

# XRootD at RAL

---

Jyothish Thomas  
jyothish.thomas@stfc.ac.uk

# XRootD

- The XROOTD project aims at giving access to data repositories of many kinds.
- It is based on:
  - a scalable architecture (client/server services)
  - a communication protocol (root)
  - and a set of plugins and tools based on those. (e.g. XrdCeph)
- It provides features such as authentication/authorization, integrations with other systems, etc..

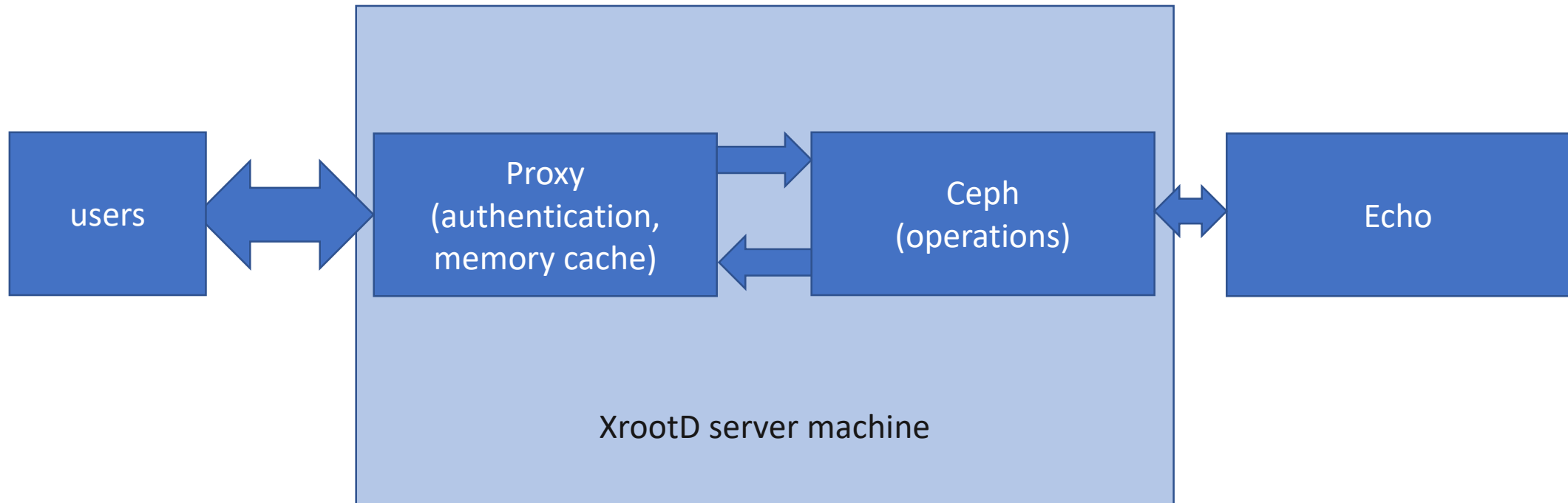
[1] <https://xrootd.slac.stanford.edu/>

# Ceph

- Ceph is an open source software-defined storage solution designed to address the block, file and **object storage** needs of modern enterprises[2].
- In object storage, instead of a hierarchical file system, objects exist in a flat structure with a unique identifier (instead of a file name).
- Highly scalable cloud storage
- Echo - Large erasure coded HDD Ceph cluster for S3/XrootD/GridFTP access at RAL

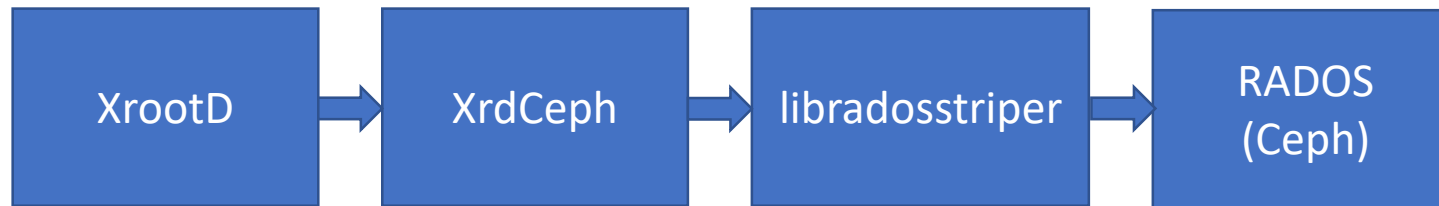
[2] <https://ubuntu.com/ceph/what-is-ceph>

