Large Scale CVMFS and DynaFed

Alastair Dewhurst

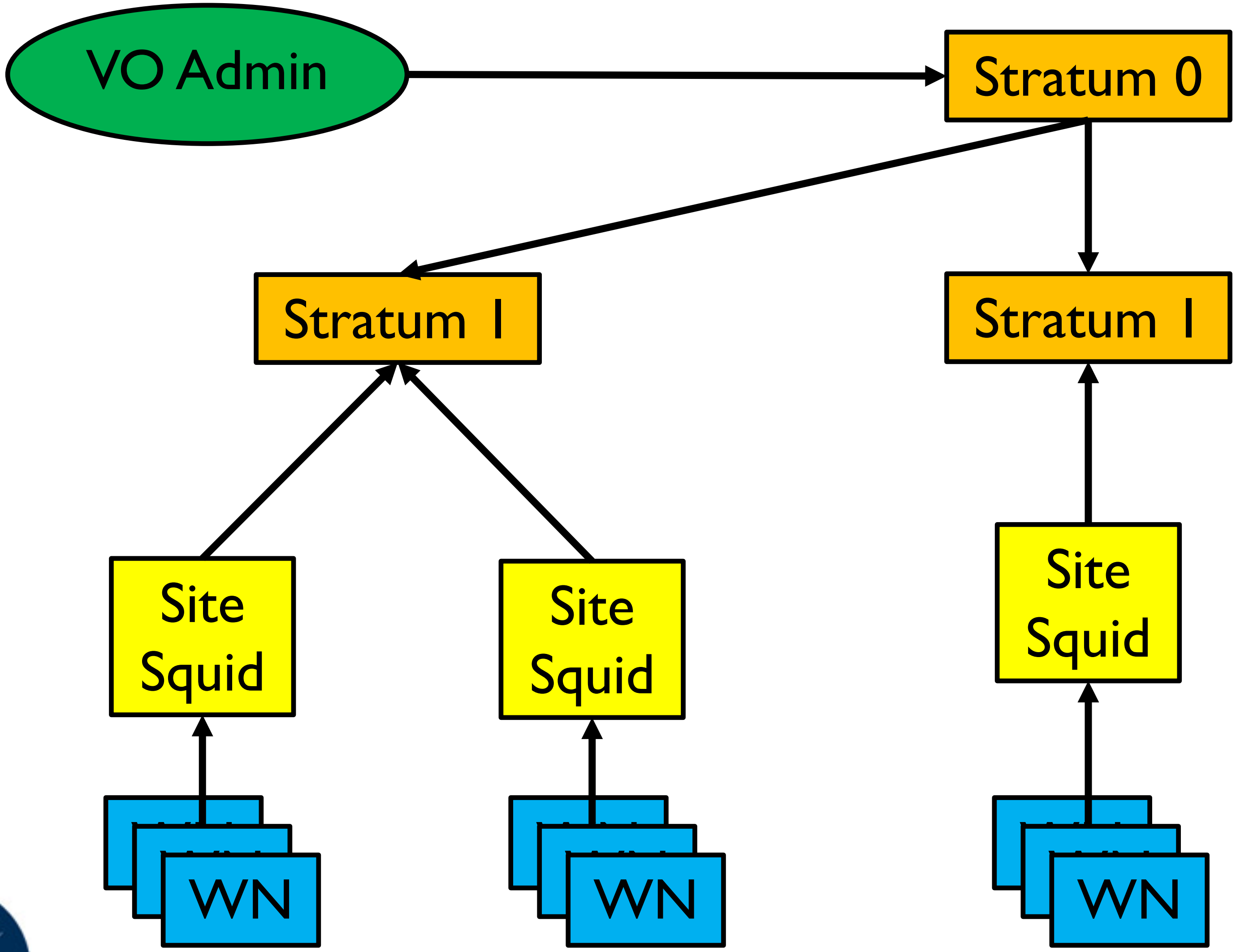
Introduction

- Effective Distributed Data Management has proven to be an extremely difficult problem to solve.
    - Large VOs have managed it but at significant effort costs.
- For small VOs it often limits where they can run their data analysis to a very small number of sites.
- Two problems:
    - Distributing data
    - Accessing data

