Science and Technology Facilities Council

# ECHO: Experiences and developments of the RAL-LCG2 Tier-1 object store in run-3 and preparing for HL-LHC

CHEP 2023, Norfolk, Virginia

James Walder, RAL

*Covering inputs and activities from (non exhaustive):*

*Tom Byrne, Alastair Dewhurst,*
*Ian Johnson , Alex Rogovskiy,*
*Steven Simpson, Jyothish Thomas*

# Tier-1 Data centre at RAL

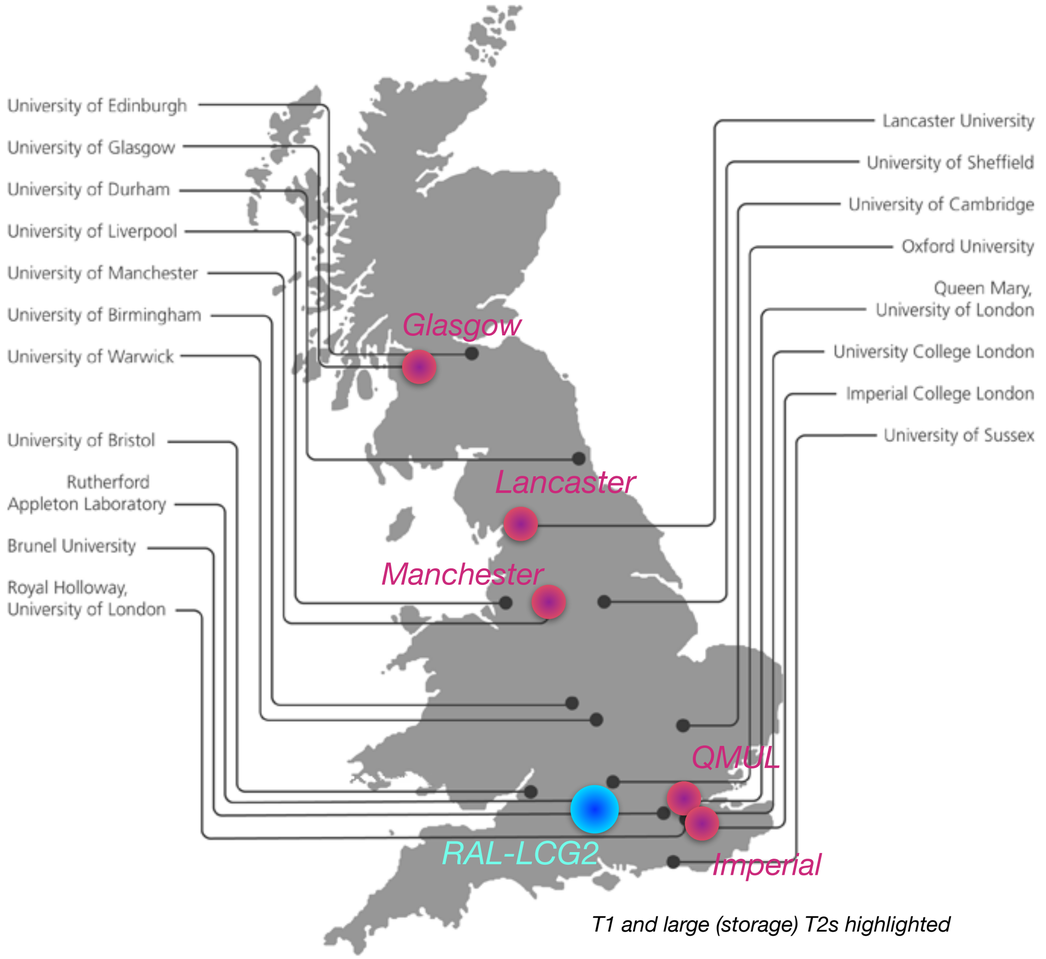
- Rutherford Appleton Laboratory based in south Oxfordshire, UK
  - Runs the UK Tier-1, supporting all LHC experiments,
  - and  an increasing number of small and larger VOs from other PP, PA, space and astronomy communities

- With recent upgrade to a 15-frames, currently is also housing the largest Tfinity library of tape storage in the UK and Europe, with 440PB capacity available.

# Tier-1 Data centre at RAL

- Rutherford Appleton Laboratory based in south Oxfordshire, UK
  - Runs the UK Tier-1, supporting all LHC experiments,
  - and  an increasing number of small and larger VOs from other PP, PA, space and astronomy communities

- Together with the other Tier-2 and Tier-3 sites in the UK
  - provides the Compute, Storage and
  - person expertise to deliver its MoU commitments to WLCG, and support non-LHC experiments
  - Managed under the GridPP project, within STFC and UKRI.




*T1 and large (storage) T2s highlighted*

# ECHO @ RAL-LCG2

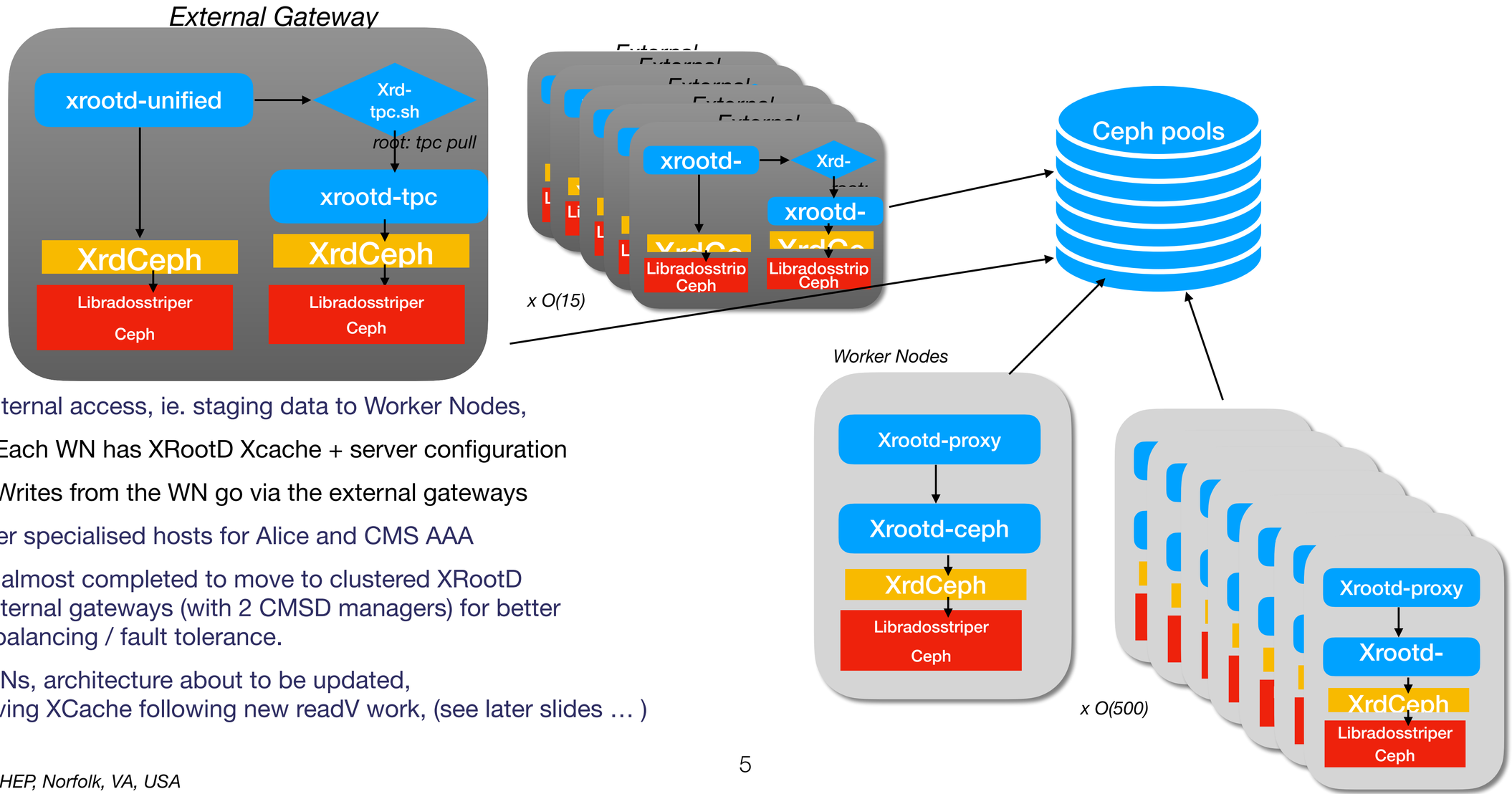
- ECHO: Ceph-based (RADOS) object store with data access provided through XRootD:
  - XrdCeph OSS plugin – originally developed by S. Ponce (CERN) using ceph's libradostriper
    - Provides the interface between XRootD and ceph at the OSS layer
  - Also deployed for UK Tier-2 site: Glasgow for ATLAS

- Over 50PB raw storage (+ 30PB with upcoming deployment).
  - 8+3 Erasure Coding
- Currently ~ 240 Storage Nodes (SN), with ~ 5000 OSDs
  - Host level failure domain (i.e. OSDs from placement group placed across different SNs).

