Science and Technology Facilities Council

XRootD experiences from the UK: ECHO and T2

James Walder
On behalf of UK storage community

XRootD + FTS Workshop, Ljubljana, Slovenia
27-31 March 2023

Covering inputs and activities from (non exhaustive):

Rob Appleyard, Tom Byrne, Rob Currie,
Alastair Dewhurst, Matt Doidge,
Katy Ellis, Gerard Hand, Alison Packer,
Alex Rogovskiy, Steven Simpson,
Sam Skipsey, Jyothish Thomas,

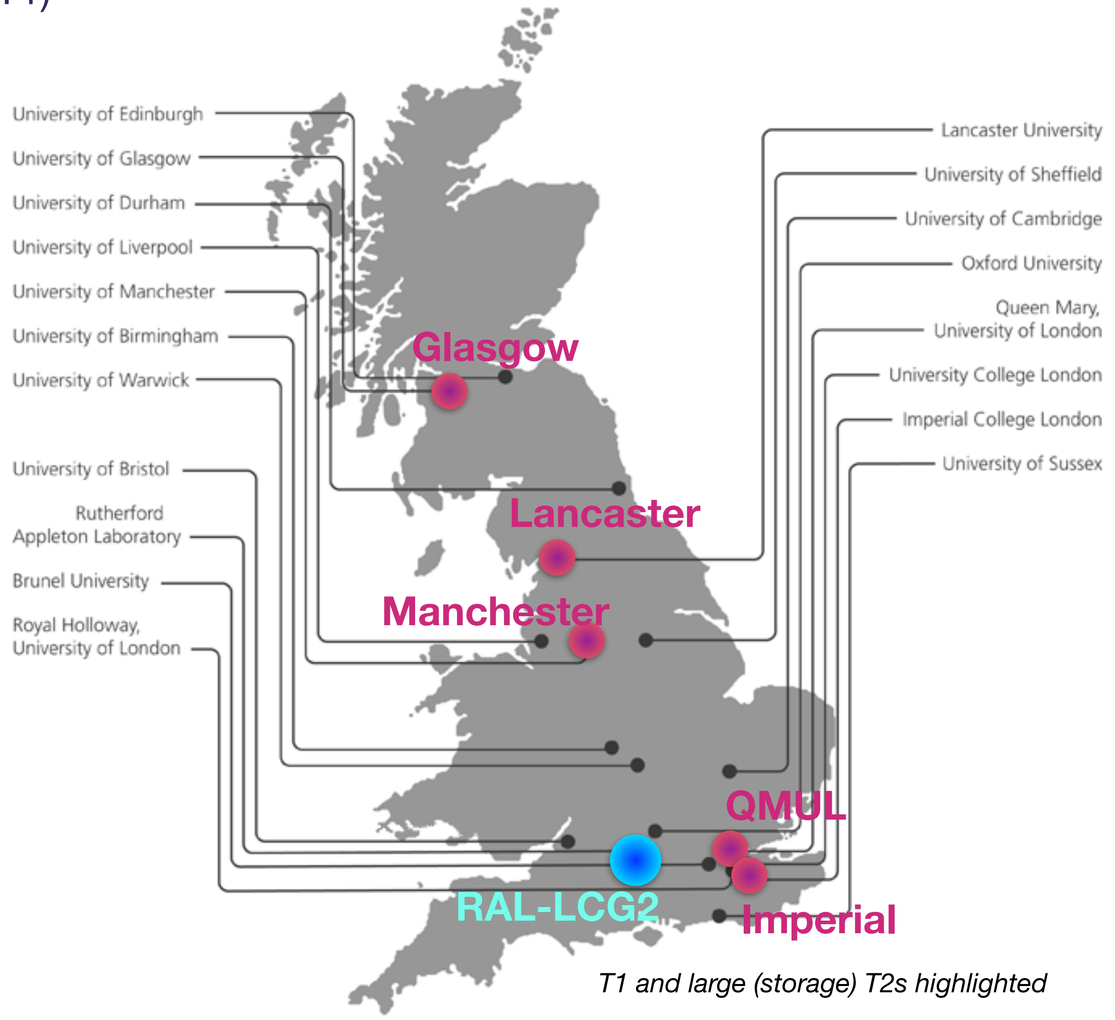
Outline

- Overview of Storage in the UK
- RAL-LCG2: ECHO
  - Object store: librados and Erasure Coding reminder
  - Main architecture developments since last workshop
  - Improvements for:
    - Deletes, Checksums, Davs, Reads/Writes, ReadV,
  - Token support
- T2s:
  - XRootD + CephFS; Lancaster
  - Monitoring
  - Caches
- UK feedback / inputs
  - Summary

Storage in the UK

- UK a heterogeneous source of storage technologies
- More recently, (significant) storage is being consolidated to 5 main T2 sites (+T1)
- With DPM EOL; smaller sites typically to become storageless:
  - Or, migrating to dCache with existing storage.
- XRootD+CephFS selected for some larger sites (see later slides)

Site	Storage (now)	Storage (if changing)
RAL-LCG2 (T1)		Echo (XRootD+Ceph)
UKI-LT2-Brunel	DPM	XRootD+CephFS
UKI-LT2-IC-HEP		dCache
UKI-LT2-QMUL		StoRM (lustre)
UKI-LT2-RHUL	DPM	Storageless (SE – QMUL)
UKI-NORTHGRID-LANCS-HEP	XRootD+CephFS (+ DPM)	XRootD+CephFS (+dCache)
UKI-NORTHGRID-LIV-HEP	DPM	dCache
UKI-NORTHGRID-MAN-HEP	DPM	XRootD+CephFS
UKI-NORTHGRID-SHEF-HEP		Storageless (SE – RAL-LCG2)
UKI-SCOTGRID-DURHAM	DPM	(TBD)
UKI-SCOTGRID-ECDF	DPM	dCache
UKI-SCOTGRID-GLASGOW		Echo (XRootD+Ceph) +
UKI-SOUTHGRID-BHAM-HEP		Storageless (SE – MAN + VP)
UKI-SOUTHGRID-BRIS-HEP		**(XRootD+HDFS)**
UKI-SOUTHGRID-OX-HEP		Storageless (SE – RAL-LCG2)
UKI-SOUTHGRID-RALPP		dCache
UKI-SOUTHGRID-SUSX		Storageless (SE – QMUL)



T1 and large (storage) T2s highlighted

RAL-LCG2 Tier-1: ECHO storage

- ECHO: Ceph-based object store with data access provided through XRootD:
  - Also deployed for Glasgow ATLAS Storage
  - Over 50PiB raw storage (+ 30PiB with upcoming deployment).
  - Nautilus + Centos7 (upgrade planning in progress)
  - 8+3 Erasure Coding

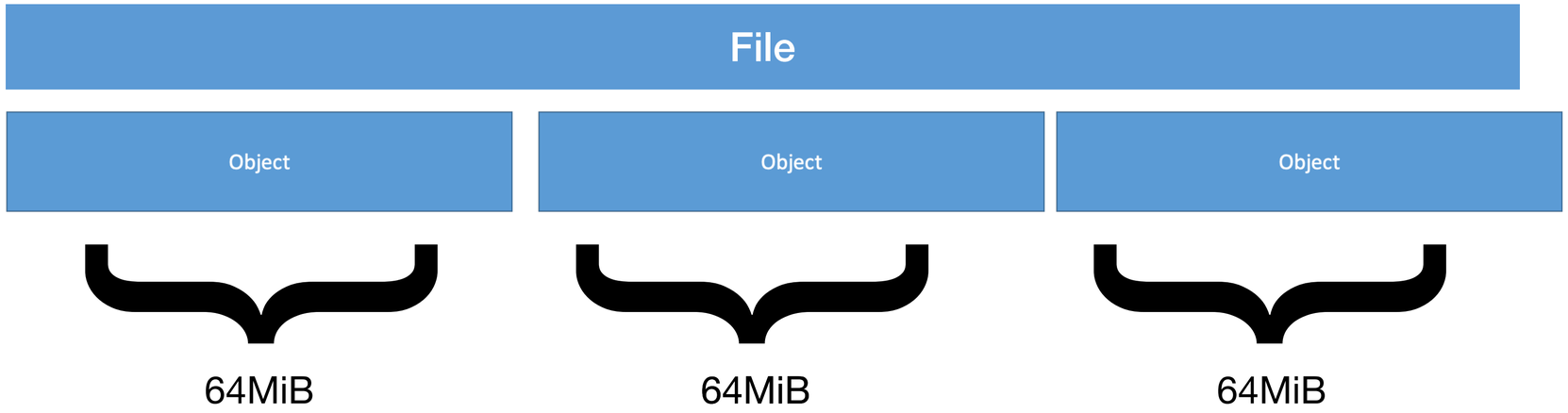
- Currently ~ 240 Storage Nodes (SN), with ~ 5000 OSDs
  - Host level failure domain (i.e. OSDs from placement group placed across different SNs).