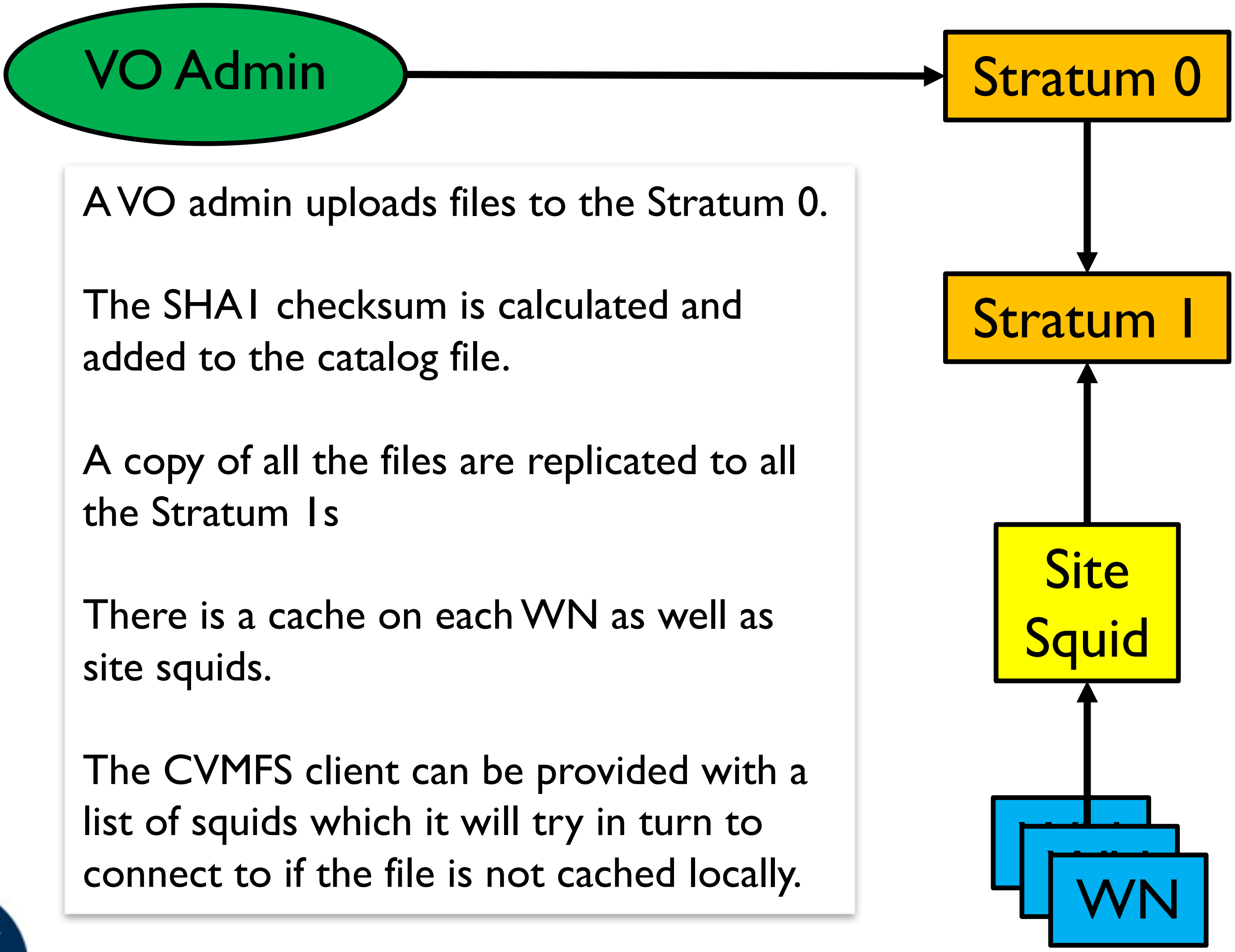
CVMFS

- CVMFS has been primarily developed for distributing large software stacks.
  - It has been very effective at allowing VOs to access their software!
- CVMFS provides POSIX like access to files which is what most users want.
- Large Scale CVMFS is an extension to the base software which allows it to distribute large, non-public datasets.
  - This requires secure CVMFS.
- Two types of setup have been tried so far:
  - Medium scale - Caching layer (via StashCace) can be used as workflows <1TB.
  - Large Scale - No Caching layer.

‘Traditional’ CVMFS



‘Traditional’ CVMFS



A VO admin uploads files to the Stratum 0.

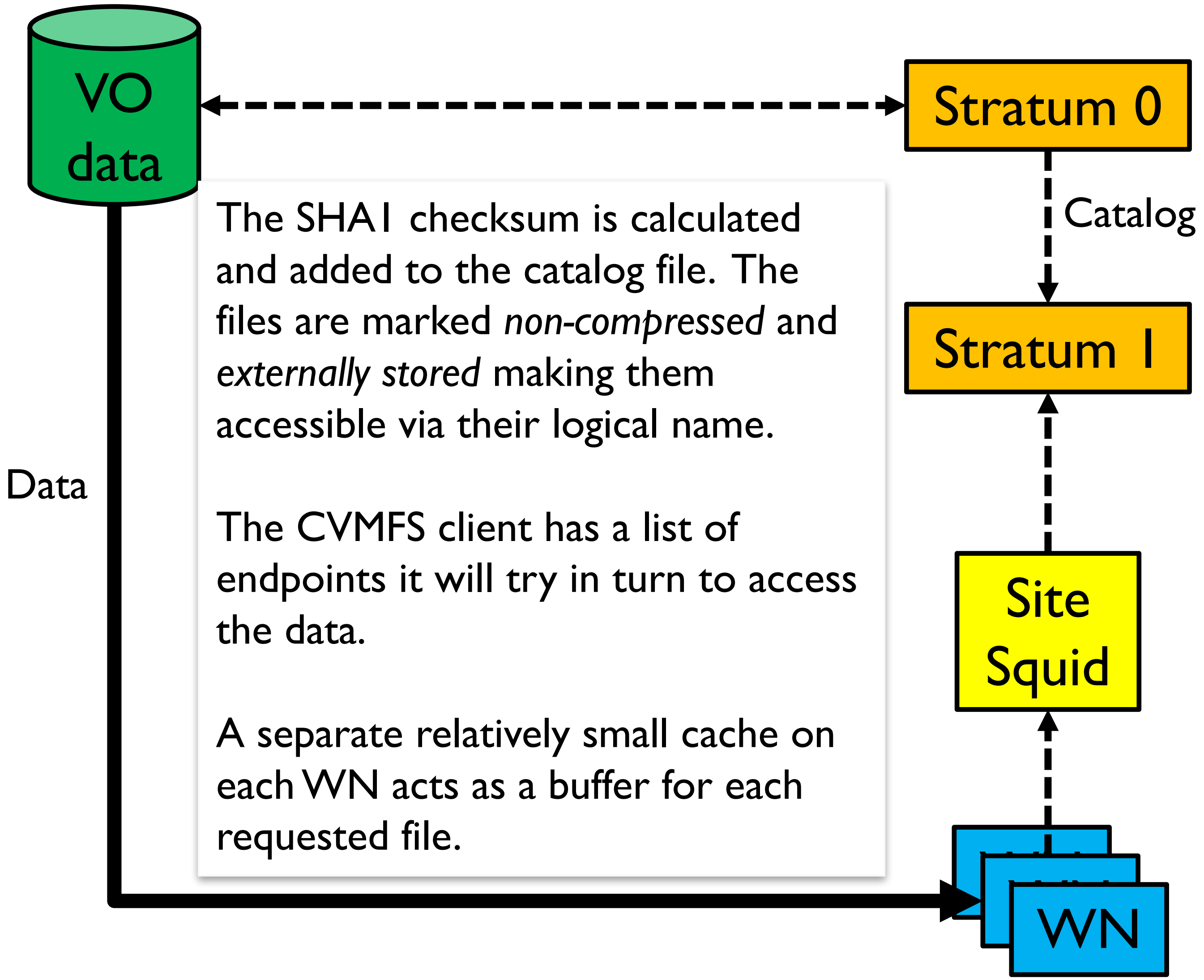
The SHA1 checksum is calculated and added to the catalog file.