- New hardware being deployed with uniform rack layouts;
  - 2 service nodes (e.g. XRootD Gateway, Ceph Mon)
    + several storage nodes per rack, with ToR routers.
  - May facilitate future move to rack-level domain failure mode
  - Nautilus + Centos7 (upgrade planning in progress)
- RAL also provides CephFS, S3 and SWIFT endpoints, etc.

# ECHO: Data access architecture

- External Access (e.g. via FTS) to ECHO provided via XRootD server/gateway hosts:
  - Currently each gateways behind round-robin DNS.

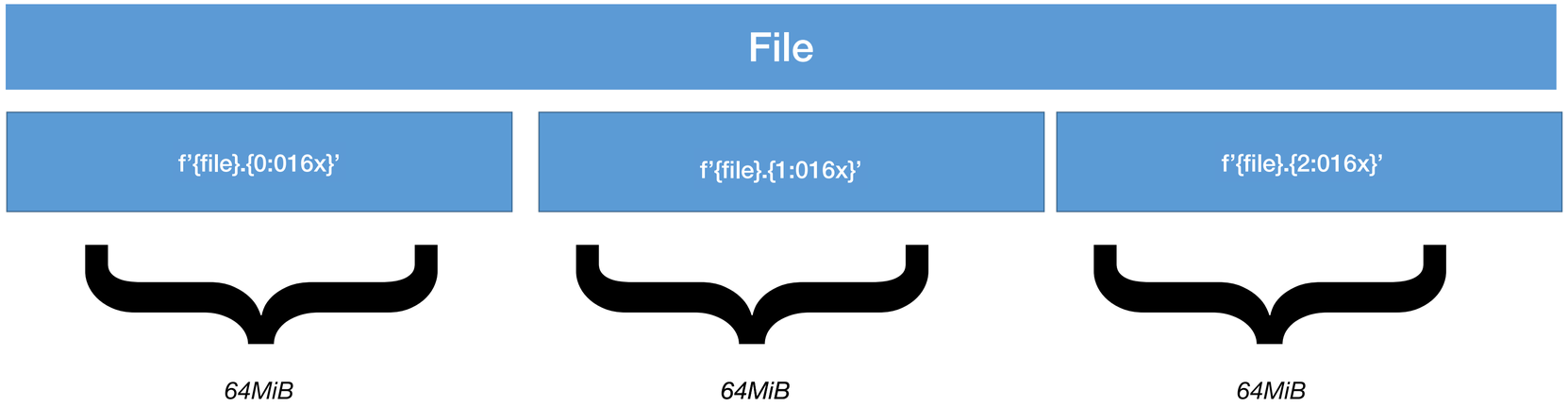



- For Internal access, ie. staging data to Worker Nodes,
  - Each WN has XRootD Xcache + server configuration
  - Writes from the WN go via the external gateways
- Further specialised hosts for Alice and CMS AAA
- Work almost completed to move to clustered XRootD on External gateways (with 2 CMSD managers) for better load balancing / fault tolerance.
- On WNs, architecture about to be updated, removing XCache following new readV work, (see later slides … )

# Storing data in ECHO: libradosstriper

- XrdCeph (xrootd-ceph) (and XRootD OSS plugin) interfaces XRootD to librados(striper)
    - GridFTP plugin also successfully deployed for production (largely deprecated by adoption of WebDav)
- Object store with flat namespace ; i.e. no directory structure - the path is the name of the file/object

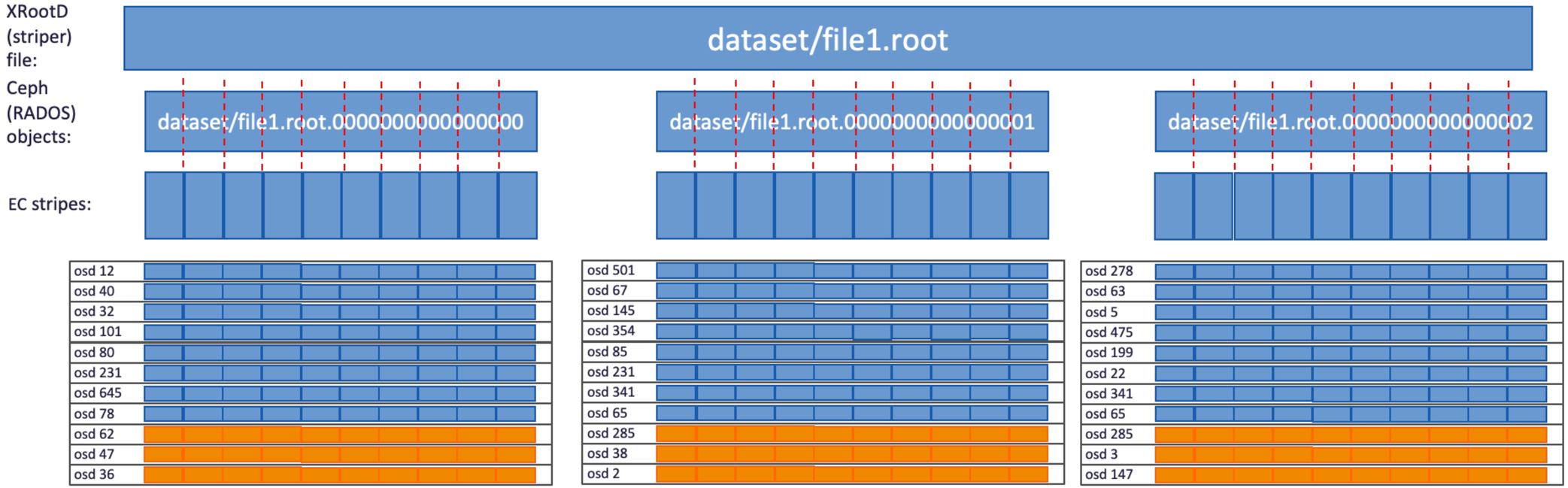
- Libradosstriper (*in a nutshell*):
    - Converts a file into (typically) 64MiB (ceph) objects (with a '.016x' encoded suffix to the 'file' path)
    - First object encodes additional information in the extended attributes metadata (e.g. total and object size).