- 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
- Also providing cephFS, S3 endpoints, etc. at RAL

- Data written to ECHO via Ceph's libradosstriper (originally developed by S. Ponce – CERN) (next slide …)

Object storage in ECHO

- XrdCeph (xrootd-ceph) OSS plugin interfaces XRootD to librados(striper)
    - GridFTP plugin also successfully deployed
    - Significant effort added recently to develop XrdCeph for efficient usage in Run-3 and beyond
- Object store; 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’ name)
    - First object encodes additional information in the extended attributes of the file (e.g. total and object size).



Object storage in ECHO

- XrdCeph (xrootd-ceph) OSS plugin interfaces XRootD to librados(striper)
    - GridFTP plugin also successfully deployed
    - Significant effort and recently to develop XrdCeph for efficient usage in Run-3 and beyond
- Object store; 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):

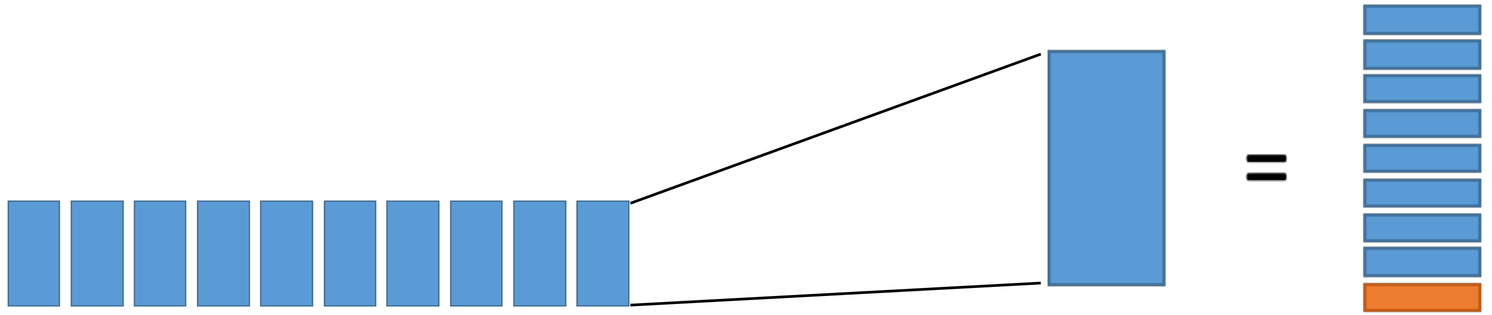
- 64MiB Ceph object:

- 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.

Object storage in ECHO

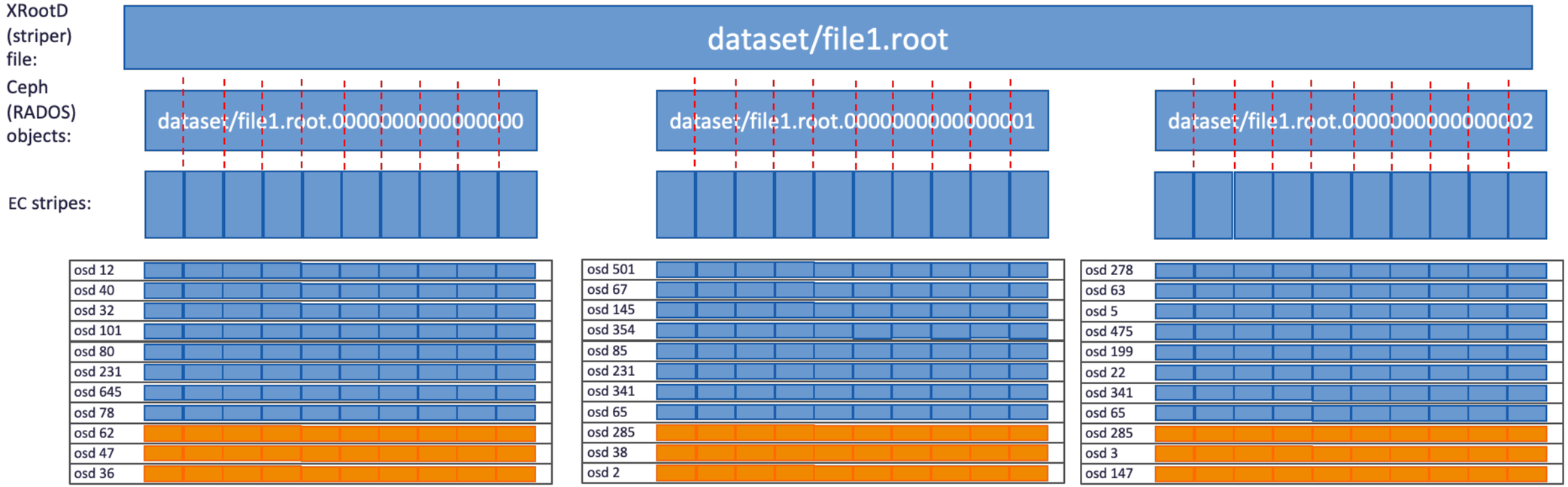
- XrdCeph (xrootd-ceph) OSS plugin interfaces XRootD to librados(striper)
    - GridFTP plugin also successfully deployed
    - Significant effort and recently to develop XrdCeph for efficient usage in Run-3 and beyond
- Object store; i.e. no directory structure - the path is the name of the file/object
- 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

Object storage in ECHO

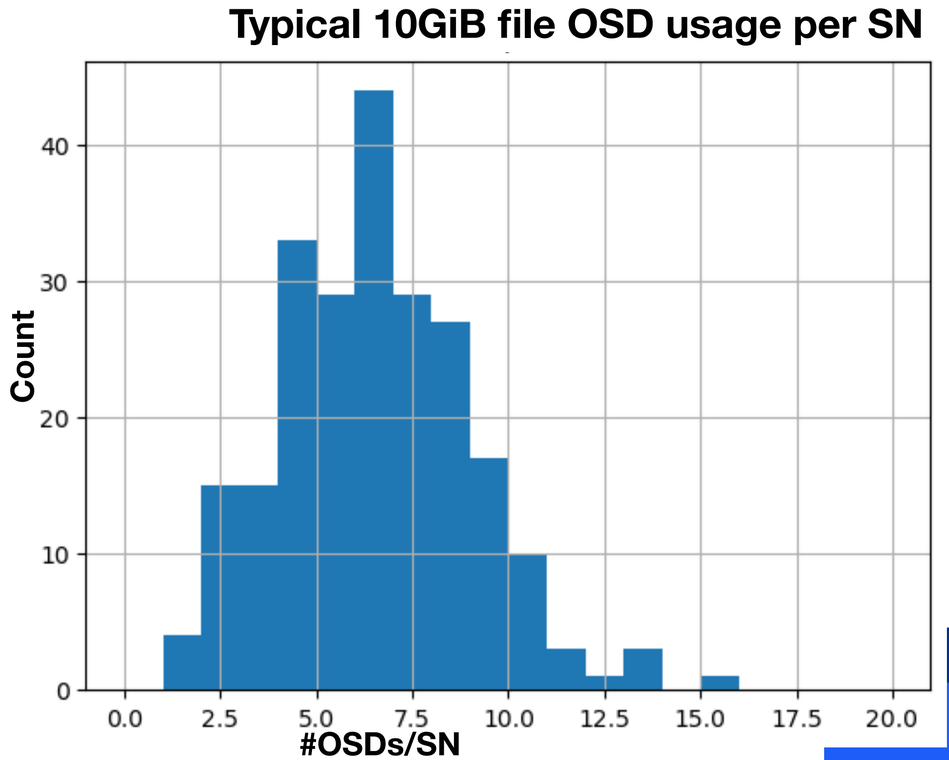
- XrdCeph (xrootd-ceph) OSS plugin interfaces XRootD to librados(striper)
    - GridFTP plugin also successfully deployed
    - Significant effort and recently to develop XrdCeph for efficient usage in Run-3 and beyond
- Object store; i.e. no directory structure - the path is the name of the file/object
- So - putting it all together:
    - Objects on disk are made up of all the chunks for that object:



Object storage in ECHO

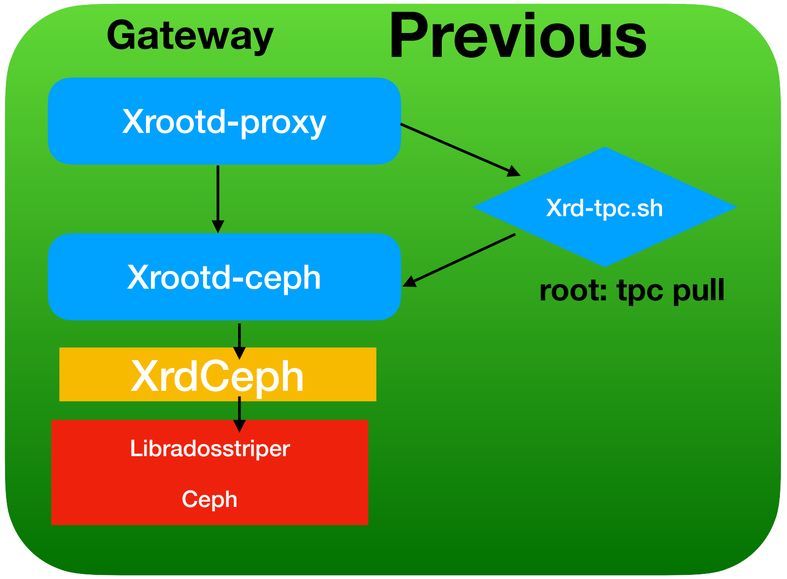
- XrdCeph (xrootd-ceph) OSS plugin interfaces XRootD to librados(striper)
    - GridFTP plugin also successfully deployed
    - Significant effort and recently to develop XrdCeph for efficient usage in Run-3 and beyond
- Object store; i.e. no directory structure - the path is the name of the file/object
- Libradosstriper (*in a nutshell*):

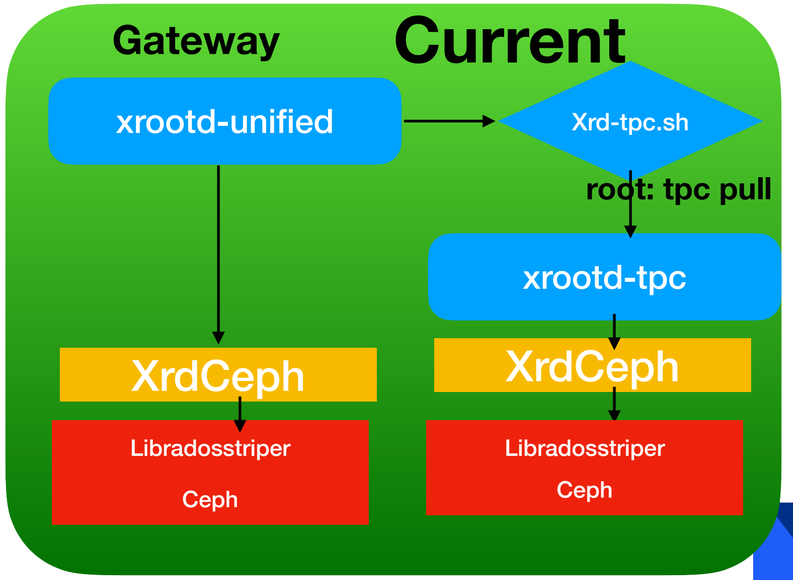
- e.g. a typical ~ 10GB file,
    - ~ 1700 total ceph objects (including the EC);
    - ~1400 unique OSDs.
  - Data situated across ~230 SNs,
    and on average occupying 6 OSDs per SN
    (typically ~ 20–24 OSDs per SN).



ECHO: Architectural updates

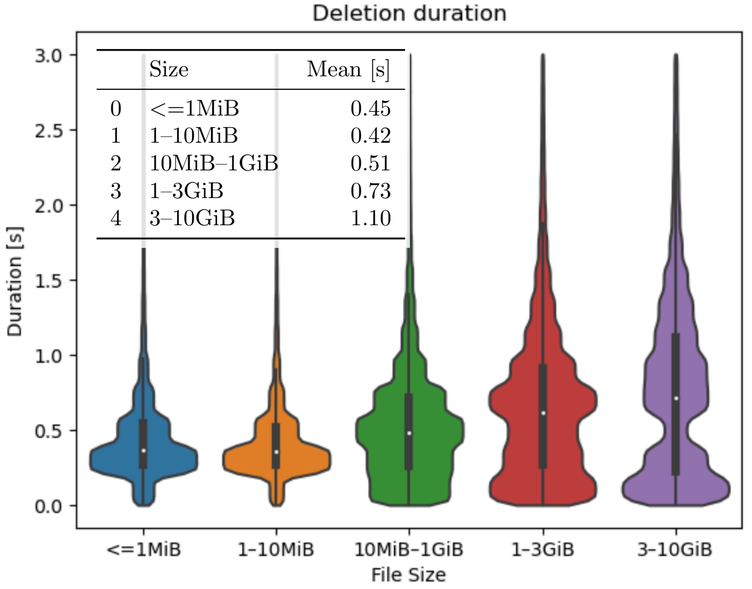
- As of the last workshop; general XRootD structure was:
- External Gateways (e.g. FTS, write-back from WN jobs)
    - Memory cache proxy + Xrootd server
    - Alice and AAA similar configs (but separate hosts)
  - Proxy:
    - caching, forwarding, and authZ/N
  - Server: OSS plugin using XrdCeph
- Updated configuration:
  - XRootD ‘unified’ server instance; Combines AuthZ/N + XrdCeph; no (XRootD) Caching (A buffer now added into XrdCeph)
  - XRootD ‘TPC’ server instance;
    - the ‘unified’ redirects to this instance for root:// TPC writes to Echo;
    - ~ same configuration as ‘unified’ (without ofs.tpc redirect).
- Future: Soon to add **CMSD redirection**; instead of DNS round-robin alias (See backup)
- **WNs (each WN host):**
  - XCache + XRootD server for stage-in; stage-out (currently) via the external gateways
  - Caching layers help readV and small read requests:
  - Improved readV code (see later), aiming to remove the Xcache



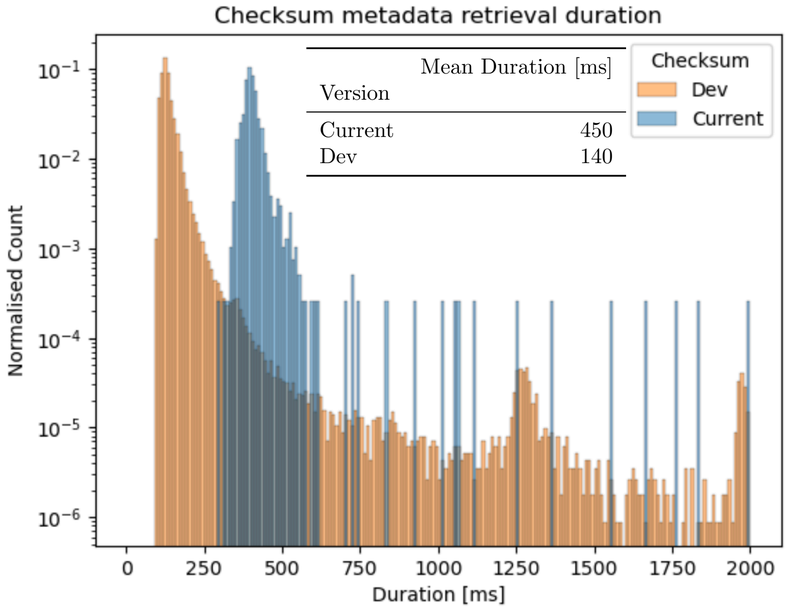


ECHO: Improved Checksums and Deletions

- **Deletions**
- performed ‘live’ against Ceph (i.e. no database / asynchronous operations)
  - 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:
  - Small dependency on file size
  - Concurrency appears to have stronger dependence
  - May require further work as filesizes and deletion counts increase.