A copy of all the files are replicated to all the Stratum 1s

There is a cache on each WN as well as site squids.

The CVMFS client can be provided with a list of squids which it will try in turn to connect to if the file is not cached locally.

Large Scale CVMFS



The SHA1 checksum is calculated and added to the catalog file. The files are marked *non-compressed* and *externally stored* making them accessible via their logical name.

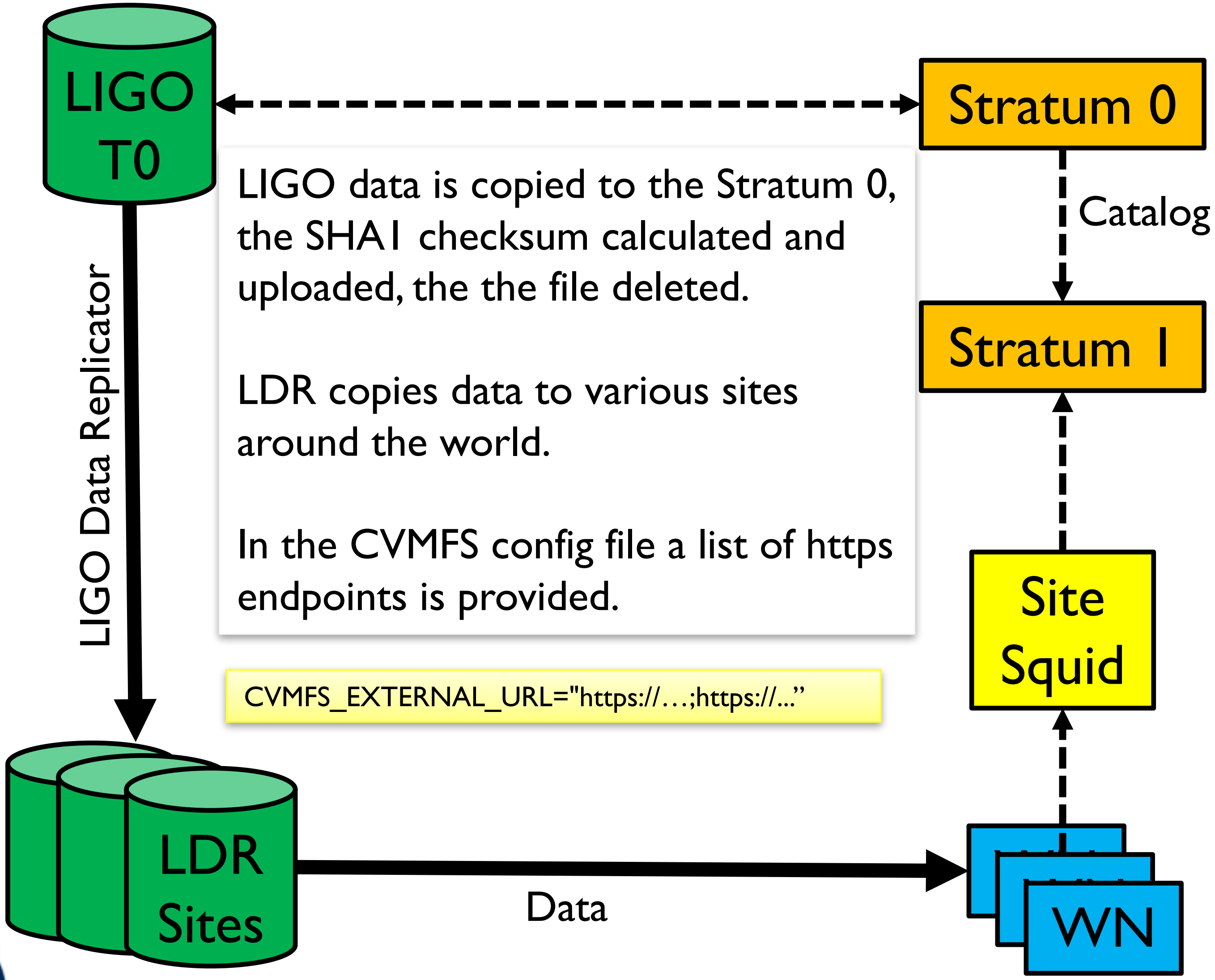
The CVMFS client has a list of endpoints it will try in turn to access the data.