# Storing data in ECHO: libradosstriper

- XrdCeph (xrootd-ceph) (and XRootD OSS plugin) interfaces XRootD to librados(striper)
  - GridFTP plugin also successfully deployed for production (largely deprecated by adoption of WebDav)
- Object store with flat namespace ; i.e. no directory structure - the path is the name of the file/object

- Together with usual EC in Ceph:
  - Objects on disk are made up of all the chunks/stripes for that object:

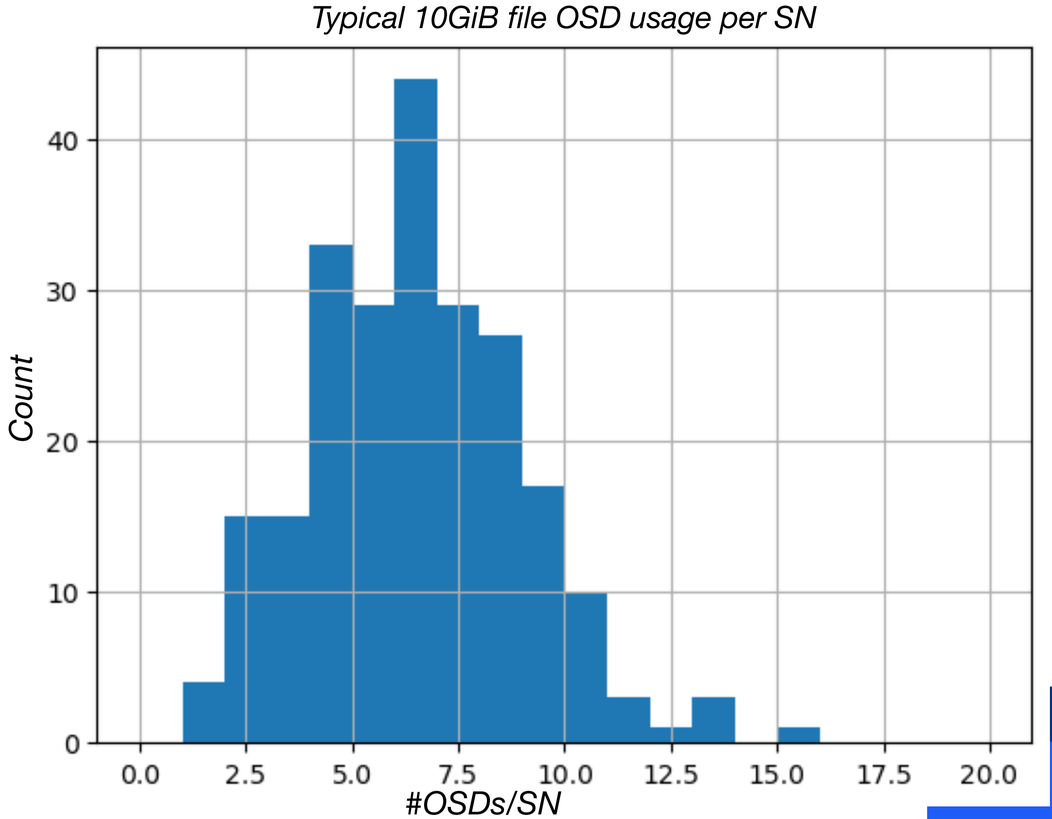


# Storing data in ECHO: libradosstriper

- XrdCeph (xrootd-ceph) (and XRootD OSS plugin) interfaces XRootD to librados(striper)
    - GridFTP plugin also successfully deployed for production (largely deprecated by adoption of WebDav)
- Object store with flat namespace ; i.e. no directory structure - the path is the name of the file/object

- e.g. a typical 10GB file,
    - Total of 157 ceph objects created
    - On ~1400 unique OSDs.

    - Data situated across ~230 SNs,
    - on average data occupying 6 OSDs per SN



# Challenges for Run-3

- Recent decisions / developments influencing ECHO:
  - Adoption of WebDav (deprecating gridFTP) for bulk of data transfers in WLCG:
  - Paged reads and writes in XRootD
- These introduced (typically) small reads/writes against the storage
  - Previously, e.g. XRootD transfer (using root://) would have ~8MiB chunks
    - Paged reads/writes => 64kb
  - ~1MiB chunks between XRootD (HTTP) and the storage layer
- Libradosstriper: no (direct) vector read support; poor performance in direct-IO like jobs

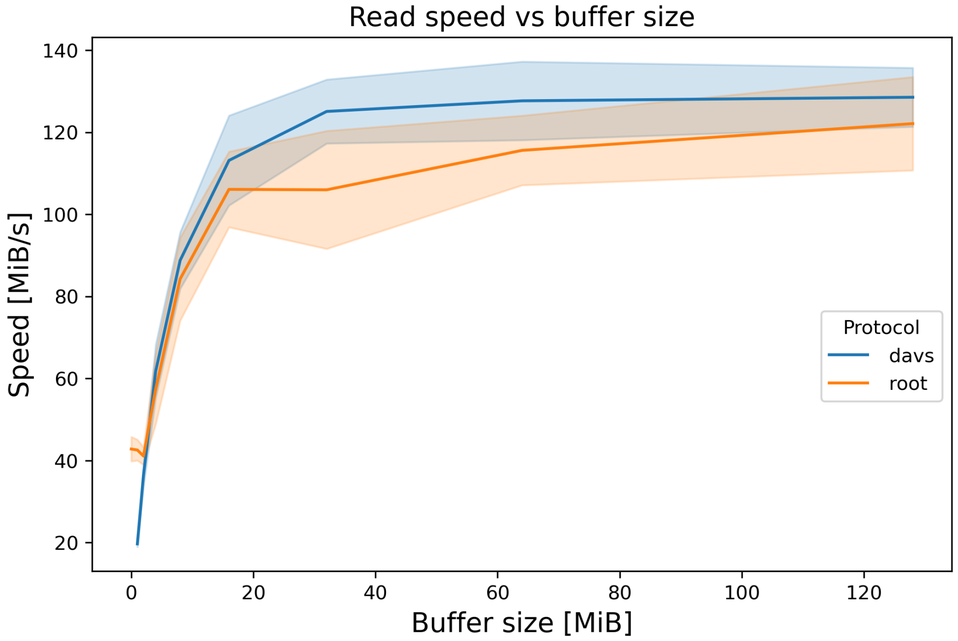
- Libradosstriper designed to provide *mostly* atomically correct behaviour for all r/w operations
  - Overhead of Locking and unlocking behaviour for small reads / writes
  - Less efficient for WORM

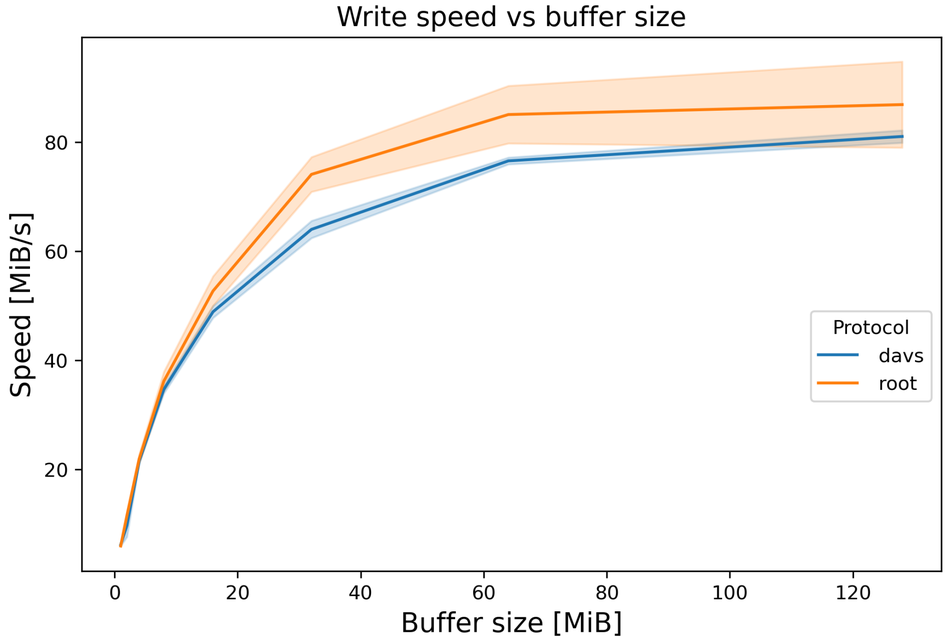
- Previously; caching (memory or XCache) proxies in XRootD to construct large IO requests;
  - Not always behaved as assumed, or in bypass/overload state it exposed all (small) reads to ceph/libradosstriper