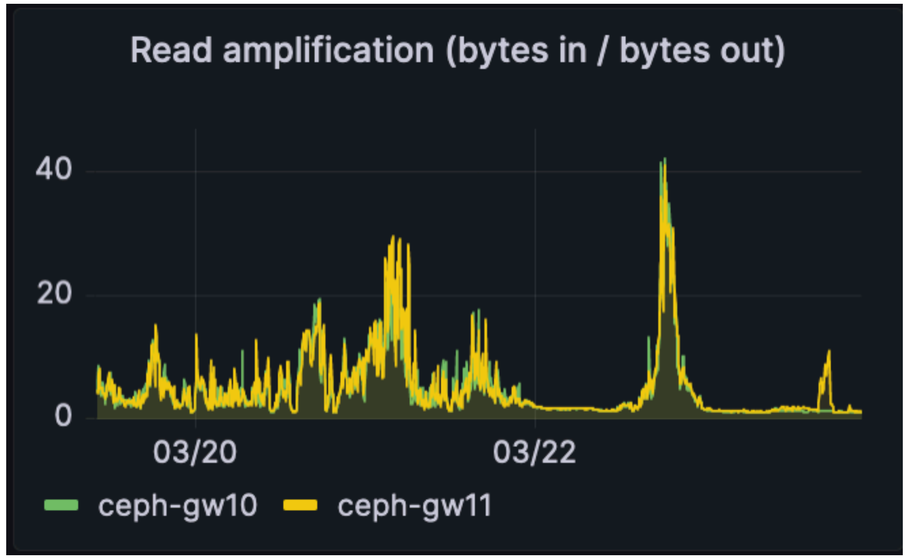


- **Checksumming**:
- External python script now used to compute / retrieve checksum.
  - Additional overhead on Gateway (compared to the data transfer):
    - data needs to be read back from Ceph to the gateway.
      - (x2 bytes received in to the NIC, x1 bytes out);
    - *safe for the paranoid*.
  - ~ 10s / GiB for checksum computation
  - Currently attempting to improve the speed of retrieval of cksums from metadata
- Several discussions on improving further: e.g. on-the-fly checksumming; and computations at the OSD level.



Reads / Writes

- Libradosstriper designed to provide *mostly* atomically correct behaviour for all r/w operations
  - Less optimised for WORM style operation
  - Locking and unlocking behaviour for small reads / writes induces overhead
- Traditionally used memory cache / XCache to try and read large blocks;
  - Not always behaved as assumed, or in bypass mode, exposes all reads to ceph
- WebDav: XRootD layer using 1MiB internal buffer and (potentially) can pass through smaller requests
  - Root: typical 8 MiB chunk size worked ok;
    - paged-reads / writes => tiny requests.
- XrdCeph – introduced internal buffer (no caching) for reads / writes:
  - 16MiB buffer is optimal in most cases for full file copies.
  - AAA (smaller buffer size) is ok, but observe some read amplification due to small read sizes.



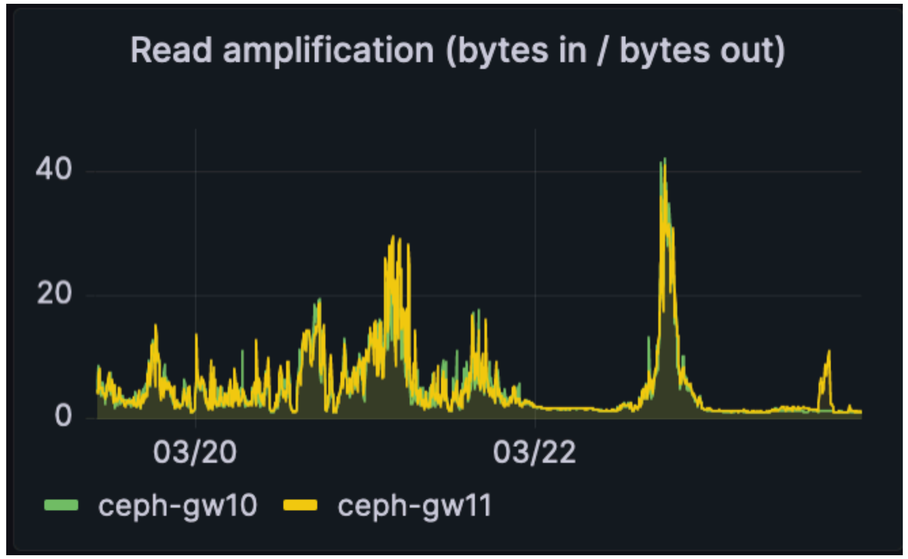
- readV developments (next slide …) may reduce the dependency on buffering reads.

Reads / Writes

- Libradosstriper designed to provide *mostly* atomically correct behaviour for all r/w operations
  - Less optimised for WORM style operation
  - (Un)Locks for small reads / writes => overhead
- XrootD caching helped to mitigate this; but with side-effects

- WebDav: XRootD 1MiB internal buffer
  - Root: typical 8 MiB chunk size worked ok;
    - paged-reads / writes => tiny requests.

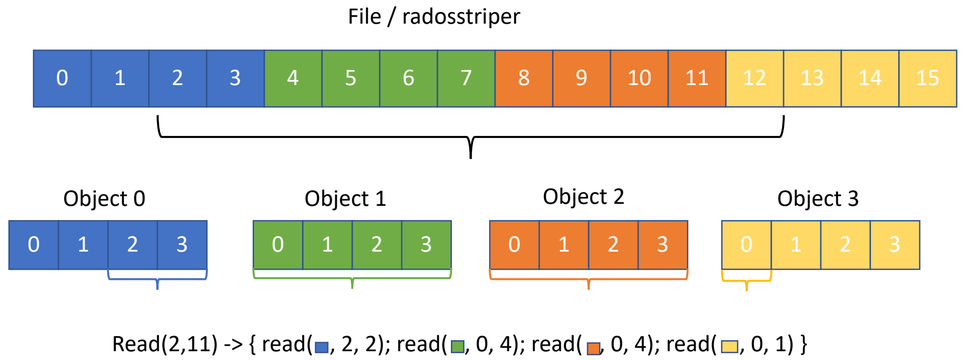
- XrdCeph – introduced internal buffer (no caching) for reads / writes:
  - 16MiB buffer ~ optimal.
  - AAA (smaller buffer size) OK:
    some read amplification when small reads.



- readV developments (next slide …) may reduce the dependency on buffering reads.

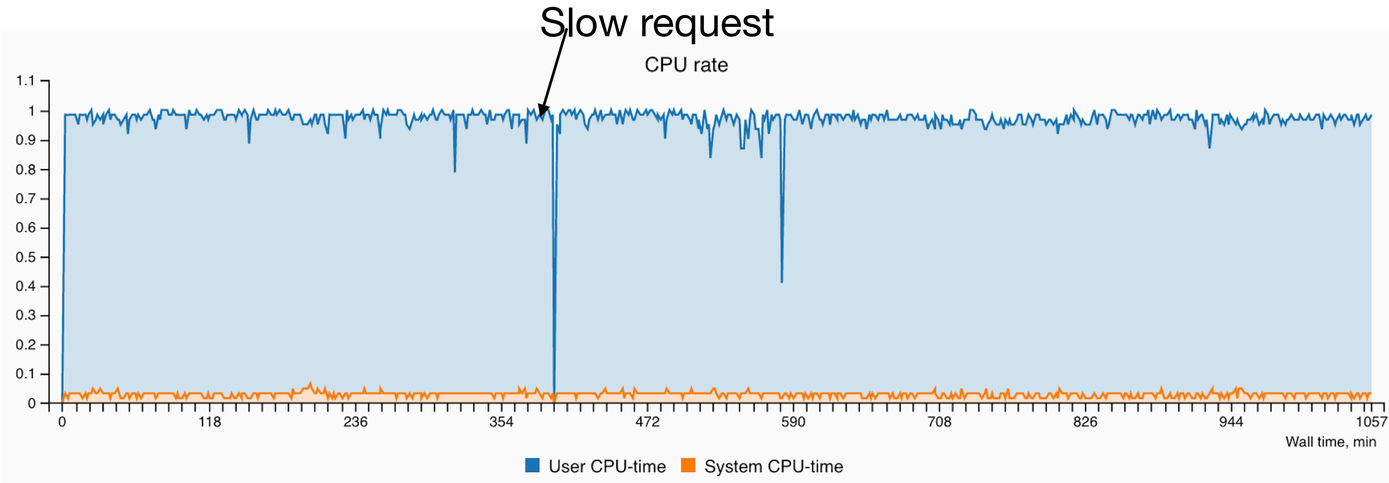
Evolving libradosstriper: readV use-case

- Libradosstriper does not support readV
  - Currently unfold a readV request into sequential reads:
    - Slow, due to striper overhead of each small read.
  - Use of XCache (on WNs) to prefetch large blocks of data:



- Now we bypass the striper for read(V):
  - Batched readVs to ceph using librados
    - Ceph on primary OSD of the PG handles the details.
  - Additional delay request timeout sent to client is also useful.

- Running on small set of production Worker Nodes significant improvements observed.



