DAV/HTTP instances
  – DPM DAV/HTTP instances
  – LFC DAV/HTTP and old Cns API instances
  – Cloud DAV/HTTP services
  – Anything that can be plugged into DMLite (the new architecture for DPM/LFC)
  – Can be extended to other metadata sources

• The system also can load a “Geo” plugin
  – Gives a geographical location to replicas and clients
  – Allows the core to choose the replica that is closer to the client

• The one that’s available uses GeoIP (free)
Why HTTP/DAV?

• **It’s there**, whatever platform we consider
  – A very widely adopted technology

• **We (humans) like browsers**, they give an experience of **simplicity**

• Goes towards **convergence**
  – Users can use their devices to access their data easily, out of the box
  – Jobs just go straight to the data

• With direct access to data pre-location becomes an optimization choice, not a constraint of the technology
System design

• A system that only works is not sufficient
• To be usable, it must privilege speed, parallelism, scalability
• The core component is a plugin-based component called originally “Uniform Generic Redirector” (Ugr)
  – Can plug into an Apache server thanks to the DMLITE and DAV-DMLITE modules (by IT-GT)
  – Composes on the fly the aggregated metadata views by managing parallel tasks of information location
    • Never stacks up latencies!
  – Makes browsable a sparse collection of file/directory metadata
  – Able to redirect clients to replicas of hosts known to be working in that moment
  – By construction, the responses are a data structure that models a partial, volatile namespace
  – Keep them in an LRU fashion and we have a fast 1st level namespace cache
    • Peak performance is ~500K->1M hits/second per core by now
Focus: performance

- Performance and scalability have primary importance
  - Otherwise it’s useless...

- Full parallelism
  - No limit to the number of outstanding clients/tasks
  - No global locks/serializations!
  - The endpoints are treated in a completely independent way
  - Thread pools, prod/consumer queues used extensively (e.g. to stat N items in M endpoints while X clients wait for some items)

- Aggressive metadata caching
  - A relaxed, hash-based, in-memory partial name space
  - Juggles info in order to always contain what’s needed
  - Stalls clients the minimum time that is necessary to juggle their information bits
  - Peak caching perf per CPU core: 0.5~1M stats/sec

- Spurred a high performance DAV client implementation (DAVIX)
  - Wraps DAV calls into a POSIX-like API, saves from the difficulty of composing requests/responses
  - Loaded by the core as a “location” plugin
  - Performance is privileged: uses libneon w/ sessions caching
  - Compound list/stat operations are supported
Concurrent metadata caching

Clients come and are distributed through:
- different machines (DNS alias)
- different processes (Apache config)

Clients are served by the UGR. They can browse/stat or be redirected for action. The architecture is multi/manycore friendly and uses a fast parallel caching scheme.
A WAN performance test

• Measuring the worst case performance is not easy
  – The cache caches also the intermediate actions, e.g. the directory lookups in a path

• Two storage endpoints: DESY and CERN (poor VM)

• One UGR federator at DESY, clients at CERN

• 10K files are interleaved in a 4-levels deep directory
  – Oddly-numbered files are at CERN
  – Evenly-numbered files are at Desy

• The test (written in C++) invokes Stat only once per file, using many parallel clients doing stat() at the maximum pace from 3 machines
  – Hence it’s supposed to be a worst-case, measuring the full roundtrip. In practice the cache has a role anyway, as the files share the path.
  – The raw speed of 2K stats/s per frontend machine is satisfying. Seems somehow capped by Apache, as the backend UGR is up to 100 times faster
WAN access (CERN-DESY)
News

• Very stable, installable from the wiki
• Survived very well any stress test we could do
• External demo in http://federation.desy.de/myfed
• Recent improvement: in the case we federate catalogues, the replicas they give can be checked on the fly.
  – Use the catalogue as name translator
  – Use the catalogue as source of file listings
  – Check the replicas in the moment they are requested
• Next item: ATLAS and Rucio
  – We have a nice testbed, federating many ATLAS SEs
  – We want to federate the Rucio services and the LFC(s) seamlessly together

• Power users wanted
  – Helping in getting the best out of the system. Your cooperation is very appreciated.
Get started

• Get it here:
  – https://svnweb.cern.ch/trac/lcgdm/wiki/Dynafeds

• What you can do with it:
  – Easy, direct job/user data access, WAN friendly
  – Friend sites can share storage
  – Storage-less sites, storage-only sites
  – Federating catalogues too
    • Combining catalogue-based and catalogue-free data
Conclusions

• Dynamic Federations: an efficient, persistency-free, scalable, easily manageable approach to federate remote storage and metadata endpoints
• Usable for fast changing caches and clouds
• Gives ways to solve some nasty Data Management problems
• Very generic, standards-based components. OK for HEP and other domains as well
• Status is stable, demoable, installable, documented
• https://svnweb.cern.ch/trac/lcgdm/wiki/Dynafeds
Thank you

• Questions?