- **Strategy 1**: Implement a buffering layer in the XrdCeph plugin to mitigate small reads/writes
- **Strategy 2**: Bypass (where appropriate) the overheads of using the libradosstriper, and utilise (optimised) librados calls directly

# Strategy 1: Buffering

- Simple buffering layer added into XrdCeph:
  - Reads: Read large chunk from ceph; hand out small chunks to client as requested.
    - Bypass the buffer if read size is at least the buffer size.
  - Writes: Accumulate writes into buffer, and flush to ceph when buffer is full (or at file end).
- Limited caching per-se, but effective for whole file copies.
- Empirically, optimised at 16MiB buffer size for external (e.g. FTS based) transfers:




*Read and write speeds for 1 GiB files on test ceph instance*

# Strategy 2: Vector read support (and improved read operations)

- Vector Read (readV) operations:
  - Some workflows, e.g. user analysis
    only subset of data across the (usually root) file is needed
  - ReadV requests previously were serialised into individual (small) reads
  - Overheads in lock and unlock metadata operations on OSDs
- By using the atomic operations in librados directly,
  - Batches up reads into single request to offload work to the primary OSD

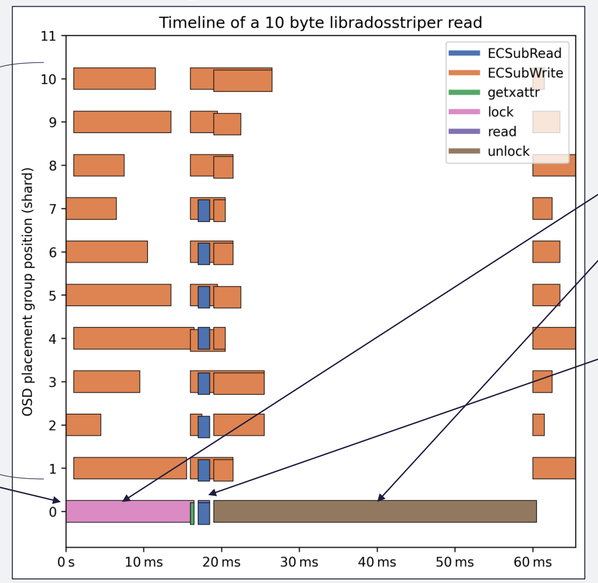
*Tom Byrne, Cephlocon '23*
*https://sched.co/1JKZT*

- Librados supports atomic operations – multiple operations on an object batched up by the client and then sent to the PG. e.g.
  1. *rados_create_read_op*
  2. *rados_read_op_read (x100)*
  3. *rados_read_op_operate*

- Slides

*Example of individual read*

## Small libradosstriper reads




Non-primary OSDs are just dealing with reading and writing to disk

Primary OSD handles requests from the client and sends "sub requests" to the rest of the OSDs in the PG

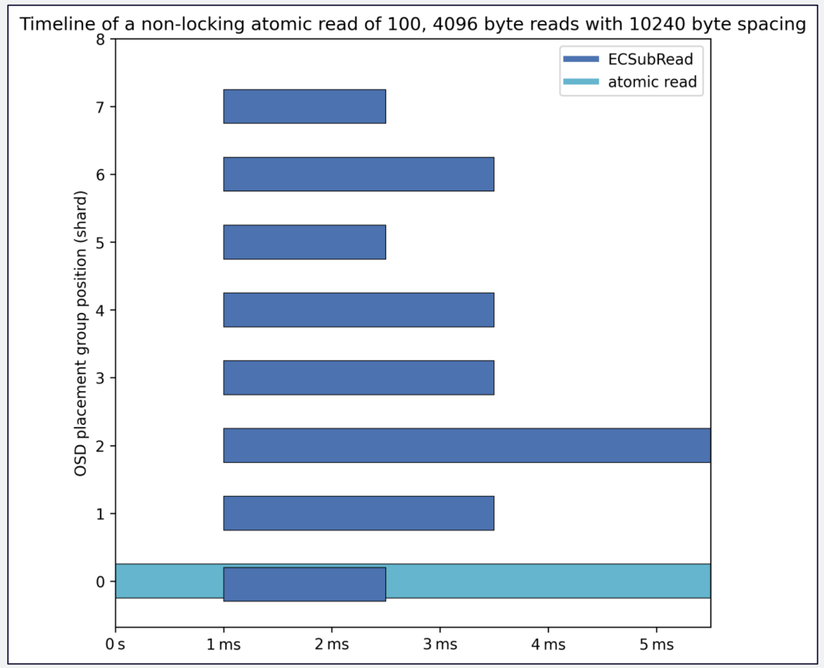
Lock and unlock require expensive updates on all OSDs in PG

The actual read is comparatively quick

(information derived from OSD messenger debug logs)

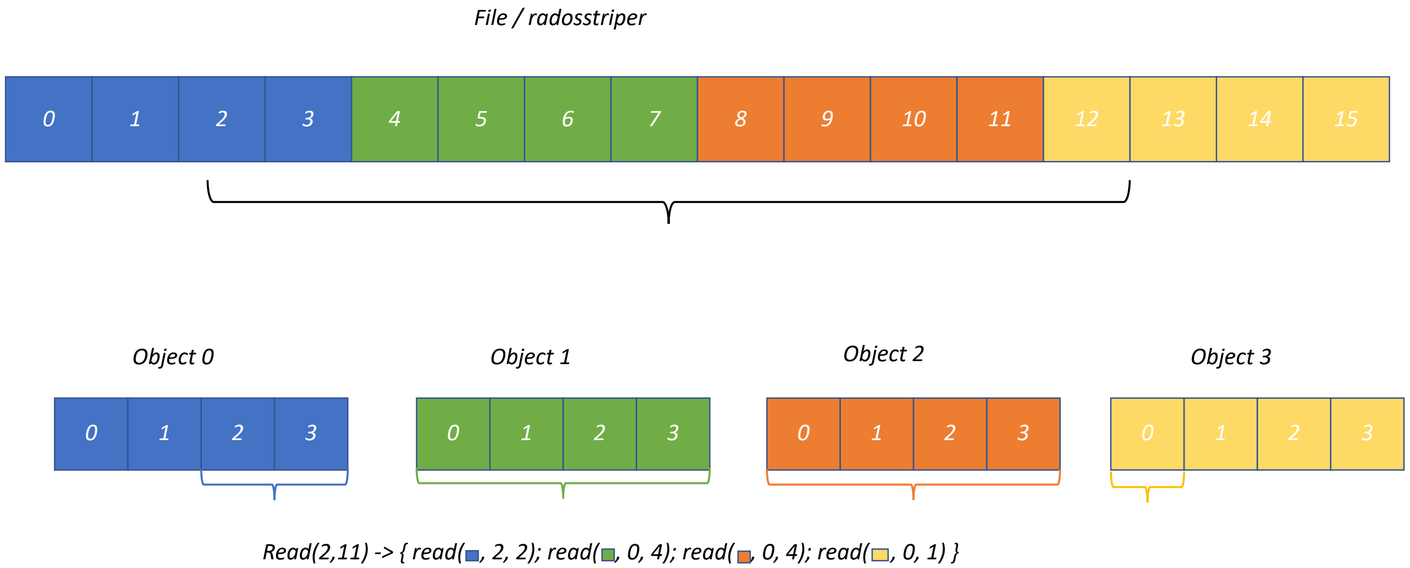
Science and Technology Facilities Council

*Example of read time for atomic read of 100 requests*



# Implementing striper-less readV support

- New code specification within XrdCeph should:
  - Reimplement the high-level features of the striper (for read operations)
    - (i.e. to chunk the file into individual ceph objects)
  - Build read operation requests within each ceph object
  - Submit and wait for completion of requests against ceph
  - Rebuild and return the readV (read) data