# The proxy/Ceph setup



# XrdCeph

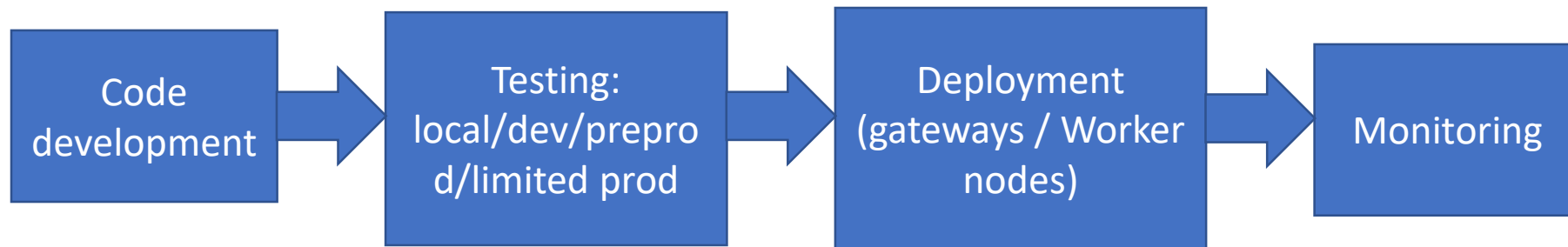
- xrootd-ceph[1] is a Storage layer XRootD plug-in for interfacing with Ceph storage platform. It uses the libradosstriper library to interface with the core storage format (RADOS).



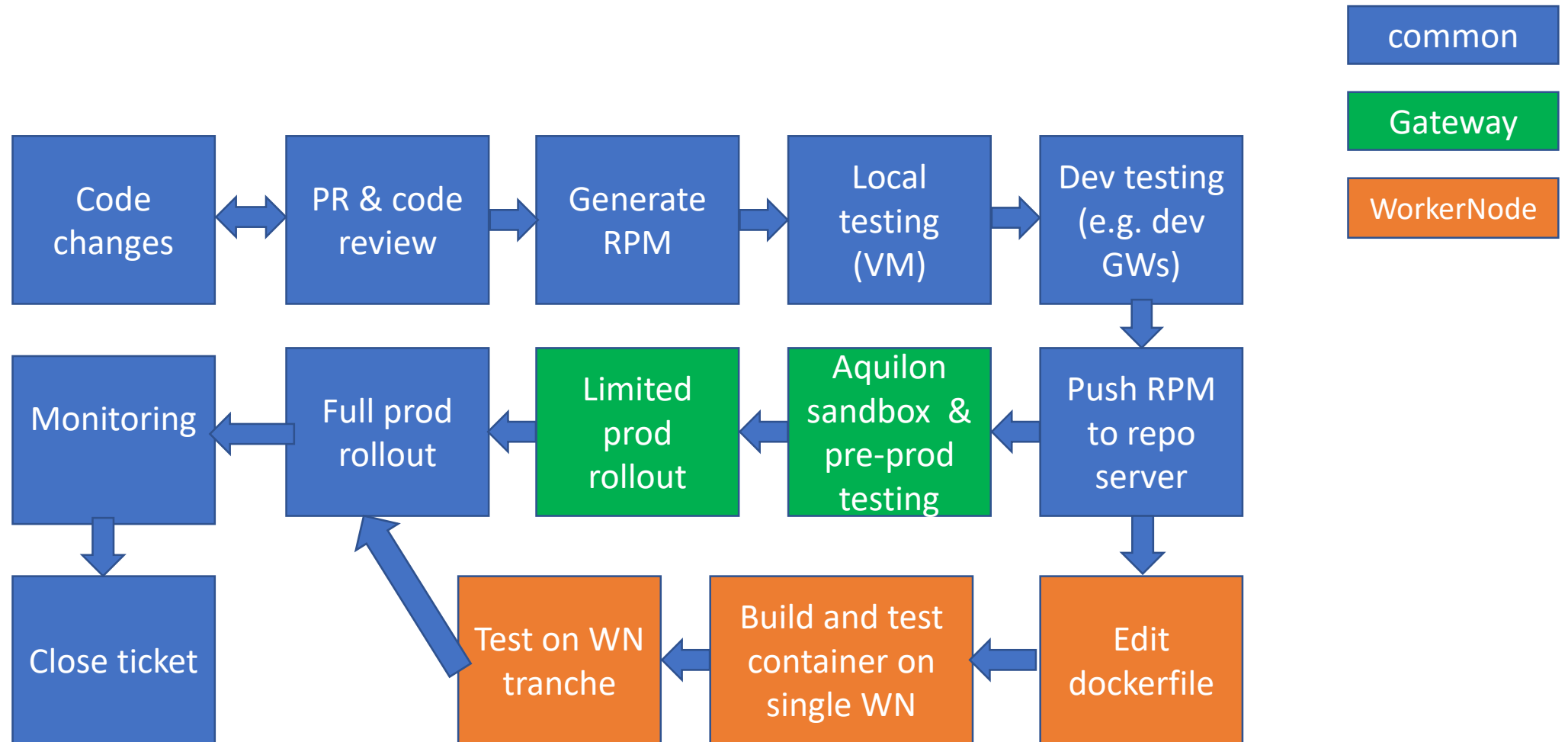
- xrootd-ceph-buffered is a new functionality of xrootd-ceph developed by James Walder that includes I/O buffering at the XrdCeph layer