A separate relatively small cache on each WN acts as a buffer for each requested file.

LIGO

- LIGO are a typical small VO in that they run a large fraction of their work on a few local batch systems.
  - Limited distributed computing infrastructure.
  - POSIX access to files is assumed.
- LIGO are not a typical small VO in that they won a Nobel prize for observing gravitational waves.
  - They have lots of people offering them help!
- LIGO use Large Scale CVMFS for some of their work on the Grid.
  - The PyCBC workflow which needs ~10TB data.
- LIGO were the VO that required secure CVMFS

LIGO setup



LIGO data is copied to the Stratum 0, the SHA1 checksum calculated and uploaded, the the file deleted.

LDR copies data to various sites around the world.

In the CVMFS config file a list of https endpoints is provided.

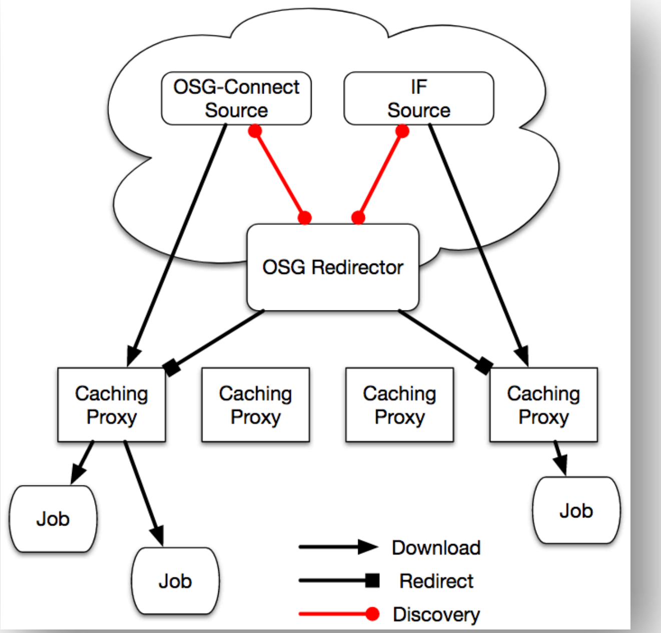
```
CVMFS_EXTERNAL_URL="https://…;https://...”
```

CVMFS-Sync

- Uploading files to a central repository just to be checksummed and deleted is not optimal (especially at scale).
- It is possible to “graft” files by creating a special file containing the necessary publication data.
    - When a graft is encountered, the file is published as if it was present on the repository machine.
    - To create these graft files you still need to calculate the checksum of the files.
- Brian Bockelman tried alternative methods such as submitting HTCondor jobs to WN to calculate the checksum of each file.
    - Worked but was operationally expensive.
- He has since extended the GridFTP server to store SHA-1 checksums.
- Brian’s work to sync repositories can be found here:
    - https://github.com/bbockelm/cvmfs-sync

StashCache

- StashCache is a service provided by OSG designed for individual users or small projects.
  - It allows them to move data to, and access it from Grid Sites.
  - Built on XRootD.
- StashCache is also used to store a replica of LIGO data to allow it to be accessed through large scale CVMFS.
  - Using http plugin for XRootD.