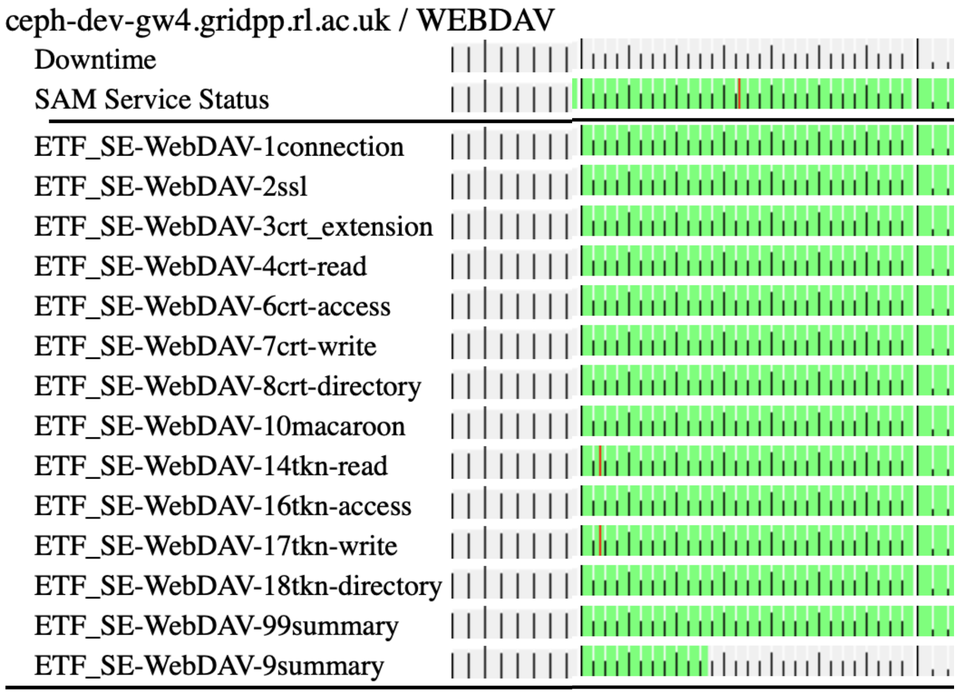
(WLCG) Token support

- Dedicated VM for tokens testbed (running 5.5.3):
  - Participation in WLCG compliance testbed and CMS token SAM tests
  - Aside from (usual) object store caveats (e.g. directories); token support should follow normal XRootD:

- WLCG compliance tests:
  - Test failures due to lack of directories in Object Store
    - Either in creation, or teardown steps (issue#45)

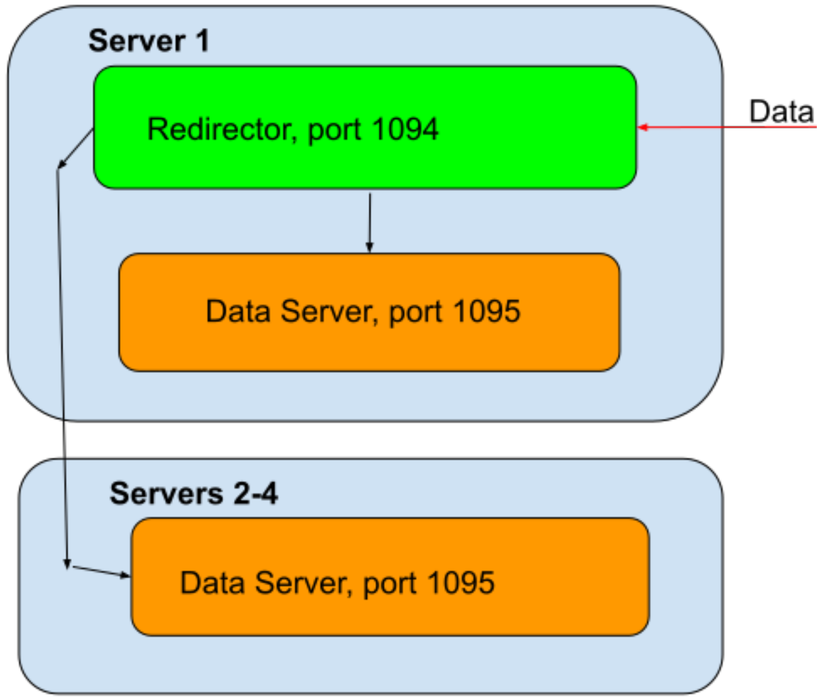
- CMS SAM test:
  - Passing current ‘tkn’ tests;

Statistics by Tag	Total	Pass	Fail	Skip	Elapsed	Pass / Fail / Skip
audience	4	4	0	0	00:00:03	
basic-authz-checks	15	15	0	0	00:00:20	
critical	24	22	2	0	00:00:34	
not-critical	2	0	2	0	00:00:05	
path-enforced-authz-checks	7	3	4	0	00:00:16	
permissive	2	2	0	0	00:00:02	
se-ral-test-xrootd	26	22	4	0	00:00:39	



CephFS+XRootD: Lancaster

- Consolidation of storage at large UK sites:
  - Lancaster: prototype of CephFS + XRootD implementation
    - Deployed ~ 10PB available storage
  - Primary motivation / requirements:
    - ‘lightweight’ and flexible frontend
    - A system where the loss of a whole server does not cause loss of data ie. CephFS
- Networking:
  - All nodes are connected with 25Gb NICs
  - racks are connected by a 100Gb backbone.
  - The site link to the NREN is a dedicated 40Gb (4x10).
- Ceph:
  - 3 admin, 2 MDS and 29 OSD nodes
  - running Ceph Pacific.
- Over time have needed to set up CMSD and scale out more XRootD servers



CephFS+XRootD: Improved WN file access

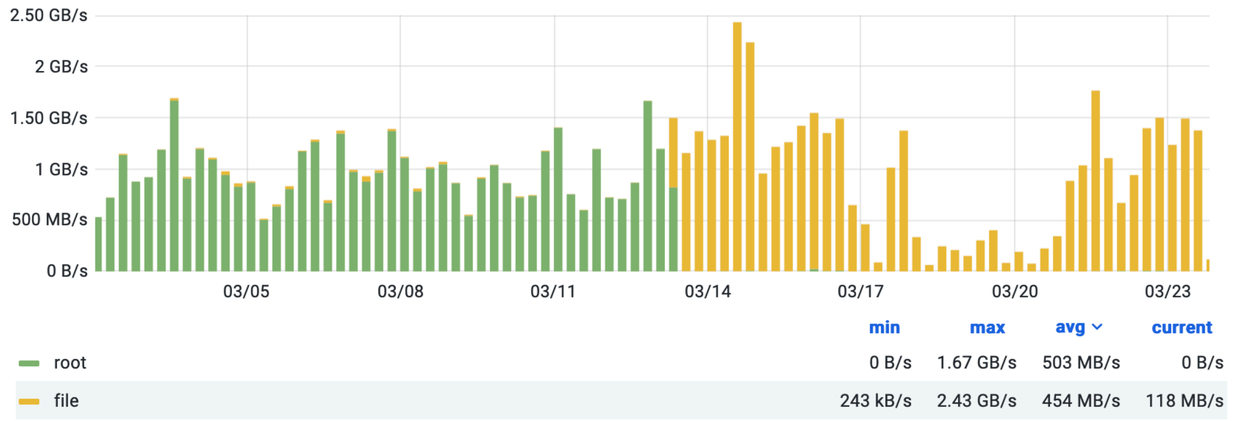
- CephFS: POSIX-compliant:
  - mounted on XRootD servers, and (read-only) on WNs
  - WNs: extended ACLs; disallow reading of directories by users not in the specified groups
    (i.e. only atlas can read /atlas)

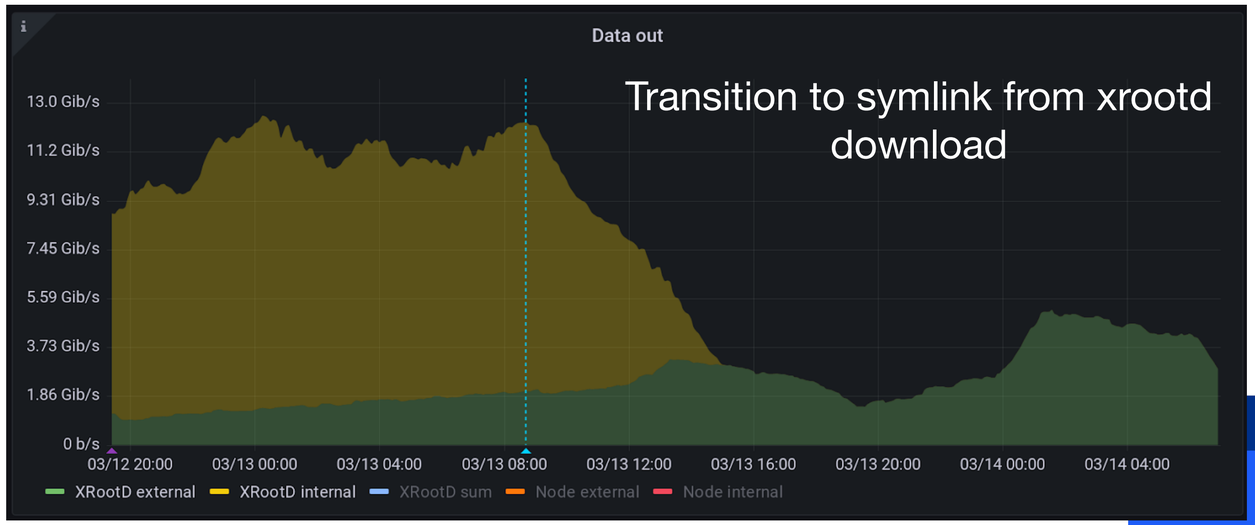
- Rucio providing new posix.Symlink protocol implementation: (testing with ATLAS)
- Keep job stage-out going via XRootD for auditing / authz / simplicity.