[3] <https://github.com/xrootd/xrootd-ceph>

# Deployment Cycle at RAL



# In detail



# Upgrade to xrootd 5.5

- Main relevant features introduced between 5.3.3 and 5.5:
  - Scitoken support for authentication
  - Paged io
- Paged IO – perform checksum in flight to reduce transmission errors
  - Will need xrootd-ceph-buffered for good performance



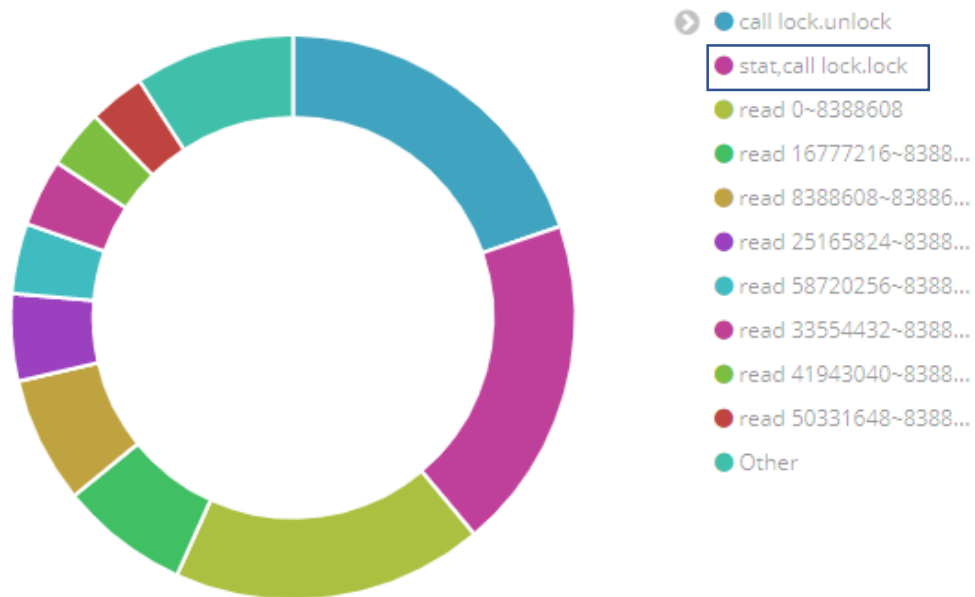
# Current Status

- Vector reads are not working as intended in jobs
- Identified an issue on paged reads
- Difference in stat speed between webdav(extended HTTP) and root
- Identified reason for slow Deletes
- Gateways are generally stable
- Additional gateways introduced
- Added XrdCeph repos for Enterprise Linux 8

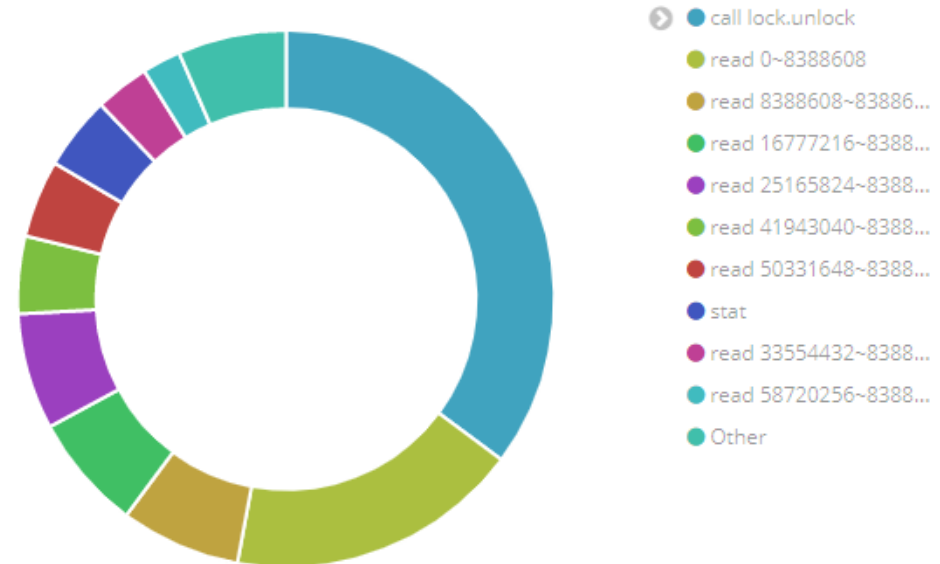
# Vector read issues

- Lockless reads deployed over the worker nodes.
  - No lock metadata operations found in monitoring

