Science & Technology Facilities Council

UK Research and Innovation

XRootD access for CMS jobs at RAL

Katy Ellis, 11th June 2019, CCIN2P3

Why is this interesting?

- RAL disk is an Erasure-coded object store, called Echo
  - Highly parallel and scalable, becoming the norm in industry
  - However, sparse reads are inefficient
  - Storage via object store is unique amongst T1s
- Several changes made to make Echo fit the CMS model
  - Can further optimization be done?
- Gsiftp is being phased out
  - Assumed that XRootD will take over
  - N.B. XRootD is currently used for reading *from* Echo but not writing *to* Echo

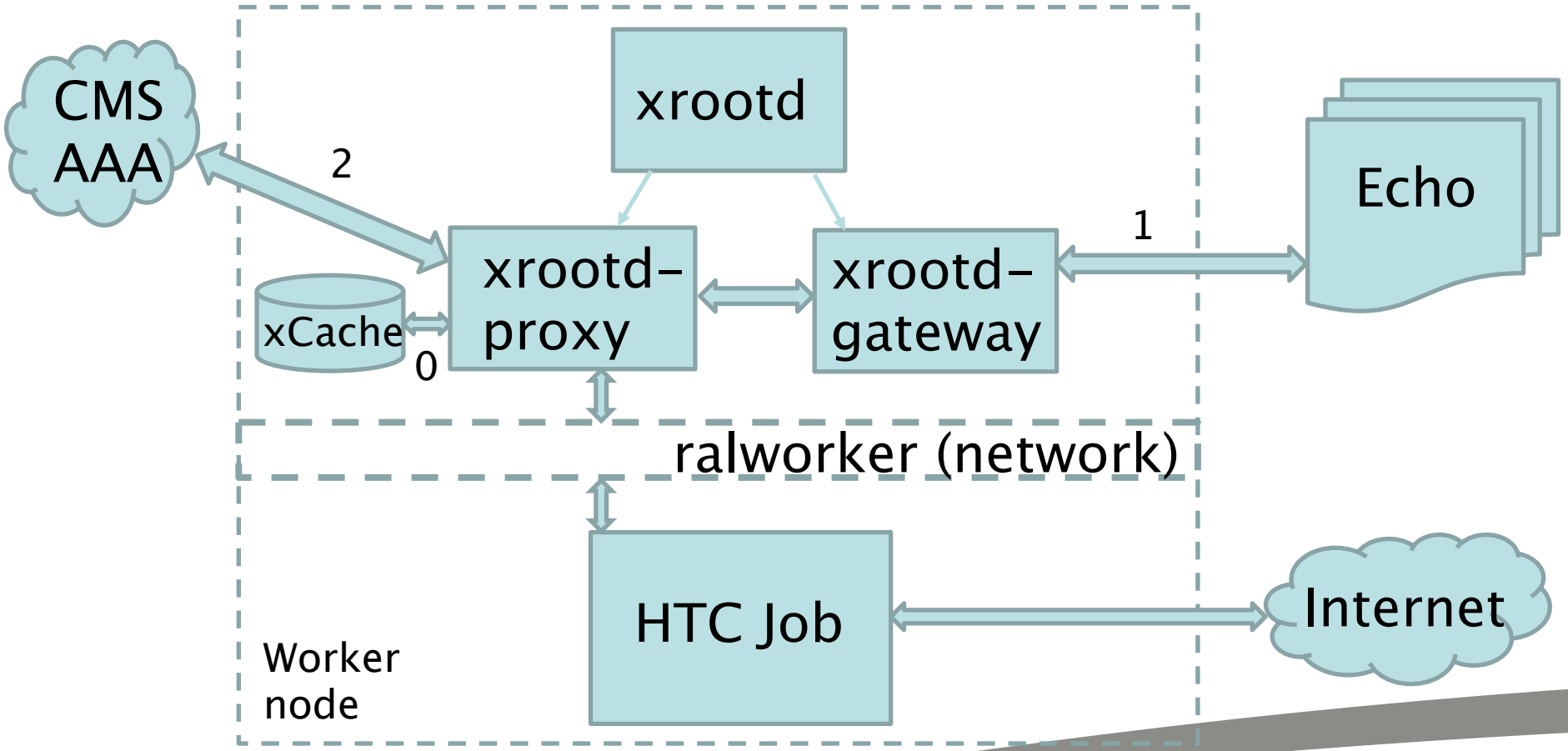
Objectives

- Develop knowledge of Production system
  - Understand how CMS accesses data
  - Understand the setup of worker nodes at RAL
  - Understand XRootD configuration and relevant parameters
- Improve efficiency of CMS jobs running at RAL
  - Particularly with regard the I/O from Echo storage
  - Can XRootD contribute?
    - E.g. size of buffers/caches