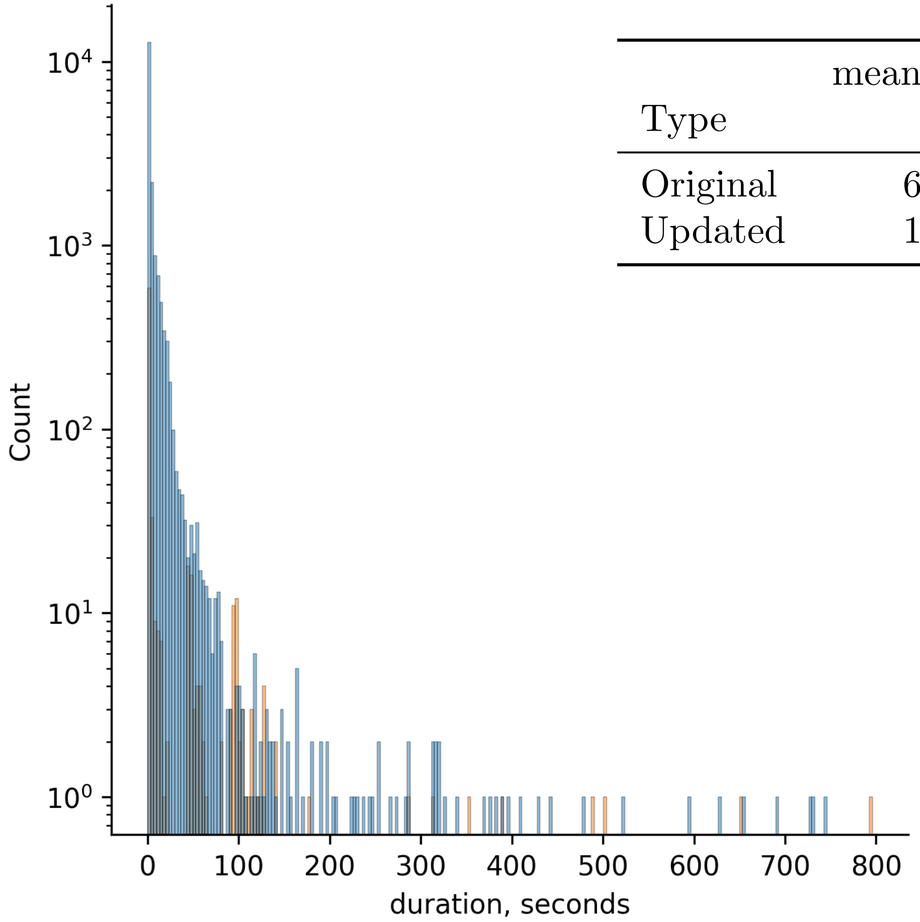



*'Cartoon' of simplied file structure, with a read request covering serval ceph-objects in a file*

# Performance of readV improvements

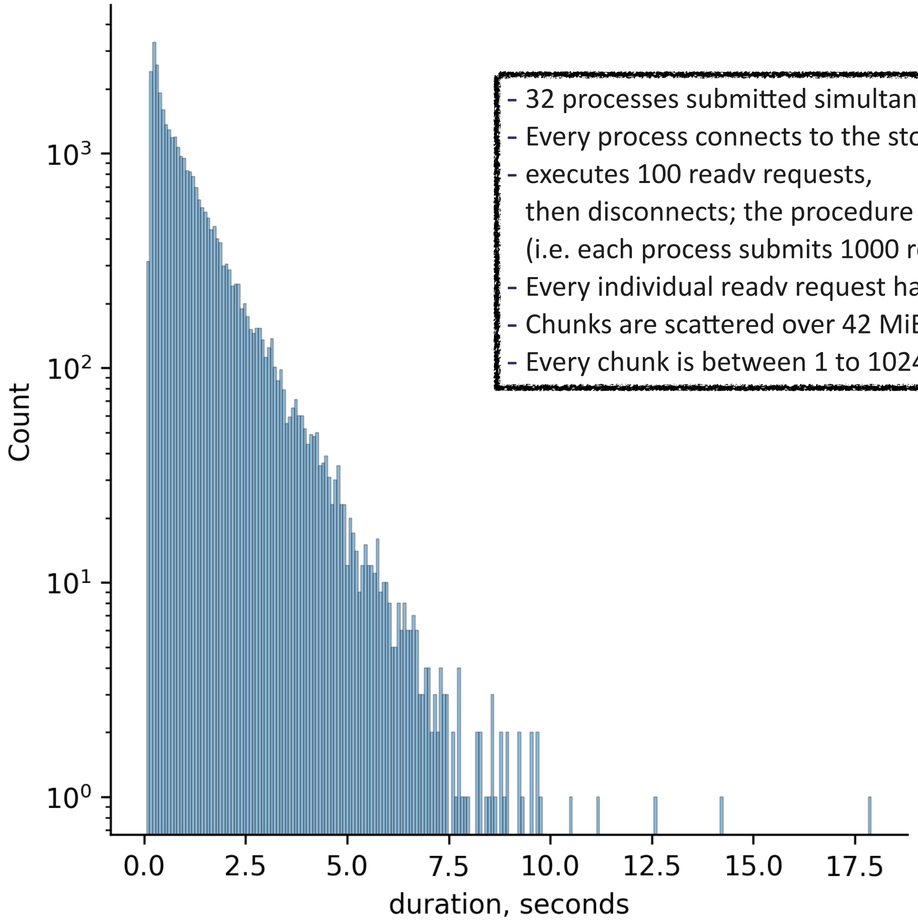
- Comparison of code using libradostriper with serialised readV requests,
  - compared to updated bypassed-libradostriper, without the additional locking, and batched readV requests:

*Original*




| Type | mean [s] | median [s] |
|---|---|---|
| Original | 6.00 | 0.84 |
| Updated | 1.07 | 0.73 |