Echo - client event detail breakdown



Echo - client event detail breakdown



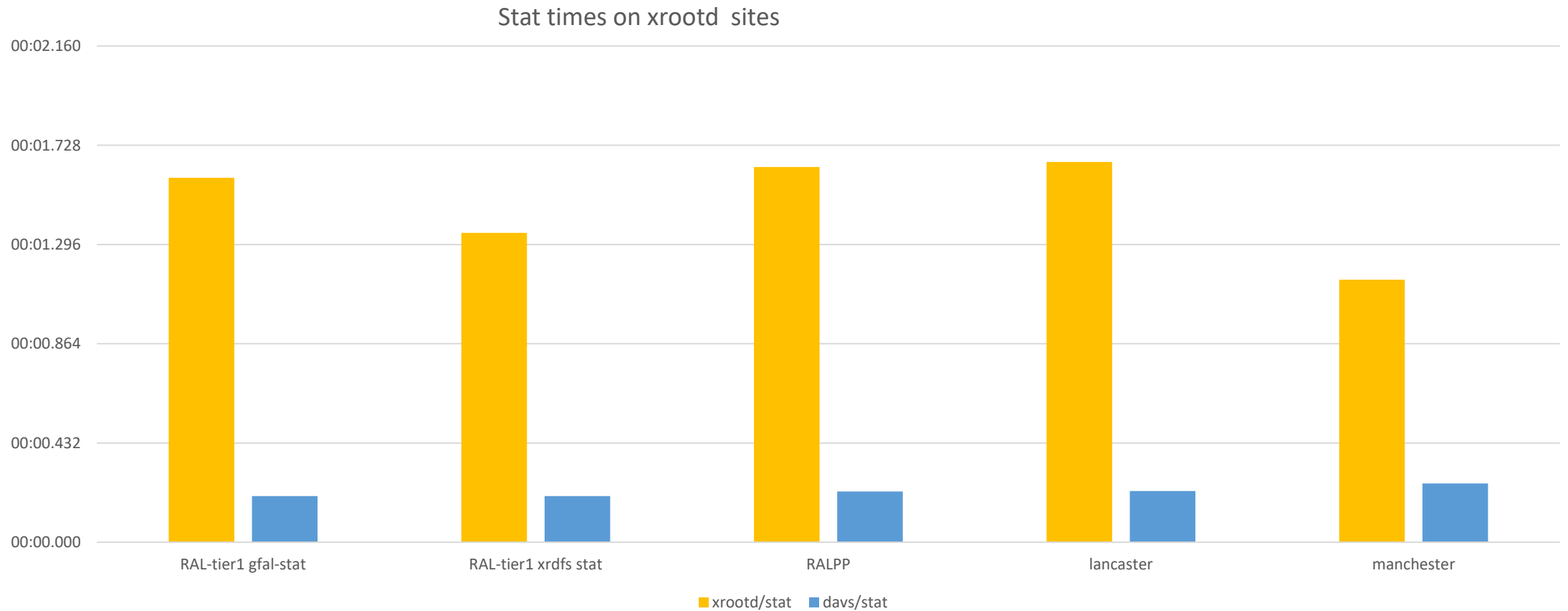
Proportional number of operations, with and without lock operation; the lock operation disappears

- No noticeable improvement on the job success rate

# Paged read issue

- We identified a bug preventing paged reads in Xrootd 5.4.3
- This has been fixed and will be available with the XrootD 5.5 clients

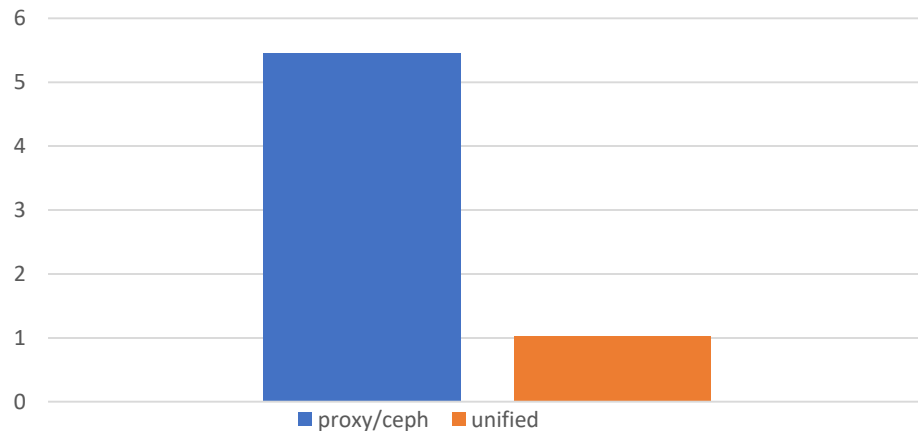
# Difference in stat speed between root and webdav



