

XRootD and Object Store: A new paradigm

Katy Ellis, Chris Brew, George Patargias, Tim Adye,
Rob Appleyard, Alastair Dewhurst, Ian Johnson

4th November 2019

Introduction to XRootD and Ceph at RAL

- Rutherford Appleton Laboratory (RAL) is the UK Tier 1
 - Supports all LHC experiments and growing number of others in HEP, Astronomy and Space
- RAL disk storage is known as Echo
 - Based on Erasure Coded Ceph Object Store
 - University of Glasgow is currently setting up a similar Ceph cluster
- Access to Echo is primarily via XRootD
 - The challenge is to optimize access for the different use cases
 - Gridftp plugin, but intention is to phase out

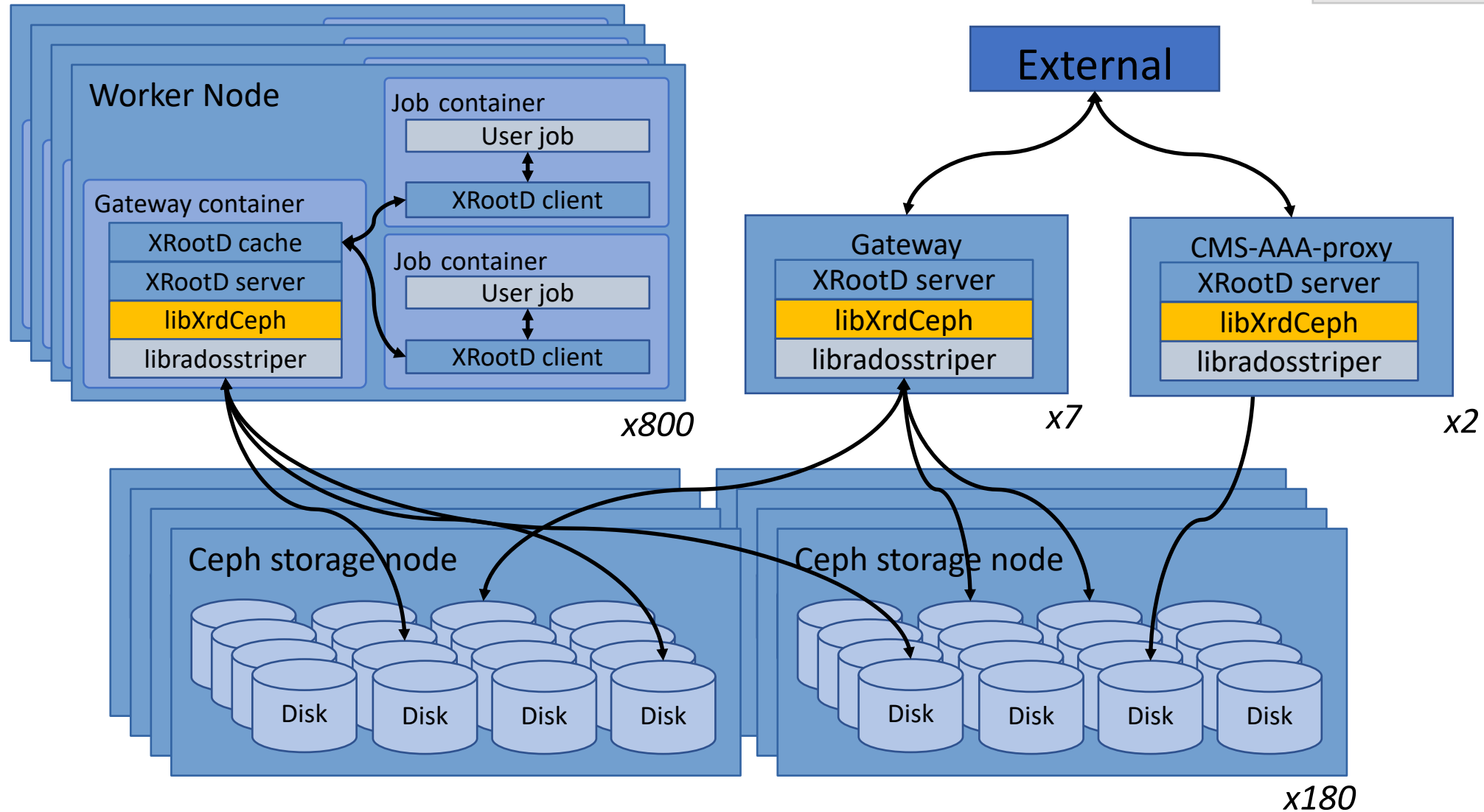


Contents

- Description of disk storage and XRootD at RAL
- XRootD data access efficiency
 - CMS-AAA (remote data access system)
 - Job access
- XRootD TPC
 - Object-store particulars
 - Delegation
- XRootD authentication
 - ALICE configuration