CMS data access

- Data access by CMS differs significantly from e.g. ATLAS
  - ATLAS downloads an entire file to the WN
    - Short-lived, high-rate connections
    - Total amount downloaded is high, but…
    - This works well with object stores!
  - CMS accesses sections of the file it needs, as and when it is needed, over an open network connection
    - Long-lived, low-rate connections
    - Total amount downloaded is lower, however…
    - This is non-optimal for object stores
- Same gateways / same XRootD config required for both

Worker node setup at RAL



Problems/changes to date

- CMS-AAA XRootD was unstable
  - Original config had pagesize matched to Echo stripe size (“for efficiency”).
  - Monitoring showed that data in >> data out
  - Couldn’t support required number of concurrent accesses
- XRootD config changes were made
  - Pagesize was reduced
  - Max2cache was reduced
  - Throttling introduced

XRootD changes for AAA

- CMS-AAA proxy machine, changes to xrootd-proxy.config

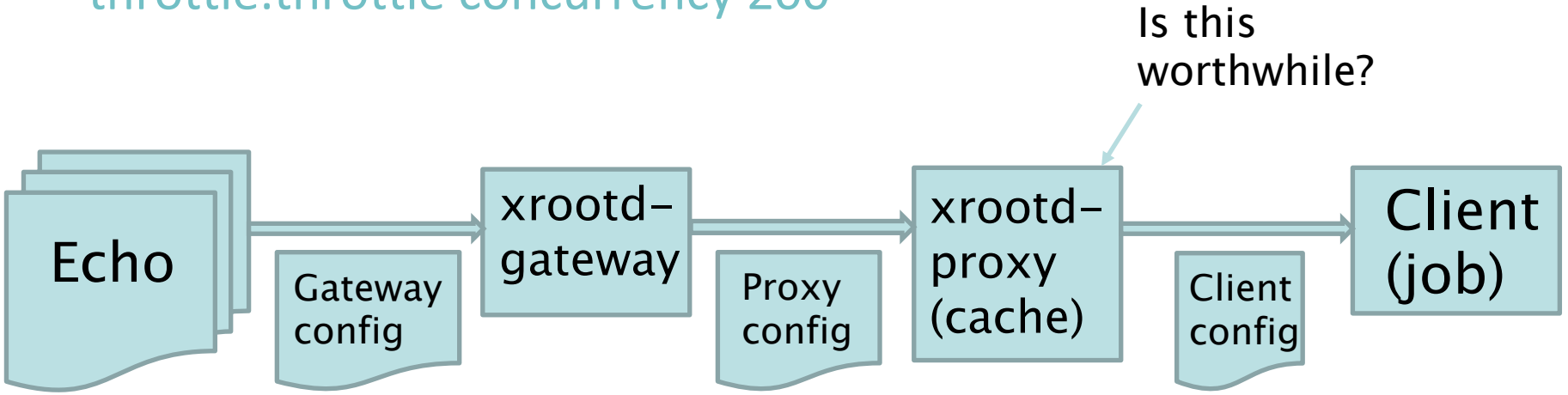
xrootd.async segsize 67108864

xrd.buffers maxbsz 67108864

} Matches stripe size from Echo

pss.cache max2cache 4m pagesize 4m size 48g

throttle.throttle concurrency 200



Problems/changes to date 2

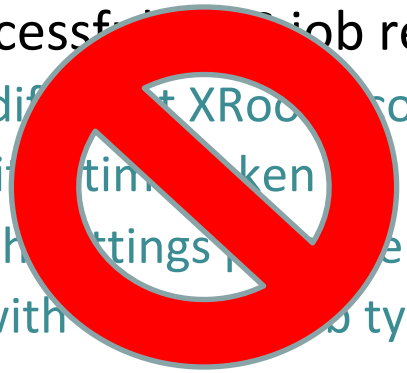
- CMS jobs doing an XRootD Vector Read for event metadata scattered across the file
  - Triggered the proxy to read the *entire* file before returning with small amounts of data
  - For large files, the client timed out
- No solution found, but masked by using 'Lazy Download' feature
- 'Recent' CMS files consolidate metadata in one part of the file
  - Old files still have dispersed metadata and always will

CMS jobs efficiency: Plan 1

- Run successful CMS job repeatedly on RAL worker node
  - Use different XRootD config settings
  - Monitor time taken
  - Which settings produce best performance?
  - Try with different job types – using varying I/O

CMS jobs efficiency: Plan 1

- Run successf[illegible]ob repeatedly on RAL worker node
  - Use di[illegible]t XRoo[illegible]onfig settings
  - Moni[illegible]tim[illegible]ken
  - Which [illegible]ttings p[illegible]e best performance?
  - Try with [illegible]b types – using varying I/O



- Stupid plan!
  - Typical high I/O job took ~7 hours to run
  - Can still fail
  - May contain several areas of potential efficiency improvement not directly related to Plan 1.

Plan 2 (keep it simple at first!)

- Copy file repeatedly to worker node using xrdcp
  - Monitor time taken
  - Change XRootD config settings
  - Which settings produce quickest copy?
- N.B. this does not represent a CMS job.