# Deletes

- Historically VOs used GridFTP for deletes and they never mentioned any problems with it.
- Since switching to XRootD/Webdav it was observed that deletes were taking a long time to process (~20 seconds). It wasn't clear if this was caused by higher load, or the fact that we had changed protocols.
- We checked the XRootD code and it should be able to parallelise deletes.
- After testing it was discovered that the XrootD proxy was serialising them so we are going to remove that.

Average time for test deletes



# XrdCeph for EL8

- Versions 5.3.5 and buffered-5.3.5 are available on el8
  - Available at [http://repos.gridpp.rl.ac.uk/yum/xrootd-ceph/nautilus/el8/x86\\_64/](http://repos.gridpp.rl.ac.uk/yum/xrootd-ceph/nautilus/el8/x86_64/)
- Workernode containers with EL8 xrootd will be deployed soon

# Plans and features

- Switch to buffered IO as the default version of XrdCeph
- Improve stability and error handling
- Unified proxy and ceph service setup
- Upgrade to XrootD 5.5
- Additional monitoring
- Implement vector reads at the xrdceph layer
- Scitoken support
  - Waiting on issue #1763 merge in xrootd 5.5 for wlcg-testing
- Evaluate use of xrootd redirectors for load balancing
- Merge libradosstriper into XrdCeph
- StatLS implementation for XrdCeph
- CI and testing improvements

# XrootD-Ceph Buffered

Asynch	Buffer	Root/webdav	Write	Read
No	No	root	28 MB/s	113MB/s
No	No	webdav	32MB/s	134MB/s
Yes	No	Root	27.7MB/s	146MB/s
Yes	No	webdav	37MB/s	128MB/s
No	Yes	Root	60.24MB/s	128MB/s
No	Yes	Webdav	61MB/s	146.3MB/s
Yes	Yes	Root	64MB/s	128MB/s
Yes	yes	Webdav	63MB/s	126MB/s

Tested on an EL7 VM with unified setup and XrootD 5.3.3. Asynch segsize of 67108864



# Improve stability and Error handling

- Covering interaction bugs between xrootd, xrootd-ceph and clients
- Testing latest xrootd versions while release candidates are in the works
- Additional periodical check for xrootd status – monitor socket states

# Unify proxy and ceph services

- Reduce cross-talk within xrootd
- High CPU loads in split setups due to memory cache
- Harder to debug/identify issues
- Easier to manage permissions

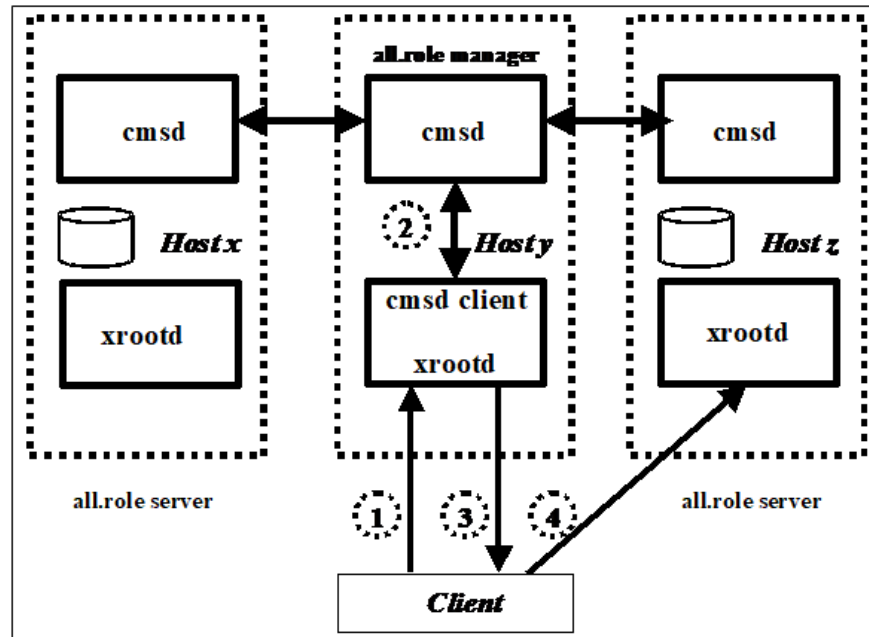
# Additional monitoring

- Currently RAL has monitoring that covers:
  - Xrootd state check
  - Bytes I/O
  - Load
  - Memory usage
  - Number of xrootd connections
  - TCP socket states
  - file operations.
- In addition to these, we made an additional monitoring for the outputs of xrootd reports [2]