XRootD interactions with Echo

↔ data flow

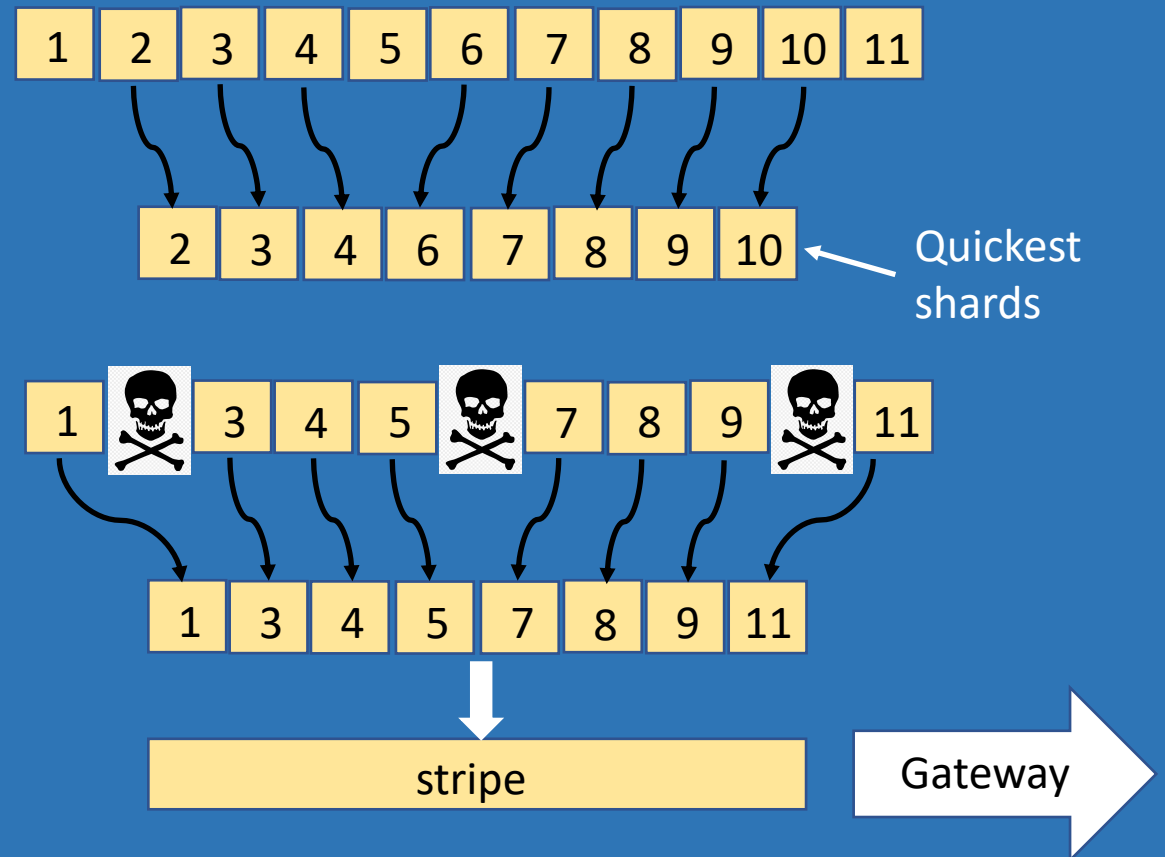


Erasure coding

- Divide each file into 64 MB 'stripes'
 - Then split into 8 x 8 MB objects called 'shards'
 - An additional 3 shards are calculated for redundancy
 - The 11 shards are stored on 11 different servers
 - Any 3 out of 11 shards can be missing or corrupted without data-loss