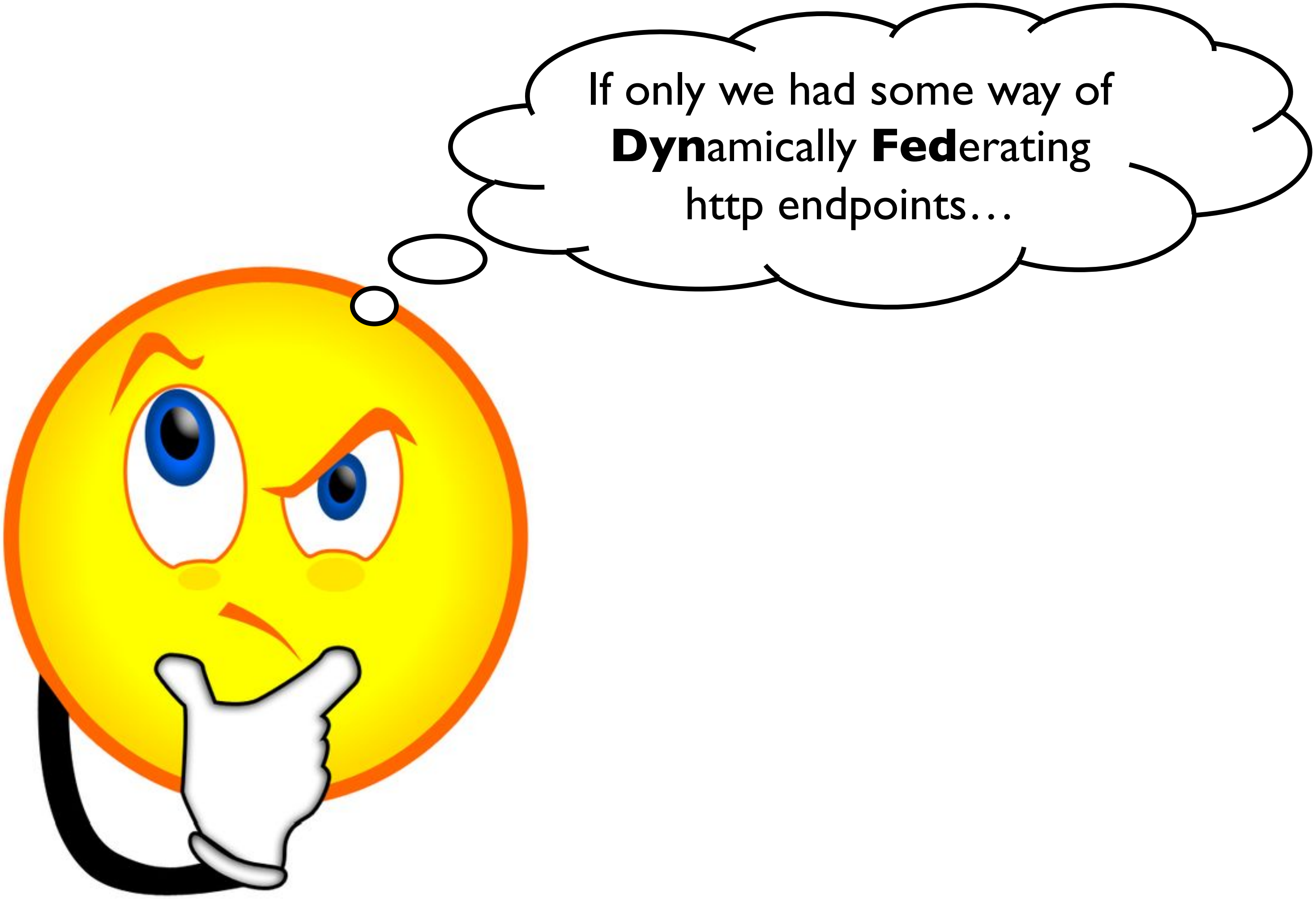


https://arxiv.org/pdf/1705.06202.pdf

- This works well although it is not the simplest setup (uses multiple protocols).

Is Caching Useful?

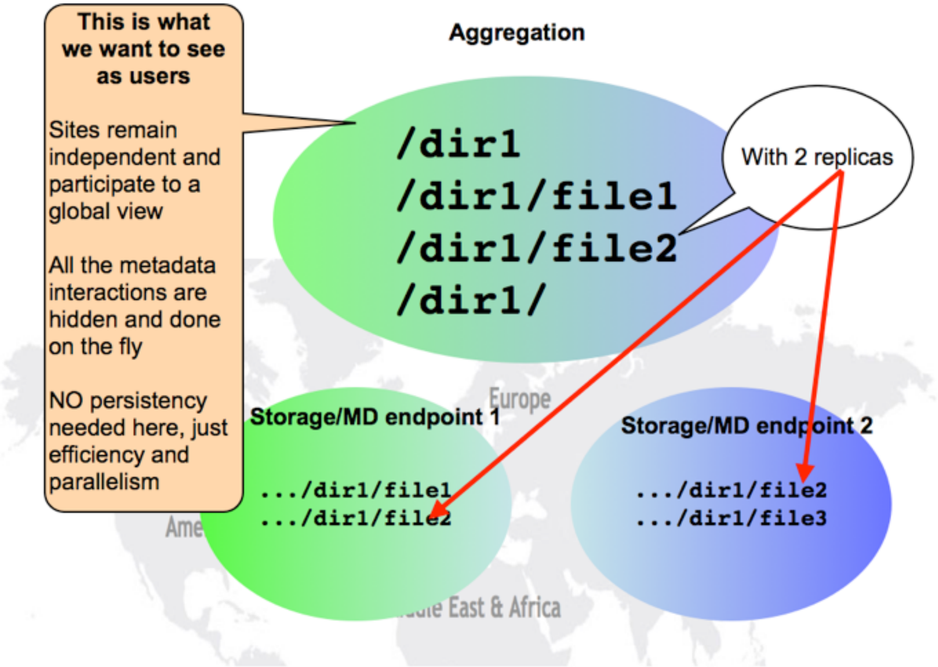
- For Caching to be useful:
  - The same data needs to be used multiple times.
  - The cache needs to be large enough to store this data so that popular data doesn't get pushed out before it has a chance to be used.
- For VOs like ATLAS/CMS most files are accessed once.
  - There have been some examples where caching actually slows things down.
- If you can make the cache large enough to get a significant number of hits, you can probably improve performance further by pre-placing data.



If only we had some way of **Dyn**amically **Fed**erating http endpoints…

DynaFed

- DynaFed aims to provide a “Dynamic HTTP storage federation”.
- DynaFed also provides access to Cloud storage.