[4] [https://xrootd.slac.stanford.edu/doc/dev54/xrd\\_config.htm#\\_report](https://xrootd.slac.stanford.edu/doc/dev54/xrd_config.htm#_report)

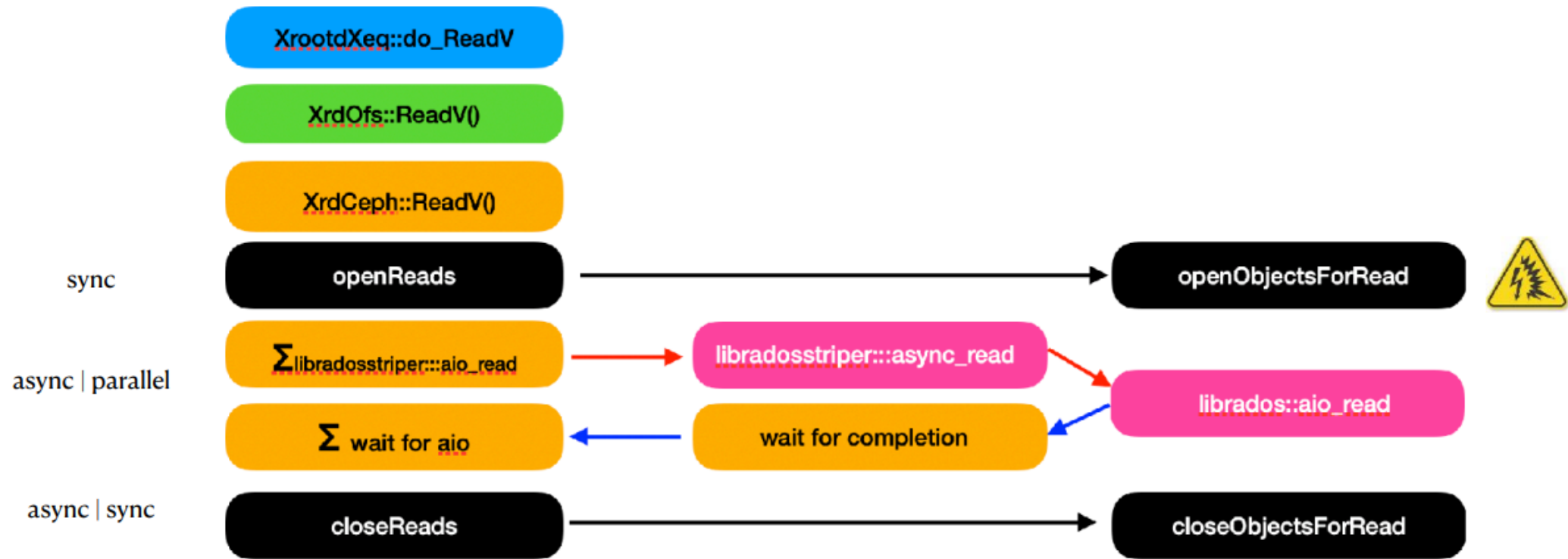
# Redirector load balancing

- To investigate efficiency of redirectors with a unified setup compared with the DNS round robin currently in use to balance load between gateways



[https://xrootd.slac.stanford.edu/doc/dev54/cms\\_config.htm#\\_Toc53611049](https://xrootd.slac.stanford.edu/doc/dev54/cms_config.htm#_Toc53611049)

# Implement vector reads at the XrdCeph Layer



# Merge libradosstriper with XrdCeph

- The goal is to have a single code base combining libradosStriper and XrdCeph, which could be easily maintained by RAL. We would like to maximise throughput by parallelising operations / transfers where possible.

# StatLS (spaceinfo and query space support)

- Allows users to query the current and available space in each pool
- Functionality implemented in XrdCeph 5.3.8, needs unified setup for the right permissions
- Example:

```
[root@host-172-16-105-133 ~]# xrd fs 172.16.105.133:1094 spaceinfo atlas
```

```
Path:      atlas
```

```
Total:     17182000000000000
```

```
Free:      17181999966445568
```

```
Used:      33554432
```

```
Largest free chunk: 17181999966445568
```