- Initially, problems with ACLs:
  - New directories with default ACLs;
    not allowing writes
  - A script now runs via ***ofs.notify*** to check and fix permissions on new directories.

- Servers could become overloaded with (External / FTS) transfers and Checksum calculation;
  will hopefully free up the bandwidth for this.

Stage-in to WN transition

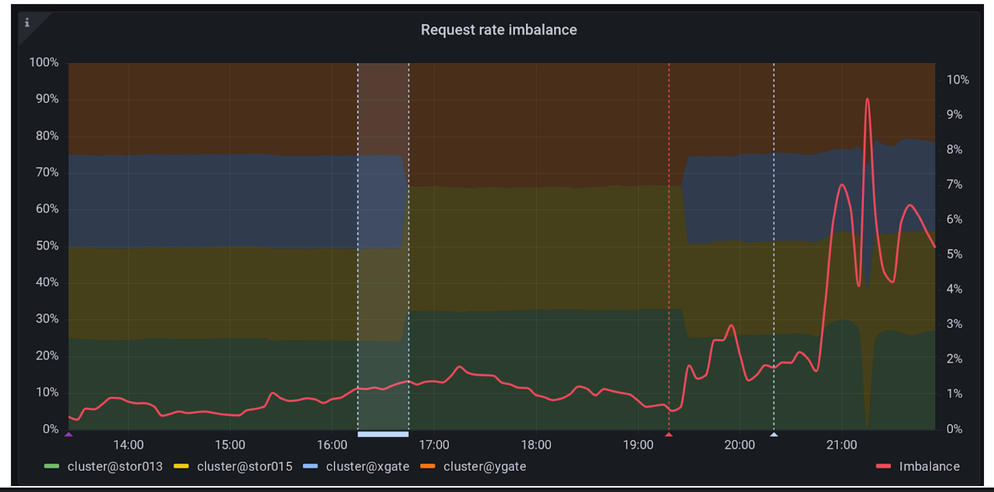


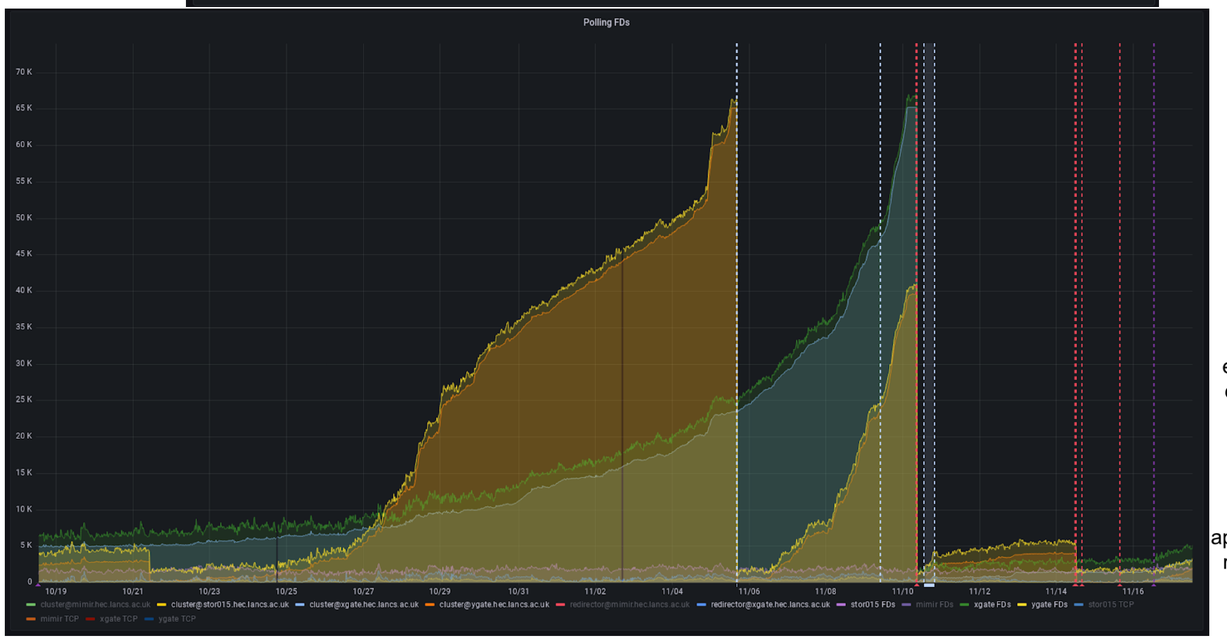


CephFS+XRootD: Monitoring

- Successful monitoring critical for successful operations (accuracy and functionality)
- CephFS+XRootD @ Lancaster

- XRootD summary reporting
  - Metadata: for correlation with redirection events
- Resource usage:
  - I/O
  - Buffers
  - FDs, connections:

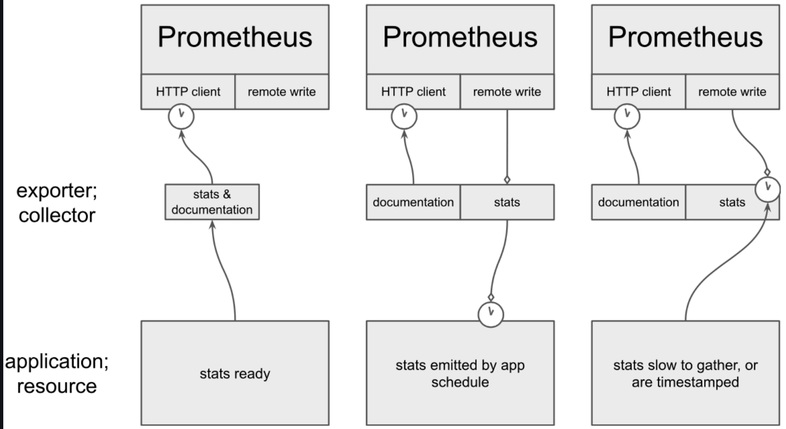




- ‘Stalling’ xrootd with observed with increasing numbers of open FDs, as connections are opened but transfers appear stalled

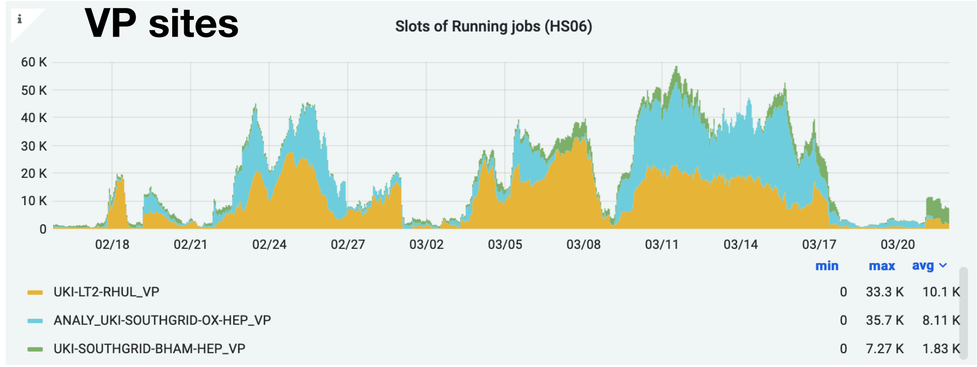
- Use of existing software:
  - Persistent, stateful services
    - Prometheus
    - Loki
    - AlertManager
    - Grafana
  - Pulled metrics
    - Node exporter
    - Ceph exporter
  - Pushed metrics
    - Loki recordings (pushed to Prometheus with remote-write)
    - Redirection events
    - Various error classes