*Updated*

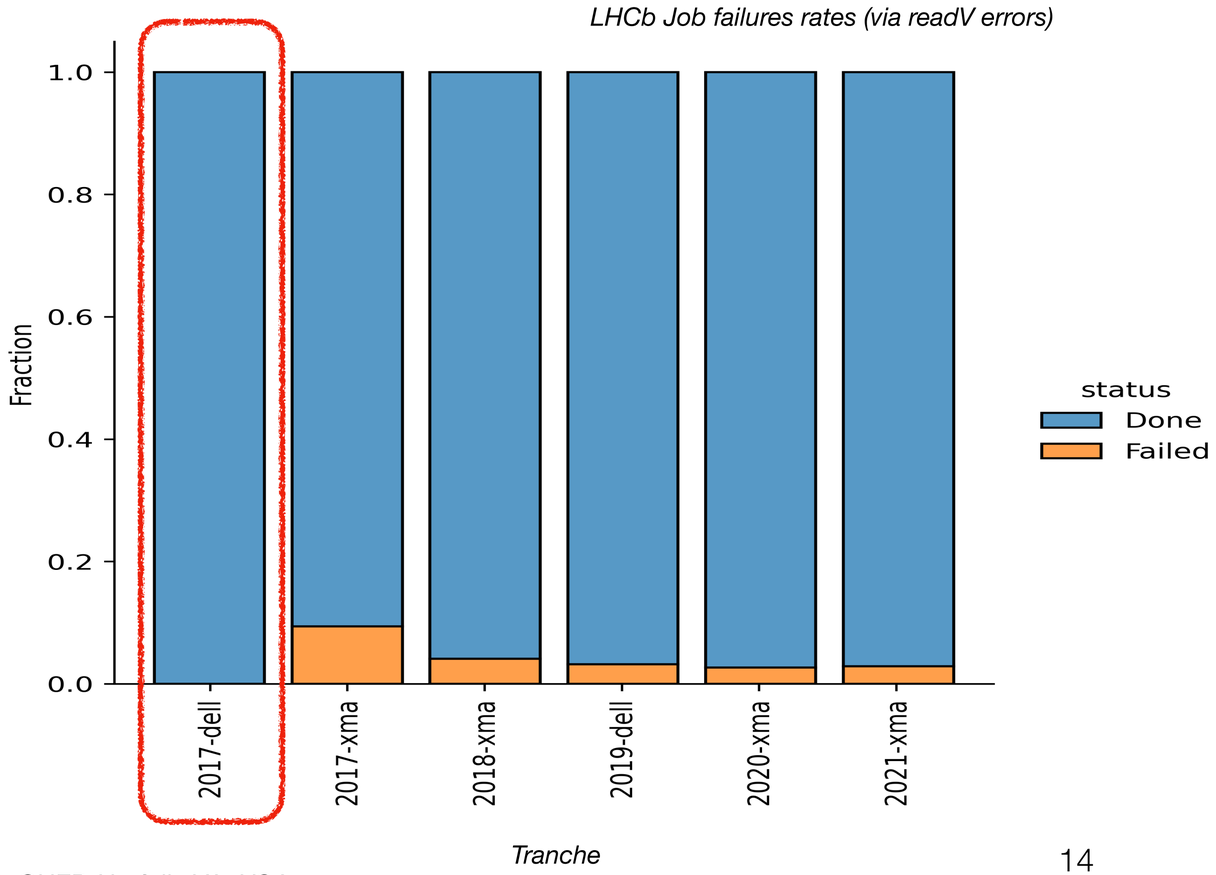



- 32 processes submitted simultaneously
- Every process connects to the storage,
- executes 100 readv requests,
  then disconnects; the procedure is repeated 10 times
  (i.e. each process submits 1000 readvs overall)
- Every individual readv request has 900 chunks
- Chunks are scattered over 42 MiB
- Every chunk is between 1 to 1024 bytes.

*Note the different horizontal scales …*

# LHC job success rate (readV)

- In final testing:
  - Updated code running on one tranche of RAL-LCG2 worker nodes:
- Currently, for LHCb jobs, negligible failure rate (for this failure mode) on the updated tranche:
  - Usual error failure type would manifest a timeout to read data



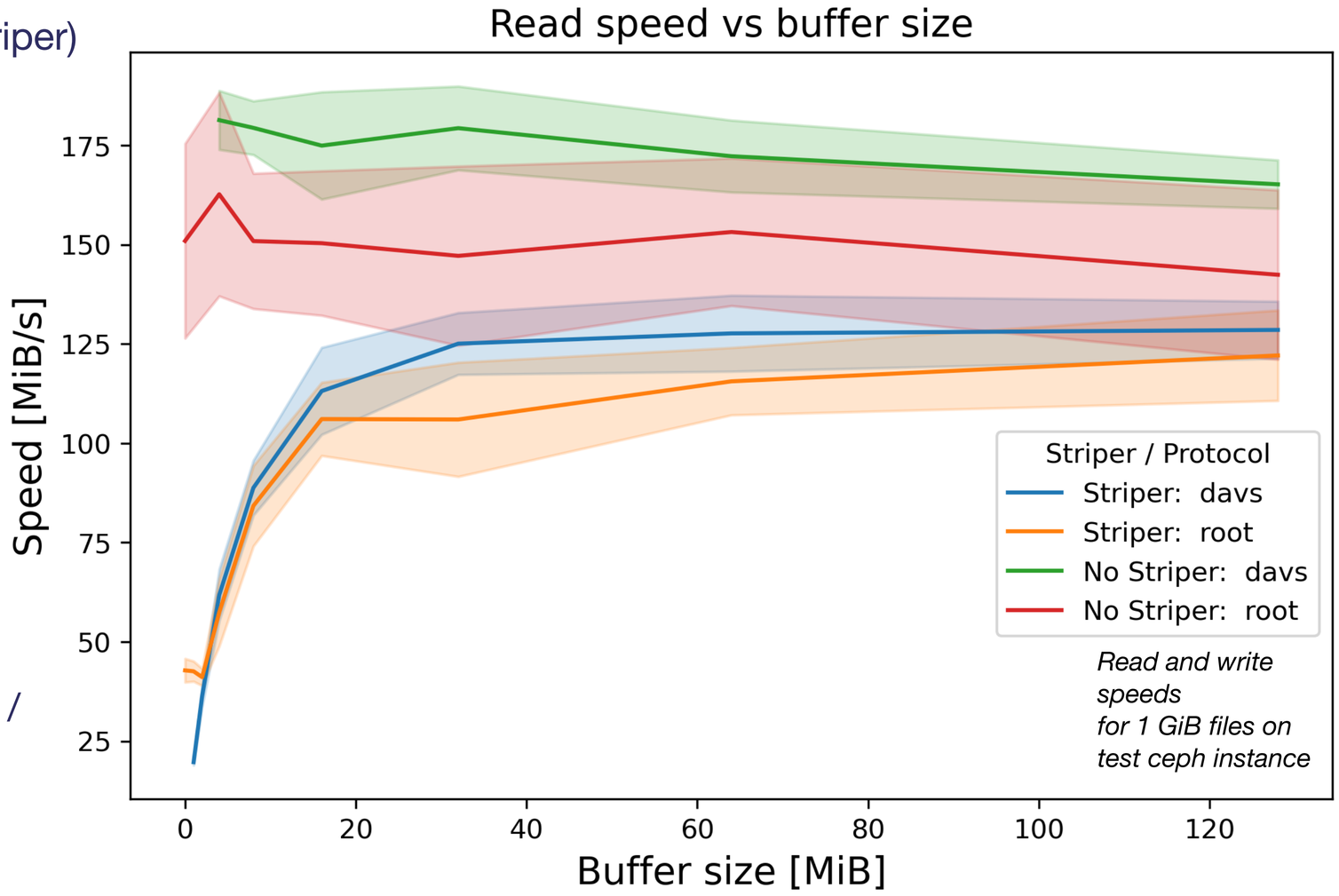
# Read improvements

- Read operations can now also bypass libradosstriper
  - Avoid the locking overhead behaviour
- Improved performance over buffering (w/ striper)

- When doing sparse file reads with buffering:
  - Low hit efficiency possible

- Buffering layer still can be important:
  - At production scale, small reads may still induce bottlenecks
  - Asynchronous buffering to hide any latency of reads from ceph

- Testing in production for different workflows / use cases (e.g. AAA, FTS-based, WN) starting
- Will deploy to all WNs this week



# Summary

- ECHO at RAL-LCG2, with XrdCeph received significant effort and developments:
  - Implementation of improvements required for successful transfers in run-3 and towards HL-LHC
  - Also for direct-IO operations using (typically) small readV requests.
- Bypassing libradosstriper and optimising with librados calls for reads can leverage orders-of-magnitude improvements in small io operations
  - Will study other potential impacts of these improvements: e.g. CMS efficiencies, by removing lazy-download
- Other developments (more info in backup):
  - Better parallelisation of deletion requests
  - Improved performance of metadata checksum operations
  - Improved space reporting functionality via xrdfs
  - Clustered CMSD redirector setup with HA 2-manager configuration using keepalived
- Further work on checksum calculation improvements would be beneficial in network throughput utilisation
- New workflows and data access patterns will continue to be studied, and further developments incorporated, if required.

# Additional updates

# Adding CMSD redirection

- CMSD should handle the load balancing of data transfers through the Gateways
- Want to provide HA for the CMSD/XRootD managers
  - Use keepalived to provide failover

- Client connects only through xrootd port 1094
- CMSD inter-communication on 1213

- DNS alias with two floating IPs is frontend
- Existing gateways act as redirected servers