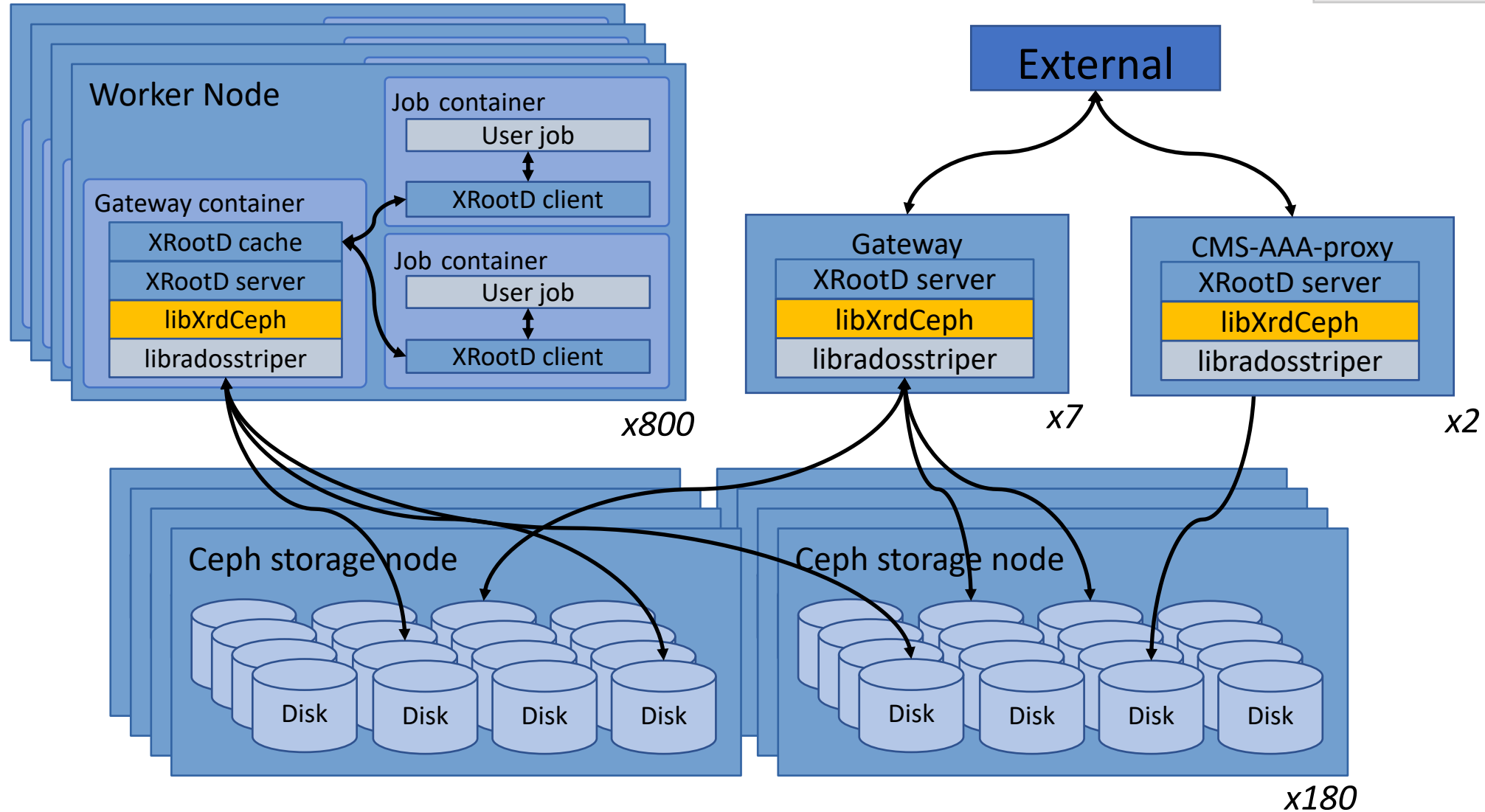
Reconstruction of a stripe

- Divide each file into 64 MB 'stripes'
 - Then split into 8 x 8 MB objects called 'shards'
 - An additional 3 shards are calculated for redundancy
 - The 11 shards are stored on 11 different servers
 - Any 3 out of 11 shards can be missing or corrupted without data-loss