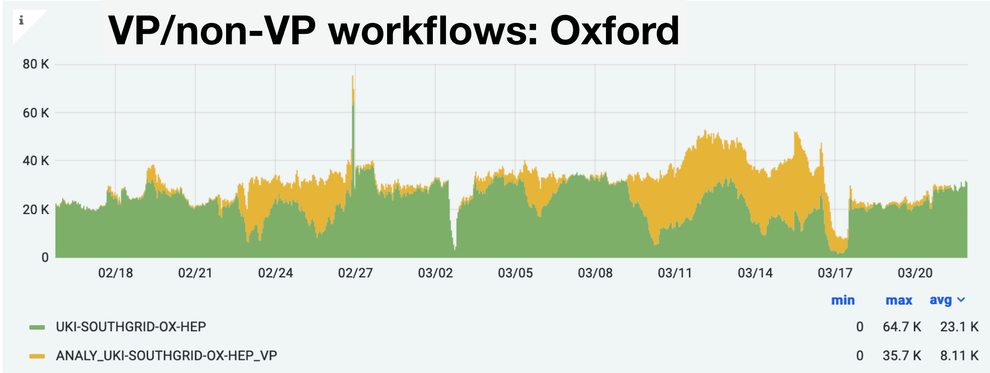
- Use of bespoke software:
  - Miscellaneous
  - Static expectations
    - Discs acting as OSDs
    - Hosts expected
    - Labelling
      - Physical location
  - Rack, position, socket
- Ceph health probes
  - Covers details missing from standard Ceph metrics
  - SMART-reported disc errors and defects
  - Slow, so remote-write staggers the delivery
  - Pool/PG complaints on OSDs



UK: Cache usage (VP and XCache)

- XCaches used:
  - Internally on each WN at RAL
  - Internally (i.e. transparently) at a few sites
  - Stashcache (ECDF)
  - Site ingress; e.g. storageless
    (more likely useful for latency, than hit rate)
  - Also exploring the usage of Virtual Placement for ATLAS:
    - Analysis workflows – using partial file reads





- Example (last 21 days); For Oxford Xcache, usage from normal production workflows included

Access type	first accesses		following accesses	
Site	UKI-SOUTHGRID-OX-HEP	RHUL	UKI-SOUTHGRID-OX-HEP	RHUL
Count	408,641	38,837	166,415	125,474
Sum of b_hit	275.8TB	1.8TB	241.1TB	14.6TB
Sum of b_miss	92TB	894.6GB	4.4TB	10.4TB
Sum of b_bypass	0B	11.5GB	0B	3.9GB
Average percentage_read	96.972%	6.141%	75.159%	14.807%
Average rate	10.43	0.23	123.319	0.688
Average sparseness	96.843%	7.561%	90.07%	52.779%

Caches and Monitoring at ECDF

- ECDF; interests in Monitoring and XCache
  - Recently observing, and trying to understand significant differences between:
    - node_exporter metrics from XCache host and,
    - Set of XRootD-based monitoring systems

node_exporter	OSG monitoring	RAL monitoring	Edinburgh monitoring
20.46 TB	5.59 TB	5.59 TB	4.79 TB. **

Total amount of data egressed by XCache between 14/02 and 28/02 (14 days)

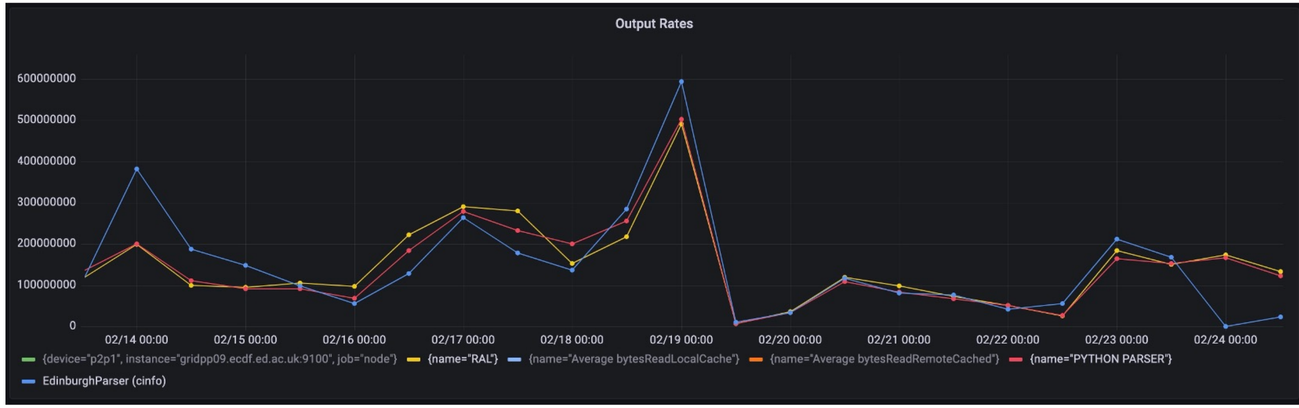
** Missing a few days of data due to service issues but same order or magnitude

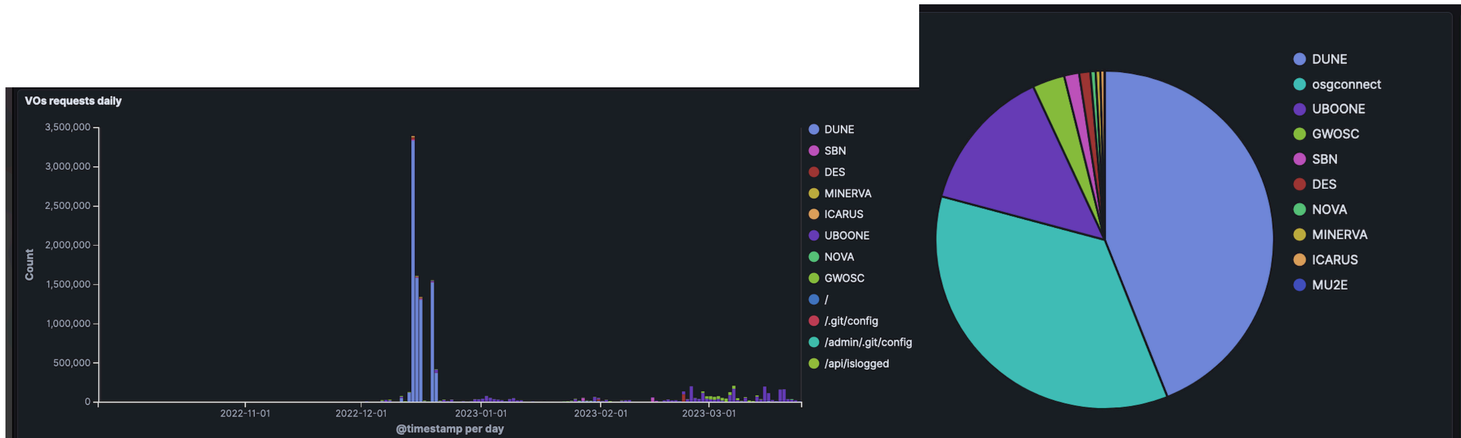
- Monitoring stacks agree, but not yet and explanation of differences to node_exporter

- Also run StashCache; while load can be high, is running well:
  - (Plot of last 6 month usage by requests and VO).

- Comparison of various Monitoring stacks:
  - Custom ".cinfo parser" and a monitoring stack ('truth' interogation') Edinburgh
  - Implemented full OSG XRootD monitoring stack OSG monitoring
  - Implemented full RAL XRootD monitoring stack RAL monitoring

•Plot showing all 3 XRootD monitoring stacks over several days. Agreement!





Feedback / Summary

- The UK runs a heterogeneous set of storage technologies at varying scales: many using XRootD
  - ECHO:
    - New dedicated effort for supporting the XrdCeph plugin.
    - Pivoting towards developments needed for the challenges of HL-LHCs (and non-WLCG VOs).
- Lancaster: Deploying CephFS takes a lot of effort:
  - Successful high-throughput XRootD deployments need to be built wide
  - Monitoring is key
- Recent releases have had some issues (particularly) for UK configurations;
  - Benefited from xrootd developer support / responses.
    - A suite of FTs using Rucio + FTS, against site test RSEs could be set up across the UK and beyond, to test our various use-cases.
- Many other activities, not mentioned here: Shoveler, packet marking, …

- The UK is gaining considerable expertise with XRootD and tends to propose it as a frontend for new users into HEP-like/large-scale data transfer orchestration and operations:
  - Improved documentation for non-experts in ‘real-world’ best-practise setups desirable;
    - Attempting to improve our feedback into the XRootD community.