This is what we want to see as users

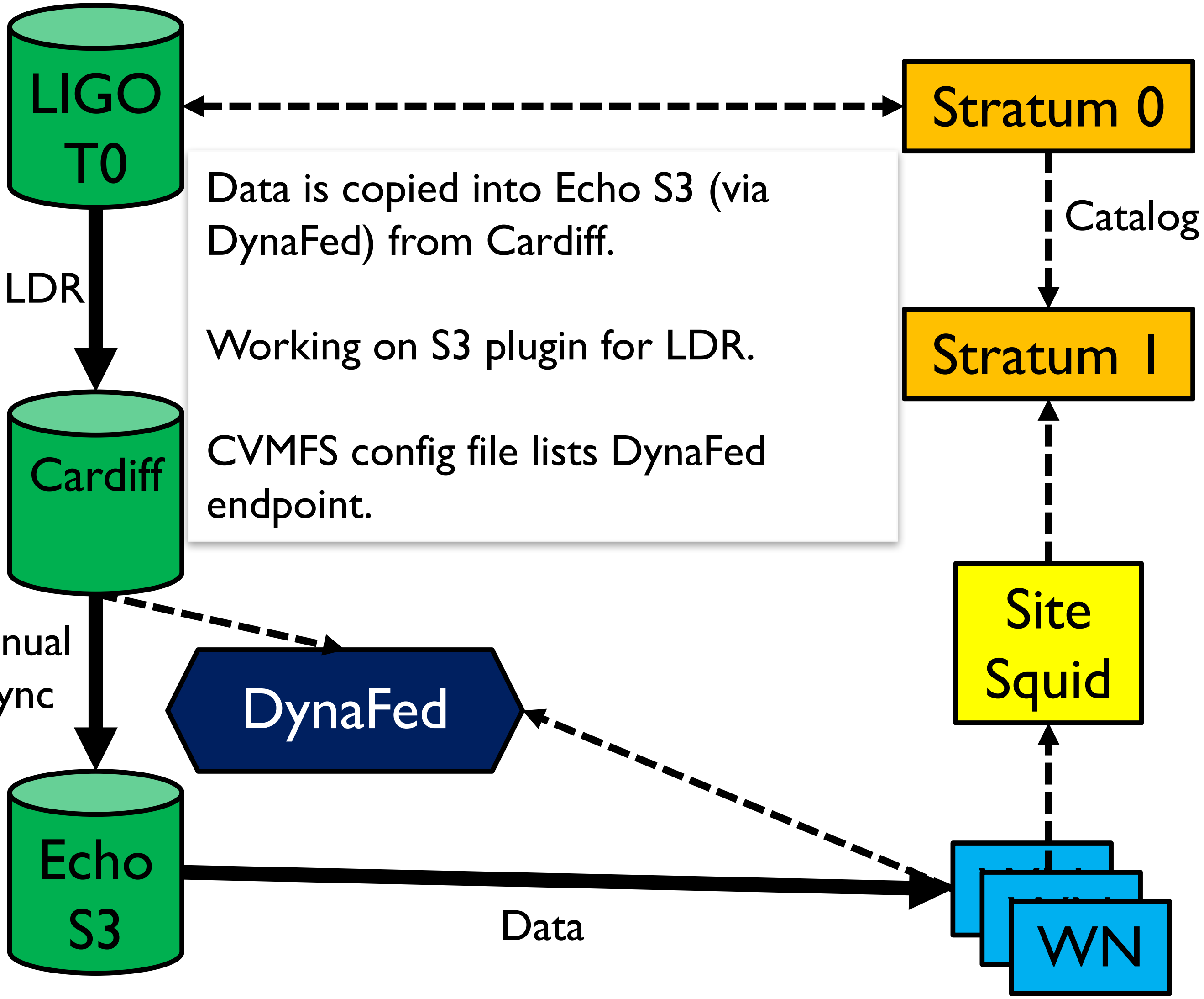
Sites remain independent and participate to a global view

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/
With 2 replicas
Storage/MD endpoint 1
.../dir1/file1
.../dir1/file2
Storage/MD endpoint 2
.../dir1/file2
.../dir1/file3
Europe
Ame
dle East & Africa

RAL/LIGO setup



Data is copied into Echo S3 (via DynaFed) from Cardiff.

Working on S3 plugin for LDR.

CVMFS config file lists DynaFed endpoint.

What does DynaFed add?

- Currently DynaFed is providing an authentication layer which allows large scale CVMFS to work with S3 storage.
- Main benefits come if we had multiple endpoints behind DynaFed:
  - Provides a single name space so that CVMFS doesn't have to try each endpoint in turn to access the data.
  - If data is available at multiple sites, it will select the closest.
  - Simplifies CVMFS configuration (which is needed on every client e.g. WN).
  - Makes it much easier to use opportunistic storage.

Limitations?

- There are no obvious architectural bottlenecks in using LS-CVMFS + DynaFed but it hasn't been tested at scale.
- LS-CVMFS + DynaFed provides a solution for the problem of how to access data but:
  - Can't upload files by just "cp" to a CVMFS directory.
  - Uploaded files need to be grafted before they are available.
    - Workflows that have the output of one job being the input of the next job would need to wait if relying on LS-CVMFS
- Data distribution is a separate problem.
  - Being able to send jobs to the site with the data will always be helpful.
  - No automated way of replication data in the federation to improve performance

Next Steps

- For LIGO:
  - Some other sites to setup Large Scale CVMFS and to enable LIGO jobs to be submitted.
  - Another site to allocate some storage space for LIGO which can join DynaFed.
    - Webdav support is required.
    - Test performance improvement from having multiple copies.
- For another (test) VO:
  - Test ways of grafting files to RAL Stratum-0.
- Aim to demonstrate that this could be an effective data access model for Non-LHC VOs in the UK.

Longer term

- Various people are looking at ways that DynaFed could store data written to it.
    - A plugin that could replicate popular data would be useful.
- If WebDav transfers could store SHA1 checksums in storage endpoints that would make file grafting significantly easier.
- The HEP Software Foundation is producing a Community White Paper that sets out a detailed research and development programme.
    - Details of a few relevant topics are in the next few slides.

HSF CWP (1)

Discover the role data placement optimisations can play, such as caching, in order to use computing resources effectively, and the technologies that can be used. The following tasks should be completed by 2020:

- Quantify the benefit of placement optimisation for the main use cases i.e. reconstruction, analysis, and simulation.