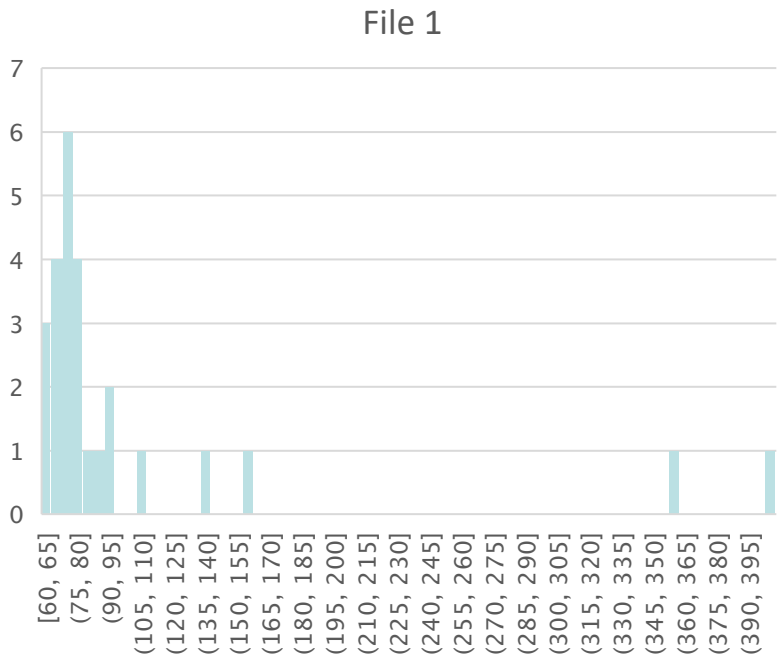
Plan 2 (keep it simple at first!)

- Copy file repeatedly to worker node using xrdcp
  - Monitor time taken
  - Change XRootD config settings
  - Which settings produce quickest copy?

?!

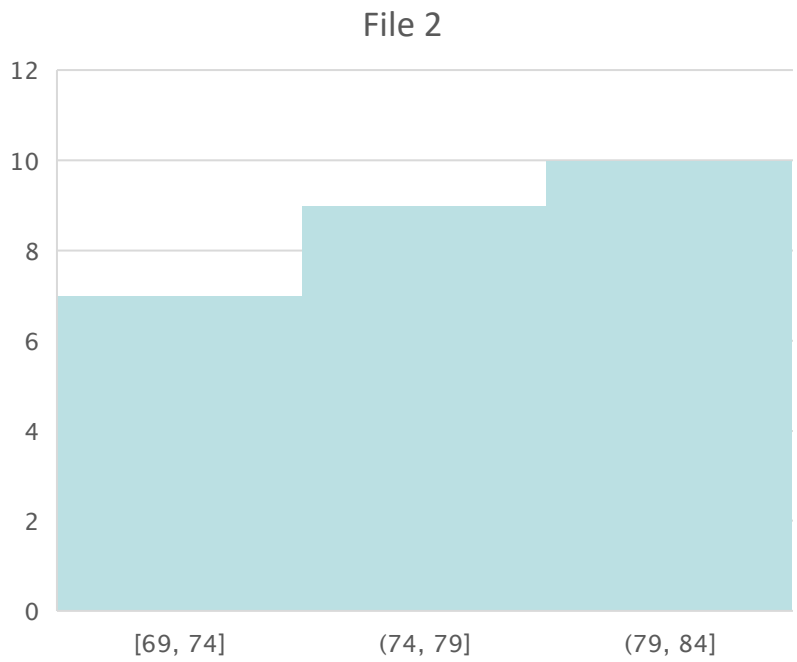
- Plan thwarted!
  - Repeated xrdcp produced a much wider than expected set of times
    - Despite using the exact same file – deleted each time and removed from /pool/xcache
    - Otherwise empty worker node

Variable response time



- Copy time over 26 attempts was between 60 seconds and 400 seconds
- For those taking longest I observed a hang, with the file ‘stuck’ at approx. 67% complete
  - In the meantime I saw the file continue to cache.

Variable response time 2



- Different file, same size ~4GB
- Much less wide spread of xrdcp times
- However, still quite variable
- May require a large number of measurements to test the effect of changes

Possible WN gateway XRootD config changes?

- Disk caching proxy (current)
- Disk caching block proxy?
- Memory caching proxy?
- No proxy??

Summary

- XRootD is being used to read from Echo disk storage for jobs at RAL.
- CMS access is not via straight xrdcp…but the XRootD settings need to be the same as for ATLAS data access .
- Changes have been made for AAA, some by intelligent guesswork.
- Lazy Download is being used.
- Even a straightforward timing test with xrdcp to a WN is not straightforward.
- There may be better ways to improve efficiency of CMS jobs at RAL before attempting to change XRootD parameters.
- Any advice is welcome.