Summary

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

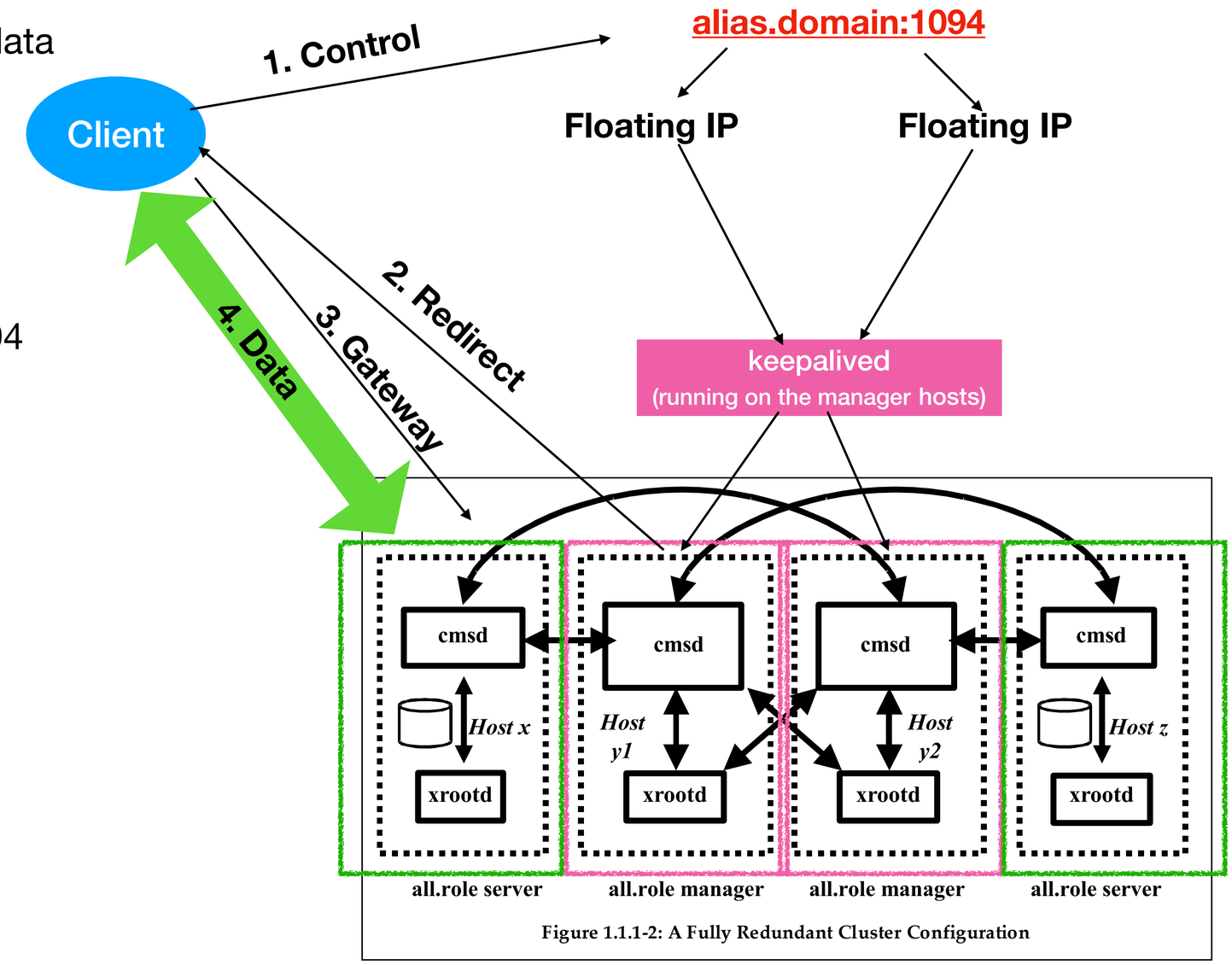
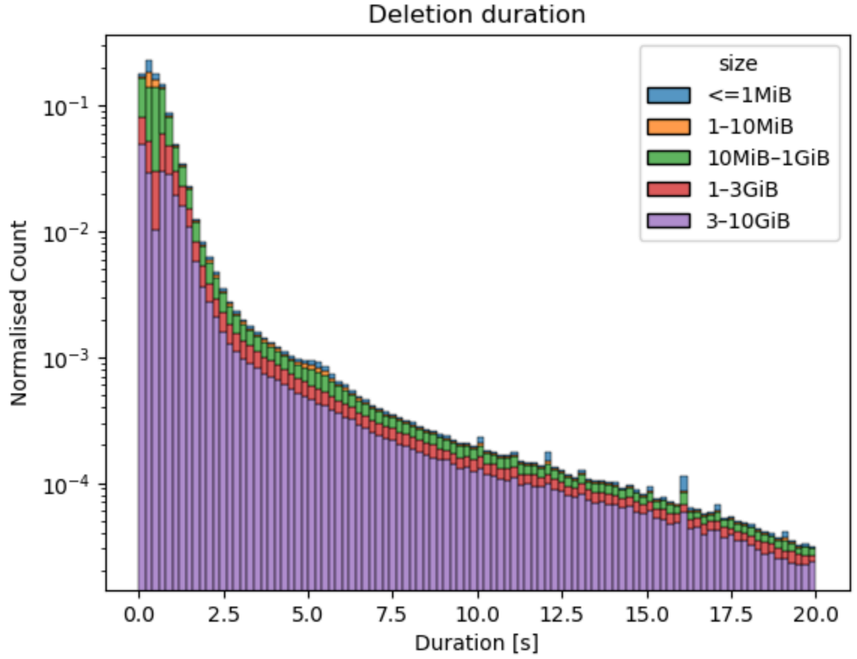


Figure 1.1.1-2: A Fully Redundant Cluster Configuration

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:

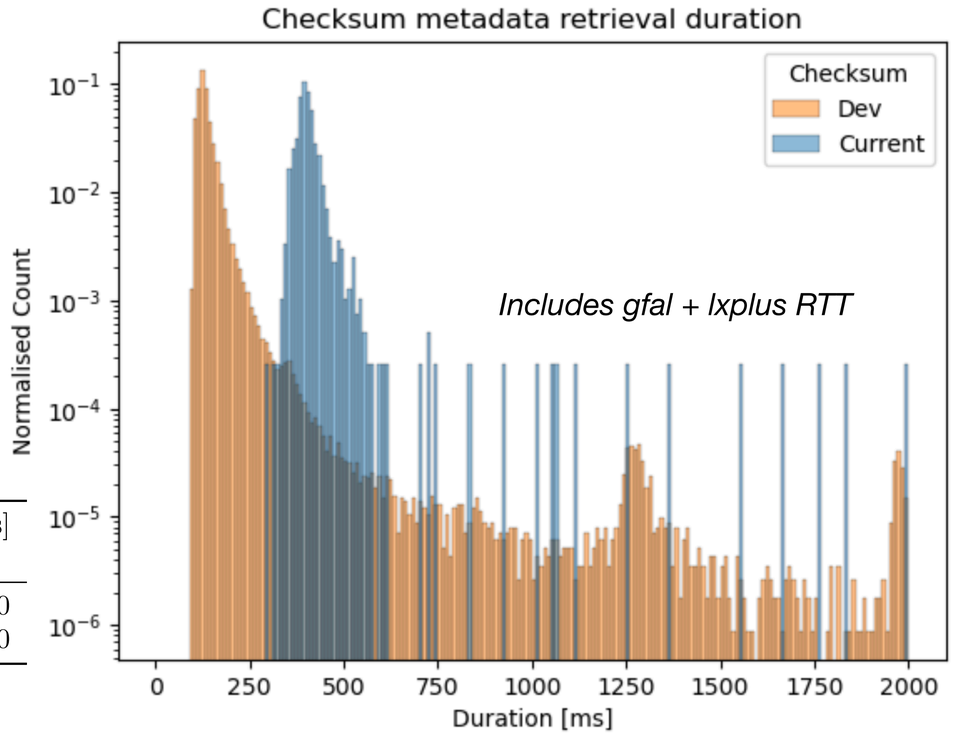
- Plot of recent ATLAS deletion times against ECHO;
  - Small dependency on file size
  - Concurrency appears to have stronger dependence
  - May require further work as filesizes and deletion counts increase.



	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