# CI and Testing improvements

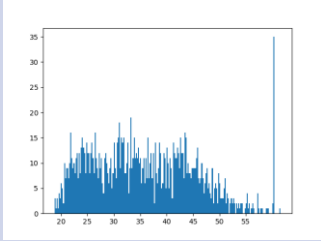
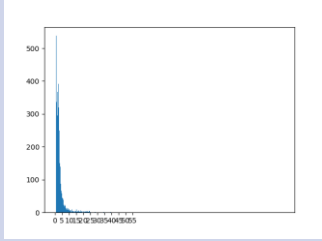
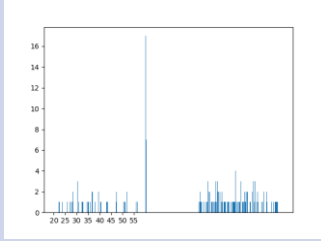
- Implemented scripts to:
  - Automate dependency installation on fresh dev VMs
  - Automate compilation of xrootd-ceph
  - automate and improve RPM deployment
    - Renaming RPMs for version containing alpha features
  - Perform basic functionality tests
- 3 step testing before full deployment:
  - Dev/VM testing
  - Pre-prod testing
  - Single prod machine testing



Thank you

# Vector read issue (more)

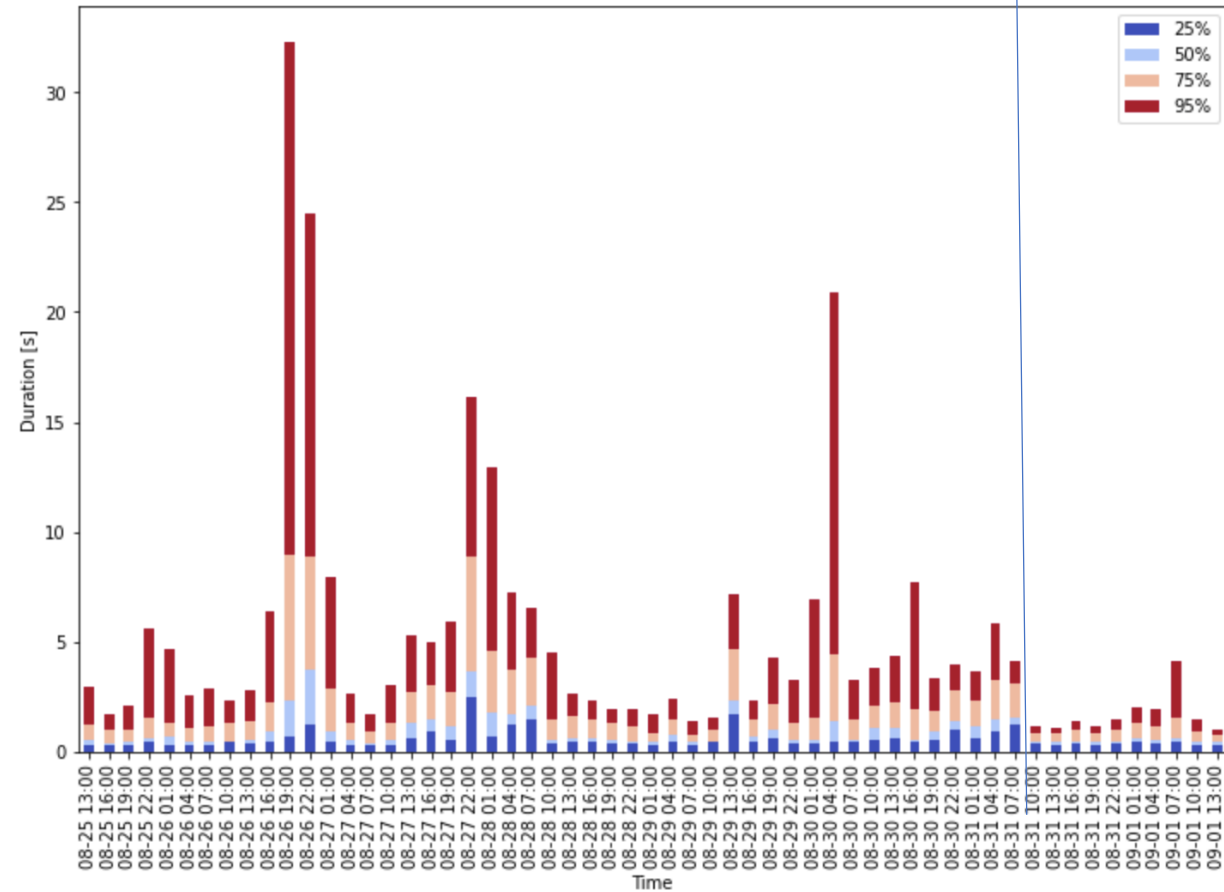
- From testing at RAL, another issue that might be causing the issue was identified to be the XRootD TLS
- This did not result in improved Job success rates.