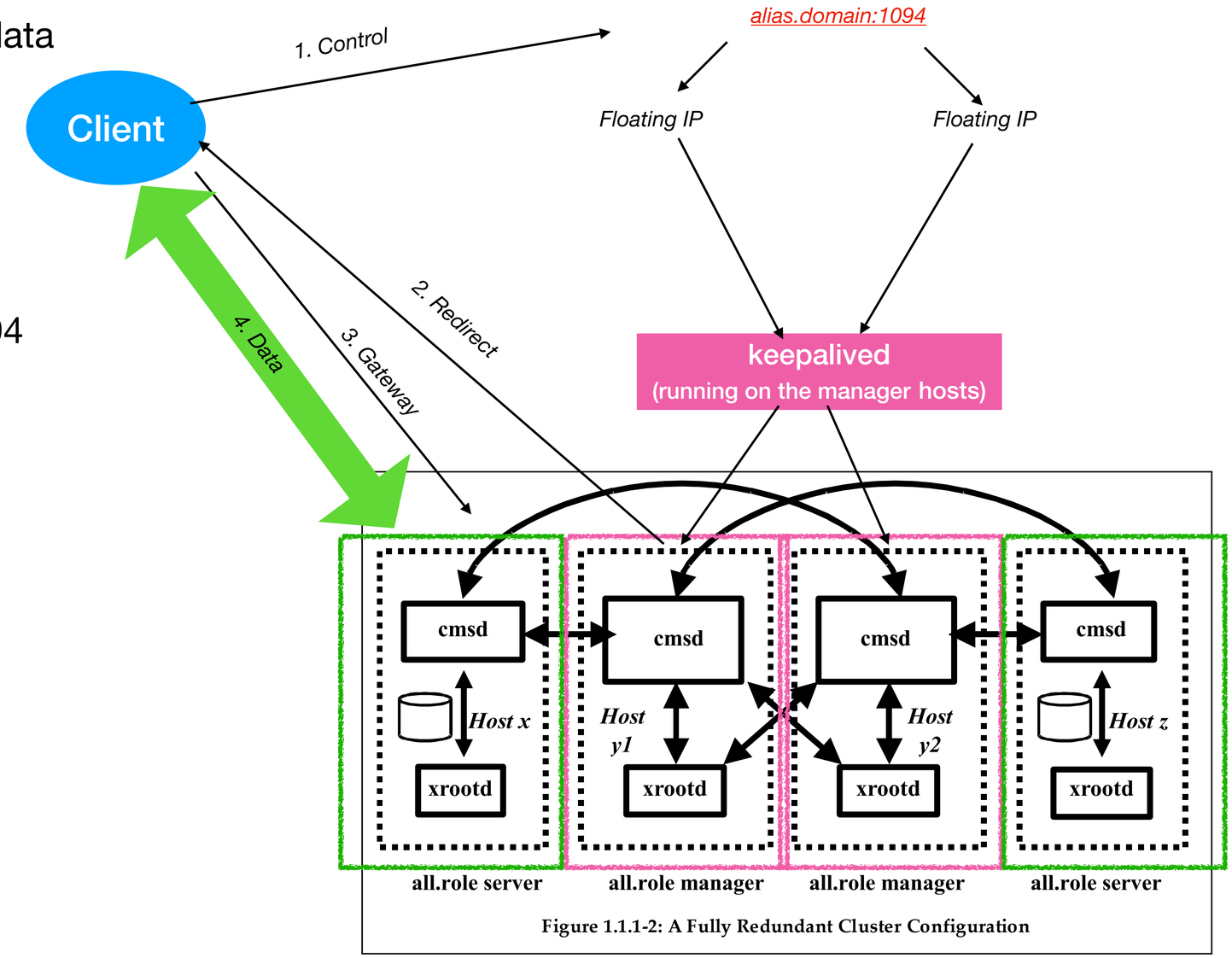
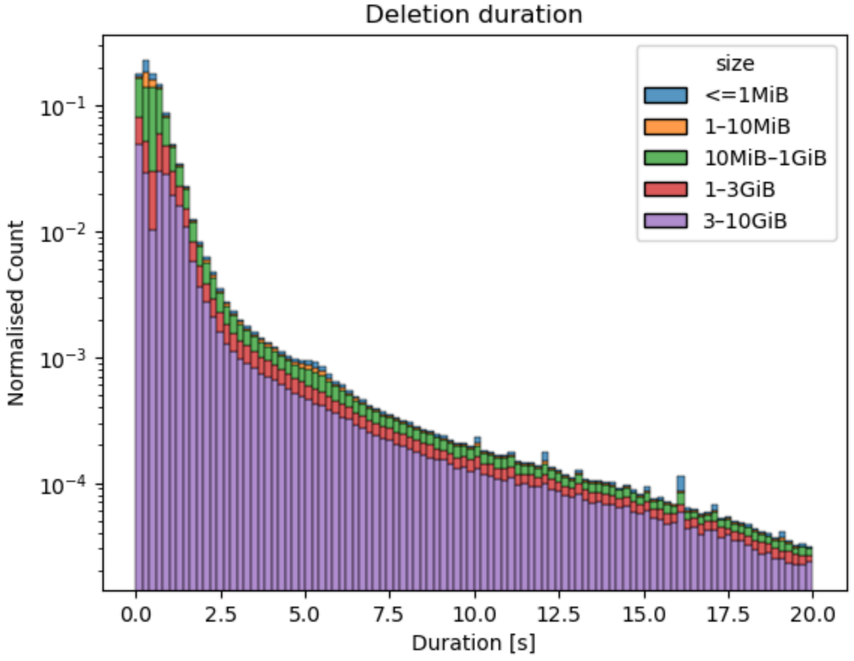


Figure 1.1.1-2: A Fully Redundant Cluster Configuration

# Updates to ECHO operations: Deletes

- Deletions performed ‘live’ against Ceph (i.e. no database / asynchronous operations)
- Moving from gridFTP to davs/root: gridFTP used a ‘python script of last-resort’ to delete files, if stuck.
- XrdCeph now includes better handling of locked files;
  - ‘stub’ (0-byte) files with missing striper metadata still needs manual handling (increasingly rare).
- Proxy + Sever configuration created serialisation of delete requests from the client.
  - i.e. one slow request (e.g. due to ceph operations, etc) would stall all subsequent queued requests
  - Removing the proxy (e.g. the ‘unified’ config) allows deletes to be parallelised:

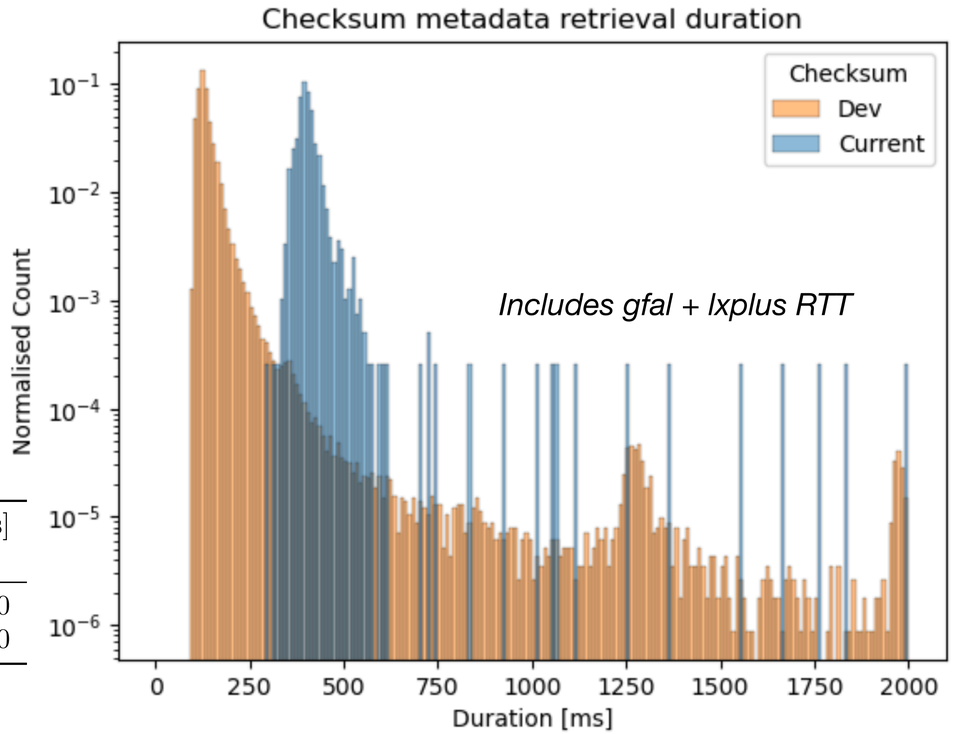
- Plot of recent ATLAS deletion times against ECHO;
  - Small dependency on file size
  - Concurrency appears to have stronger dependence
  - May require further work as filesizes and deletion counts increase.