- Data federations have struggled in the past because of performance / reliability issues with data access over the WAN.
- LS-CVMFS is great for small VO because it provides POSIX like access to their data.
  - It doesn't matter if a small number of files are accessed inefficiently.
- Understanding the importance of data placement for various workflows is key to understanding how scalable LS-CVMFS would be.

HSF CWP (2)

In the longer term, it is planned to also study the benefits that can be derived from using different approaches to the way HEP is currently managing its data delivery systems. Two different content delivery methods will be studied, namely Content Delivery Networks (CDN) and Named Data Networking (NDN).

- Study how to minimise HEP infrastructure costs by exploiting varied quality of service from different storage technologies. In particular, study the role that opportunistic/tactical storage can play, as well as different archival storage solutions. A proof-of-concept should be made by 2020, with a full implementation to follow in the following years.

- This is what DynaFed is designed for:
  - Provides uniform access to both existing and new storage technologies which would help minimise costs.
  - Opportunistic storage can be dynamically federated and quickly provide performance improvements for jobs running at nearby resources.
    - If jobs were running on a cloud resource with low efficiency, data could be transferred in and even jobs that were “in flight” would be redirected to the local data as soon as it was available.

HSF CWP (3)

Contribute to the prototyping and evaluation of a quasi-interactive analysis facility that would offer a different model for physics analysis, but would also need to be integrated into the data and workload management of the experiments. This is work to be done in collaboration with the Data Analysis and Interpretation WG.

- "Quasi-interactive analysis" = POSIX please.
- For example, ATLAS could:
  - Graft all their DxAOD files.
  - A complete copy of this could be stored in a region (e.g. UK) which is federated into a single name space.
  - Users have interactive access to their data if they are working on a machine with LS-CVMFS installed.
  - Jobs submitted to the analysis facility would work as long as it had LS-CVMFS installed.

Conclusions

- LS-CVMFS provides POSIX like data access.
    - This is what users want.
    - It is working in production for LIGO.
- DynaFed provides functionality that allows LS-CVMFS to scale up allowing efficient access to data across many sites.
- No new services required for GridPP:
    - CVMFS + DynaFed already run at RAL.
    - Config changes are all that is required at Tier 2.
- Aligns well with strategic goals of GridPP and HEP community in general.