No memcache/tls	Memcache, no tls	Tls, no memcache
		
Average successful read time: 28.7s	Average successful read time: 3.63s	Average successful read time: ~35s
Success rate: 97.58%	Success rate: 100%	Success rate: 22%

[Full test report](#)

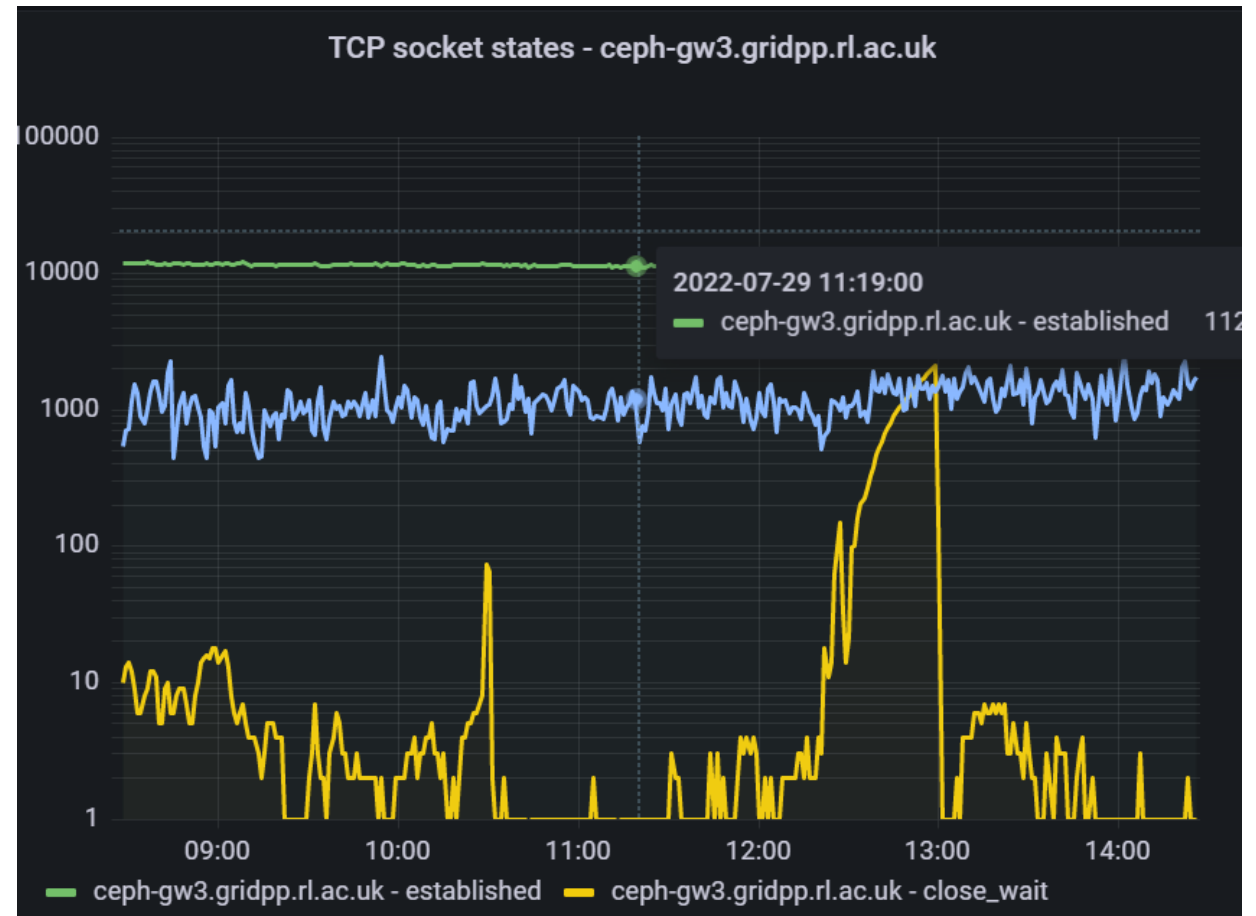
# Delete Speed (more)

Switch to unified



# CLOSE\_WAIT

- Identified by an exponential rise in sockets in CLOSE\_WAIT state and number of XRootD connections
- Client requests fail at this point
- Caused by failure to end sessions
- Gets fixed on service restart



# Servers using XRootD

- XRootD gateways
- Webdav gateways
- Alice gateways
- Worker nodes
  - Xcache instead of proxy
  - Perform reads in xcache, writes on external gateways
- CMS AAA redirectors

# Paged read issue in details

- Identified bug that resulted in incorrect system [readV](#) call (iovcnt greater than maximum), causing clients to crash when requesting preads.
- Fixed client-side in 5.5 with commit <https://github.com/xrootd/xrootd/tree/26eb65159521fd4bfe6b557f933ee1cc9bd34411>

# Debugging

```
#include <fstream>
{
using namespace std;
ofstream myfile;
myfile.open ("/tmp/debug2.txt", std::ios_base::app);
myfile << "example:"<<var<<"\n";
myfile.close();
}
```