| | Size | Mean [s] |
|---|---|---|
| 0 | <=1MiB | 0.45 |
| 1 | 1–10MiB | 0.42 |
| 2 | 10MiB–1GiB | 0.51 |
| 3 | 1–3GiB | 0.73 |
| 4 | 3–10GiB | 1.10 |

# Updates to ECHO operations: Checksums

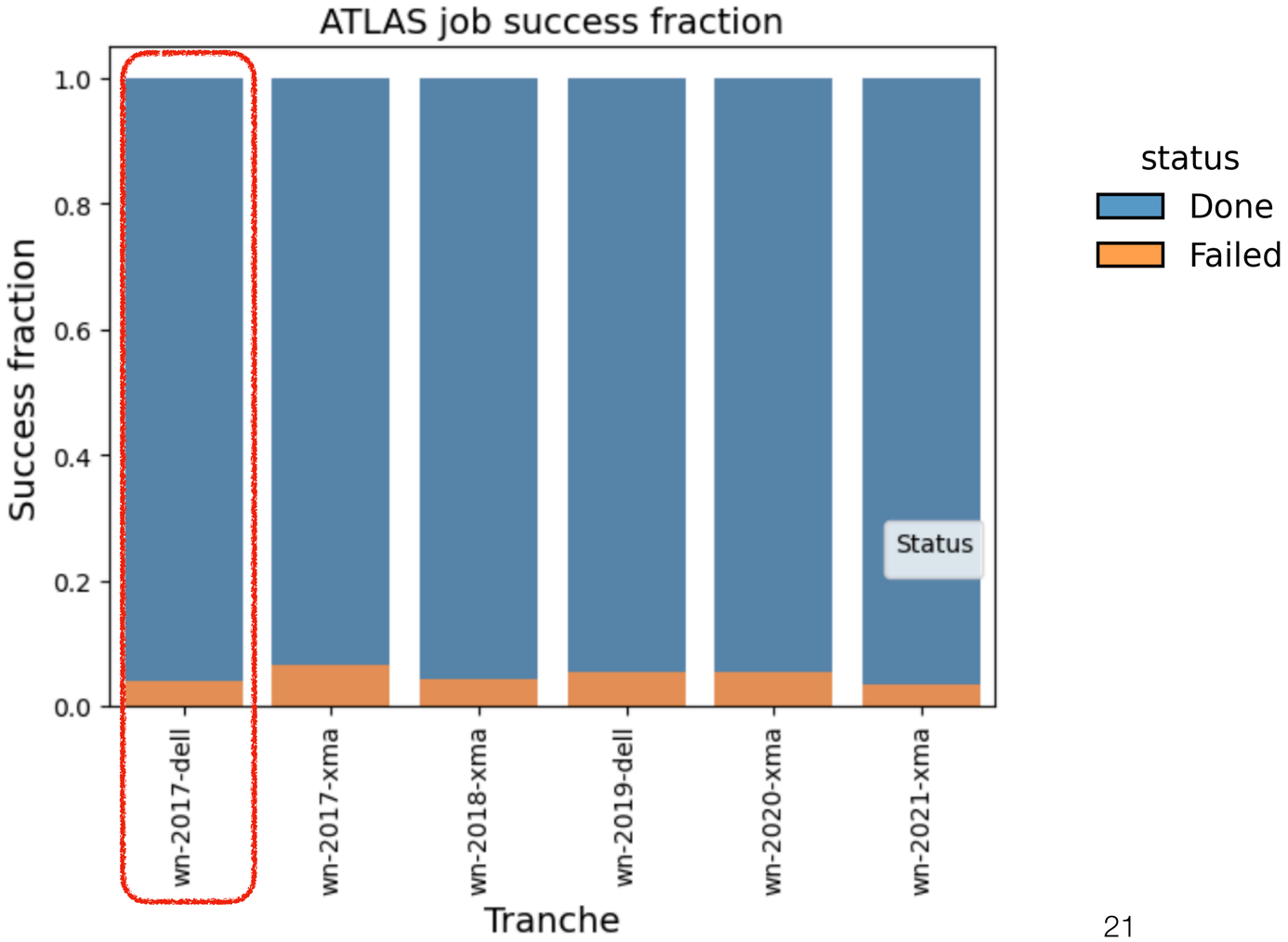
- Originally (in xrootd) could only calculate checksum from the data, when requested:
  - unable to read gridFTP computed checksums, due to endian-ness issues; GridFTP used the XrdCks format
- External python script now used to compute / retrieve checksum.
  - Additional overhead on Gateways, as data needs to be read back from Ceph to the gateway. (x2 bytes received in to the NIC); safe for the paranoid.
  - ~ 10s / GiB for checksum computation
- Currently improving this to avoid the overhead of setup / teardown of rados client connections per request:
  (important for retrieval of data from metadata).
- Several discussions on improving further: e.g. on-the-fly checksumming; and (*my* preferred) computation at the OSD level.
- Also considering developing Checksum plugin (dev documentation?)




| Version | Mean Duration [ms] |
|---|---|
| Current | 450 |
| Dev | 140 |

# ATLAS job success fraction

- ATLAS job success fraction for jobs
  - Failed jobs are characterised by “failure in the payload” for user analysis jobs using direct-IO.
    - These include file access errors,  user errors, etc.



# Storing data in ECHO: libradosstriper

- XrdCeph (xrootd-ceph) (and XRootD OSS plugin) interfaces XRootD to librados(striper)
    - GridFTP plugin also successfully deployed for production (largely deprecated by adoption of WebDav)
- Object store with flat namespace ; i.e. no directory structure - the path is the name of the file/object

- Libradosstriper (*in a nutshell*):
- The following steps are standard Erasure Coding for Ceph (librados):

- A 64MiB Ceph object:

f’{file_name}.{object_index:016x}’

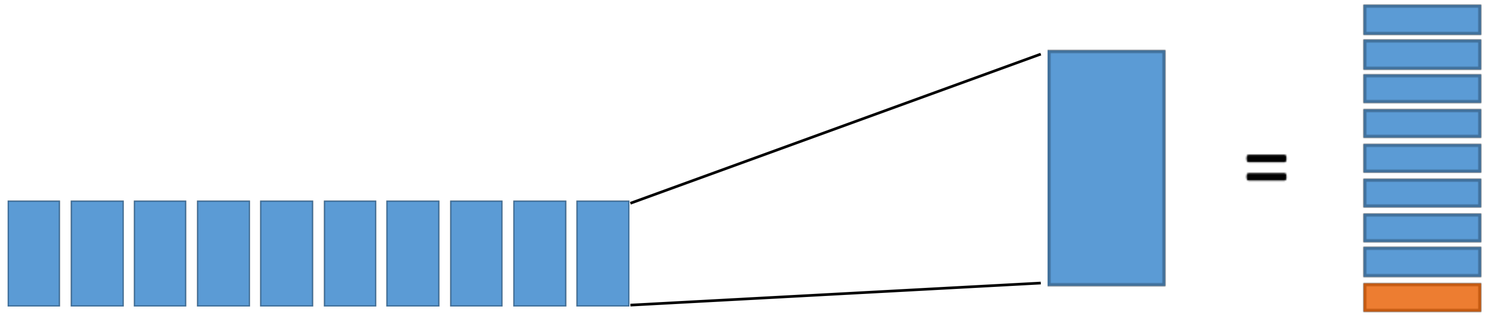
- Data is split into 4kb (or 32kb depending on pool) stripes on the primary OSD:


- Stripe size define the smallest amount of data that can be reconstructed.

# Storing data in ECHO: libradosstriper

- XrdCeph (xrootd-ceph) (and XRootD OSS plugin) interfaces XRootD to librados(striper)
    - GridFTP plugin also successfully deployed for production (largely deprecated by adoption of WebDav)
- Object store with flat namespace ; i.e. no directory structure - the path is the name of the file/object

- Libradosstriper (*in a nutshell*):
- The following steps are standard Erasure Coding for Ceph:





- Each stripe encoded into data (8) and parity (3) chunks (8+3EC) and stored across the (11) OSDs