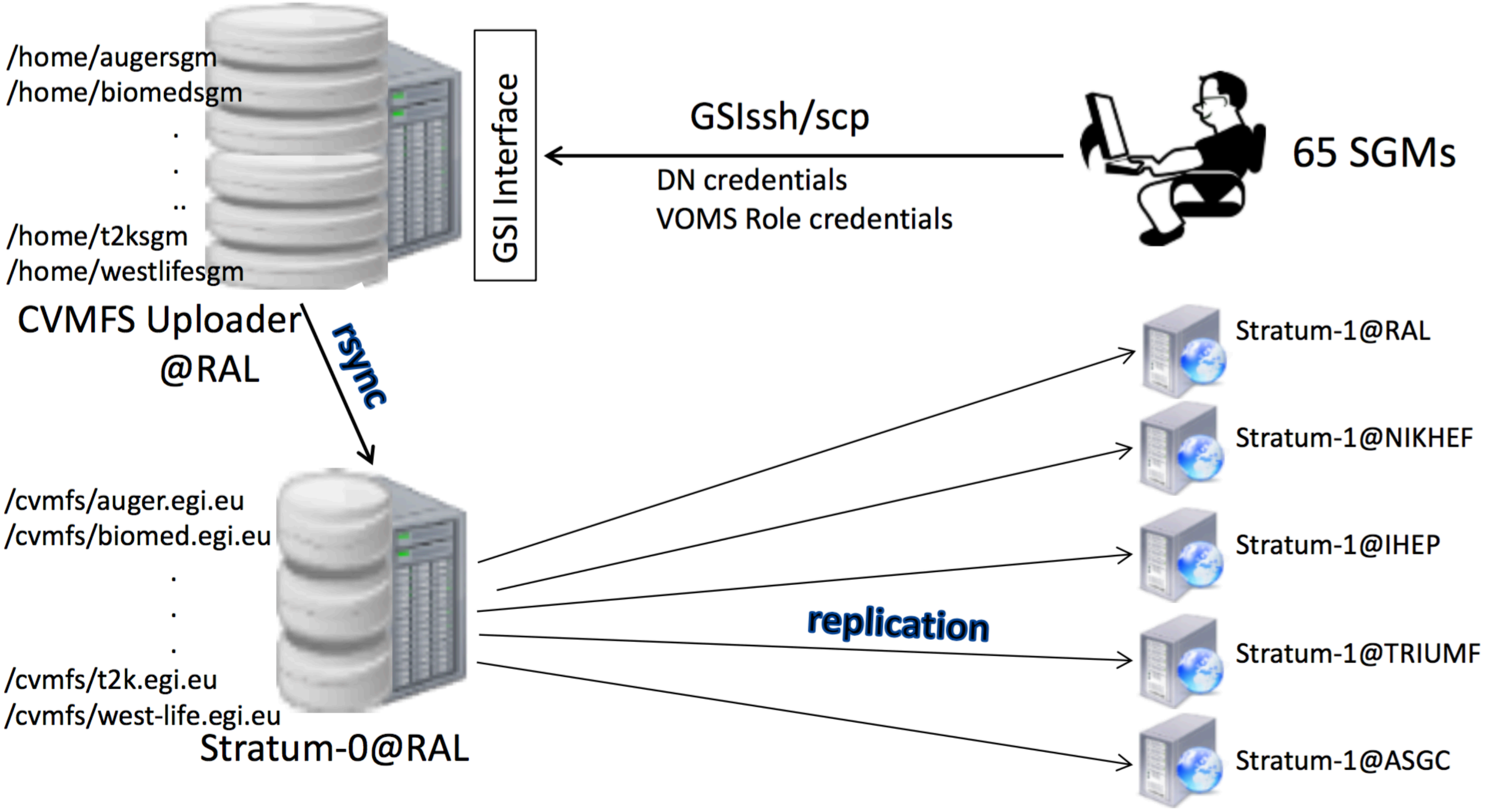
References

- Large Scale CVMFS documentation: http://cvmfs.readthedocs.io/en/stable/cpt-large-scale.html
- Accessing data federations through CVMFS (2017): https://drive.google.com/file/d/0B_RVv_OjWcURUi15cmtUaXotVkU/view
- CVMFS for data federations (2016): https://indico.fnal.gov/event/10571/session/7/contribution/34/material/slides/0.pdf
- DynaFed whitepaper (May 2017): http://svnweb.cern.ch/world/wsvn/lcgdm/ugr/trunk/doc/whitepaper/Doc_DynaFeds.pdf
- Data Access for LIGO on the OSG (May 2017): https://arxiv.org/pdf/1705.06202.pdf
- StashCache documentation: http://opensciencegrid.github.io/StashCache/
- Derek Weitzel's blog explaining StashCache: https://djw8605.github.io/2017/06/14/stashcache/
- “StashCache: Data Services for the OSG” (March 2015): https://indico.fnal.gov/event/8580/session/5/material/slides/0?contribId=47
- HEP Software Foundation Community White Paper: https://github.com/HEP-SF/documents/blob/master/CWP/papers/roadmap/HSF-Community-White-Paper-v0.2.pdf

Backup

RAL CVMFS setup

- Repository uploading mechanism



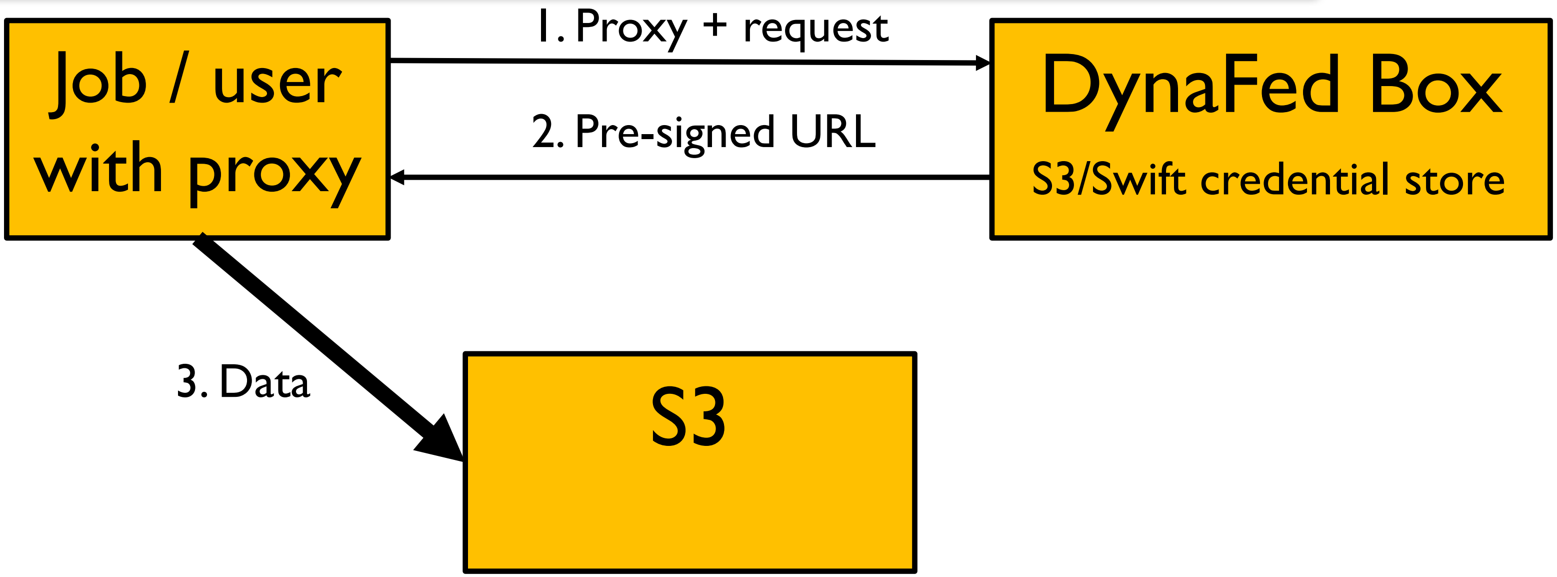
DynaFed

Directly to S3 endpoint:

```
davix-ls --s3alternate --s3secretkey XXXXX --s3accesskey YYYYY
        s3s://s3.echo.stfc.ac.uk/dynafed-test/
davix-put --s3alternate --s3secretkey XXXXX --s3accesskey YYYYY
        testfile s3s://s3.echo.stfc.ac.uk/dynafed-test/testfile
```

With Dynafed:

```
voms-proxy-init
davix-ls -P grid https://dynafed.stfc.ac.uk/gridpp/echo/
davix-put -P grid testfile https://dynafed.stfc.ac.uk/gridpp/echo/testfile
```


Job / user with proxy
1. Proxy + request
2. Pre-signed URL
DynaFed Box
S3/Swift credential store
3. Data
S3