XRootD interactions with Echo

↔ data flow

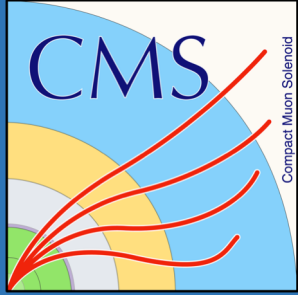


XRootD cache current setup

- Disk caches on the Worker Nodes
- Memory caches on the external Gateways
- No cache on the CMS AAA

CMS jobs and Echo

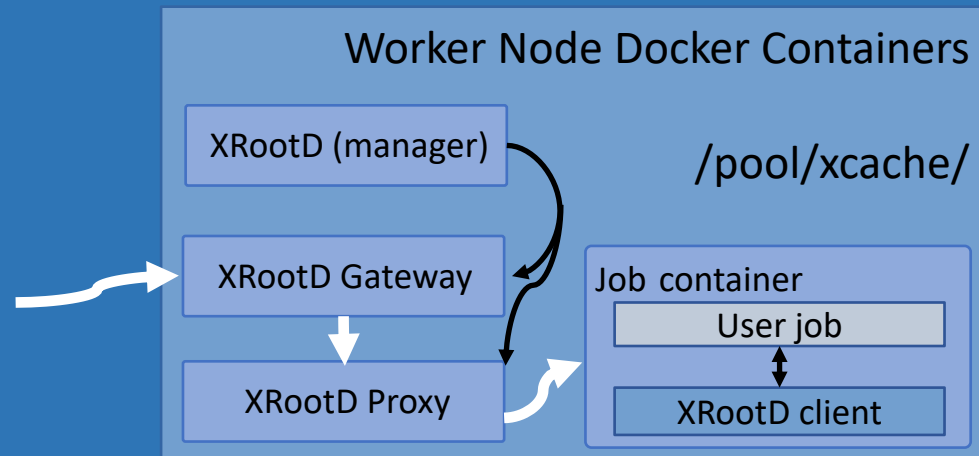
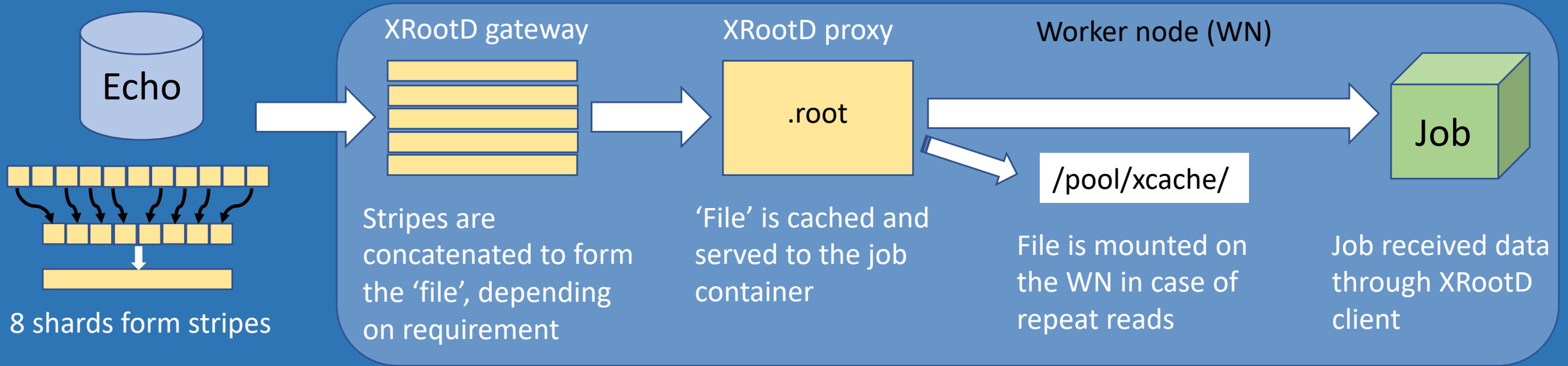
- Goal – learn about and improve the I/O for jobs run at RAL



CMS jobs and Echo – test setup

- Goal – learn about and improve the I/O for jobs run at RAL
- RAL worker node, drained and isolated from the farm
 - One job at a time
 - Using the full containerized setup
 - Submission via `condor_submit`
- Various types of CMS jobs, previously run at RAL
 - Chose to use an analysis job written by RAL colleagues for the main test
- Adapted site configuration file
- (CMS jobs have an added option of “cache-hint”)

CMS jobs and Echo



- Tested different values of 'cache—hint'
- These affect the way data is transferred to and stored in the *job container*
 - <application-only>, <lazy-download>, <storage-only>, <auto-detect>

CMS jobs and cache-hint

- Analysis job – making NTuple from 4 .root files with total size ~9 GB.
- ~10 tests for each cache-hint value.

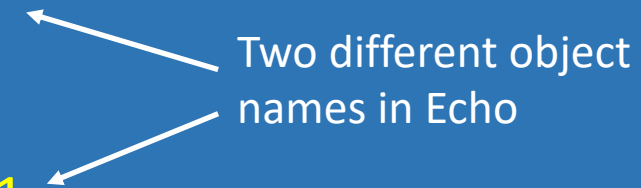
CMS jobs and cache-hint

- ~10 tests for each cache-hint value.
- Analysis job – making NTuple from 4 .root files with total size ~9 GB.

Cache-hint	Mean total (min:sec)	Highest - lowest	NetworkInput Mb
Application-only	32:11	1:45	~6900
Lazy-download	36:04	1:25	9173
Storage-only	35:15	1:46	~6680
Auto-detect	32:13	1:47	~6900

- Fairly tight distributions.
- ‘User’ time very similar for all tests except slightly greater for Storage-only.
- Auto-detect appears to choose Application-only.
- Lazy-download (used currently at RAL) is slowest.
- However...this is one job (type).
- What about when multiple jobs running simultaneously?
- Reliability is a concern.

XRootD Third Party Copy (TPC) and Echo

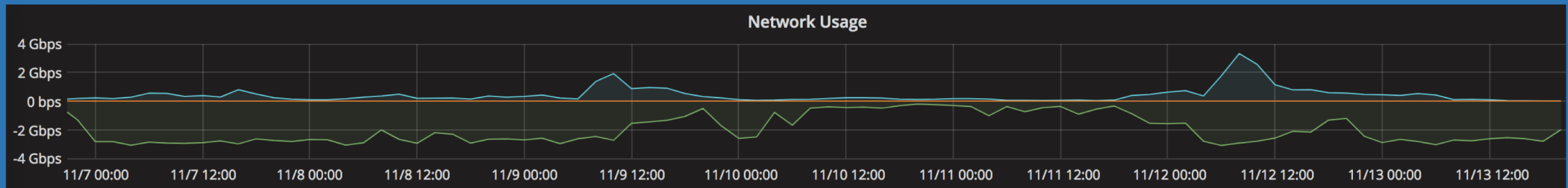
- As Echo is an object store, the path name has to be specified exactly
 - For example, if this 'path' was used to write or read
 - `root://xrootd.echo.stfc.ac.uk:1094/dteam:test1/testKaty1`
 - XRootD can replace the slash with a double slash
 - `root://xrootd.echo.stfc.ac.uk:1094//dteam:test1/testKaty1`
 - To accept either path, and not rely on other sites to provide only pathnames with single slashes, a workaround was applied at RAL
 - Define additional paths with extra slashes in the authdb and then translate them back in the TFC (Trivial File Catalogue)
- 

XRootD Third Party Copy (TPC) and Echo

- Concerning authentication, this setup requires the use of delegation of the command issuer's credentials to the destination server, e.g.
 - `xrdcp --tpc delegate only root://xrootd.echo.stfc.ac.uk:1094/dteam:test1/testSourceFile root://griddev03.slac.stanford.edu:2094//xrootd/atlas/tpctest/testDestFile [140B/140B][100%][=====][35B/s]`

CMS AAA and Echo

- CMS AAA is a service to deliver “Any data, any time, anywhere” to jobs running at any CMS site.
- Partial reads are common in CMS but not efficient for object store – Echo works best when serving whole stripes (64 MB section of a file).
➡ Much more data into proxy than out!

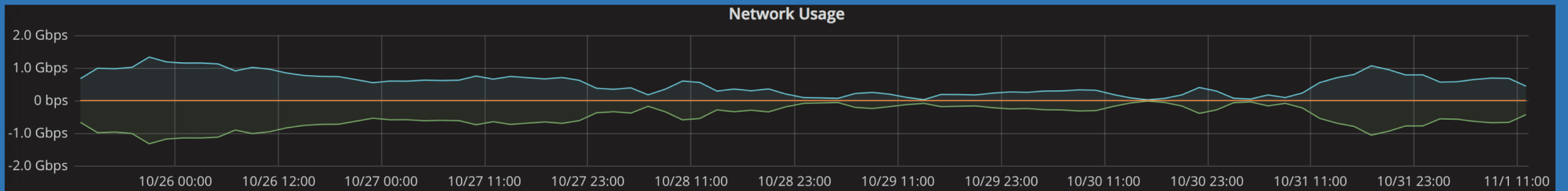
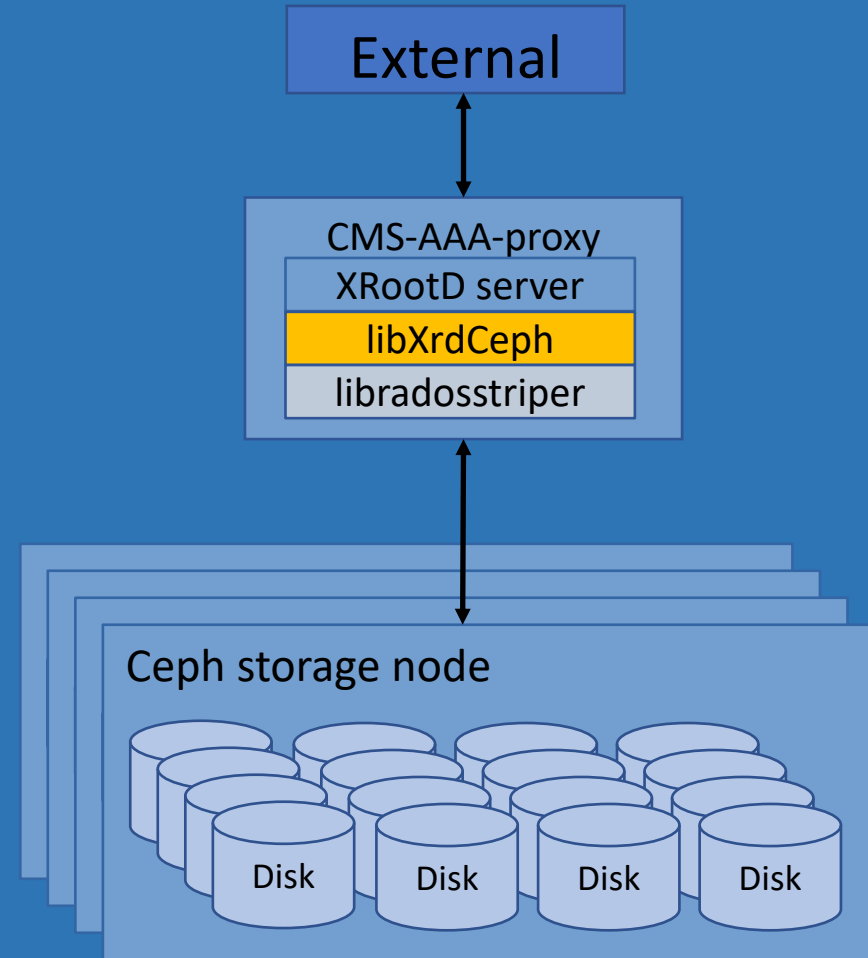


- Unclear whether (disk) caching is helpful – we believe the XRootD cache was demanding whole file, even if data from only a small number of stripes was requested.
- As a result – overloaded service that required frequent intervention.

CMS AAA and Echo

- AAA proxy cache was turned off.
- More recent CMS files now store the metadata in one place near the start or end of files, instead of dispersed throughout.

➔ Service is now much more stable and rarely requires intervention

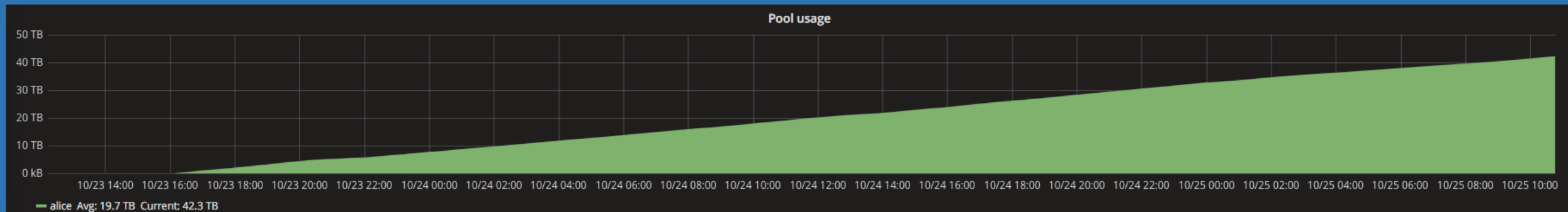


ALICE and Echo



ALICE

- ALICE require their own XRootD authentication plugin
- Blocker for ALICE to move from CASTOR onto Echo
- The source code was modified to accommodate the pool name alice:/
- ALICE transfers are currently somewhat isolated from other VOs
 - Have their own Echo gateways
 - Have their own XRootD alias: `alice.echo.stfc.ac.uk`



Summary

- Echo is working effectively with XRootD
 - Looking for continuous improvement
- All four LHC experiments are up and running
 - Each saw their own individual issues
- Non-LHC experiments are also, or will soon, use Echo for disk storage
- Thanks to the XRootD developers for their collaboration!

Backup

CMS jobs and Echo – XRootD proxy config

- Located on the WN : `/etc/xrootd/xrootd-ceph.cfg`
- Matches the Echo stripe size:
 - `xrootd.async segsize 67108864`
 - `xrd.buffer maxbsz 67108864`
- `pfc = proxy file cache`
 - `pfc.ram 7g`
 - `pfc.blocksize 16m`

Cache-hint options

- `application-only`

- This is the default and means **ROOT will do the caching**. If `PoolSource.cacheSize` is non-zero, a `TTreeCache` of that size will be created per open file. Asynchronous read-ahead will be turned off and the cache will be filled with normal reads.

- `storage-only`

- Means ROOT will drive the caching using a prefetch list, but will not allocate a cache of its own. If `PoolSource.cacheSize` is non-zero, a `TTreeCache` with a read-list of that size will be created, but no actual cache buffer -- the ROOT cache will be "virtual" and could in fact be very large. ROOT will hand over the prefetch list to the storage layer, which is expected to do its own caching. This method only makes sense if the underlying storage binding is capable of prefetching, which is currently true for local files (and anything downloading into a local file, such as `srm`, `storm`, `gsiftp`) and `RFIO`. Using this method with an incompatible storage system such as `dCache` will trigger an error.

Cache-hint options

- `lazy-download`
 - Means **remote files will be downloaded to a local shadow file on demand in 128MB segments**. ROOT reads will be directed to this local file; ROOT will never read directly from the remote file. If PoolSource specifies a non-zero cache, it will behave as a "storage-only" virtual / prefetch cache. Note that the file will be downloaded lazily even if PoolSource.cacheSize is zero. The local shadow file will be created in the specified temporary directory and will be removed automatically when the corresponding remote file is closed. If no suitable local temporary directory with sufficient free space can be found, lazy download is automatically switched off.
- `auto-detect`
 - This tells the I/O layer to pick the best strategy suited for the I/O technology in use. This will be "lazy-download" for RFI/O, dCache and the "file" protocol, including any method which downloads remote files to local disk.

Changes required for TPC // problem

authdb:

VO = Atlas

u atlasprod \atlas:datadisk/ a \atlas:scratchdisk/ a /atlas:datadisk/ a /atlas:scratchdisk/ a

u atlasuser \atlas:datadisk/ r \atlas:scratchdisk/ a /atlas:datadisk/ r /atlas:scratchdisk/ a

New xrootd storage.xml

```
<lfn-to-pfn protocol="direct" path-match="/*(*)" result="$1"/>
```


CMS jobs and Echo

