A Blueprint for a Contemporary Storage Element

The WLCG Tier-2 Site "UKI-NORTHGRID-LANCS-HEP" at Lancaster University required a new solution for the site's storage element (SE), with an opportunity to start with a fresh installation "from the ground up".

It was decided that a modular system, with data storage separate from the technologies used to serve the data, would be optimal. A comprehensive monitoring infrastructure is desired, both to provide alerting to problems and allow measurement of performance. All components are desired to be from standard industry or community sources - "off the shelf".

The basic requirements were:

- Data Server: Provide at a minimum an https endpoint, and be capable of authenticated, integrity checked third party transfers.
- Data Storage: Allow for node-level failures without data loss, be capable of self-repairing and provide a mountable, posix-like interface.
- Monitoring: Real-time metric measurement and alerting for all components.
- All: Cope with the data access requirements for the site, in both the transfers to and from the site and feeding the 8000 job slots Lancaster provides to WLCG VOs.

The chosen solution was an XROOTD cluster, mounting CEPHFS , monitored by a combined Prometheus/Loki stack.

Why Choose Ceph?

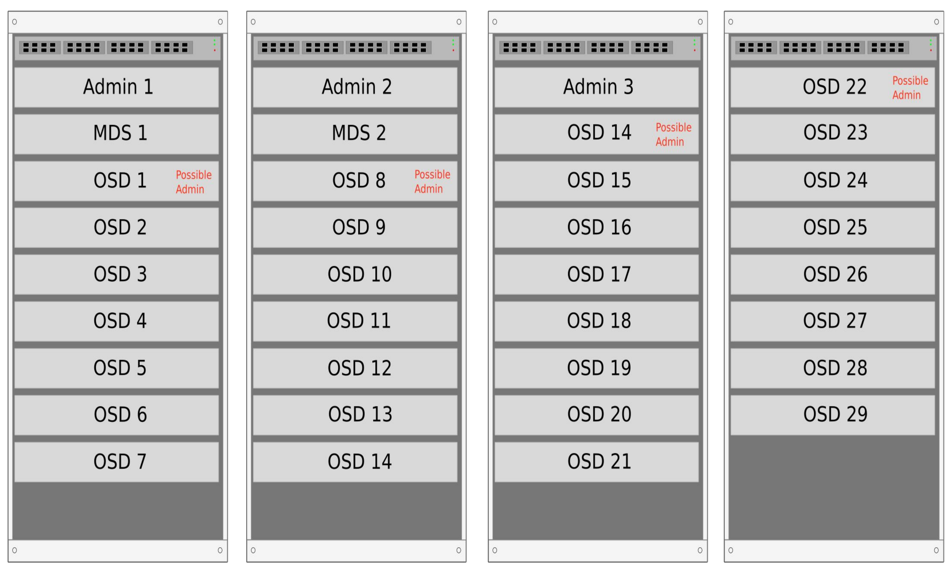
Ceph is capable of scaling easily without down time. It is designed to run on commodity hardware and mitigates the risk of single point of failure.

Installation and Configuration

Cephadm was used to install Ceph Pacific. Ceph orchestration is used to configure and control administration of the nodes. Configuration of all the nodes was straight forward with the use of a few YAML files and completed without any issues.

Looking at the Ceph Architecture diagram below, the installation of Ceph looks like a typical installation. There are a couple of things that aren't obvious from the diagram. (Limited cabinet space for the cluster and alternative OSD configuration)

The cabinet space limitations, means we are limited to four cabinets for the Ceph cluster, which isn't ideal. In the event of a rack level failure the cluster will switch to a "read-only" state.



The admin nodes are in separate cabinets to avoid losing more than one admin node in the event of a cabinet level failure. An OSD node in each cabinet has been designated as possible a manager+monitor node in the event of an admin node failure. If an admin node should fail, the orchestration will automatically select an available OSD node and install the monitor and manager services, thus restoring the minimum required number of monitor+manager nodes. Once the admin node has been restored, the OSD node can be returned to being just a OSD Node. We chose to use the minimum of 3 admin nodes to maximise the available storage using the available budget.

The OSD nodes are fitted with 2x1.6TB NVMe drives. Instead of configuring these drives to use RAID, we assigned 12 OSD Daemons to each NVMe drive which not only increases the amount of available space for the OSD DB/WAL but also increases the life expectancy of the drives compared to using them in a RAID Configuration. (See OSD Configuration below).

Ceph Hardware

OSD Nodes

- 2 x Intel Xeon Gold 5215 20 cores/40 threads
- 256GB RAM
- 2 x 480GB SSD RAID for OS
- 2 x 1.6TB NVMe for OSD DB
- 24 x 16TB (384TB) HDD for cluster data
- 25GbE Network Interface

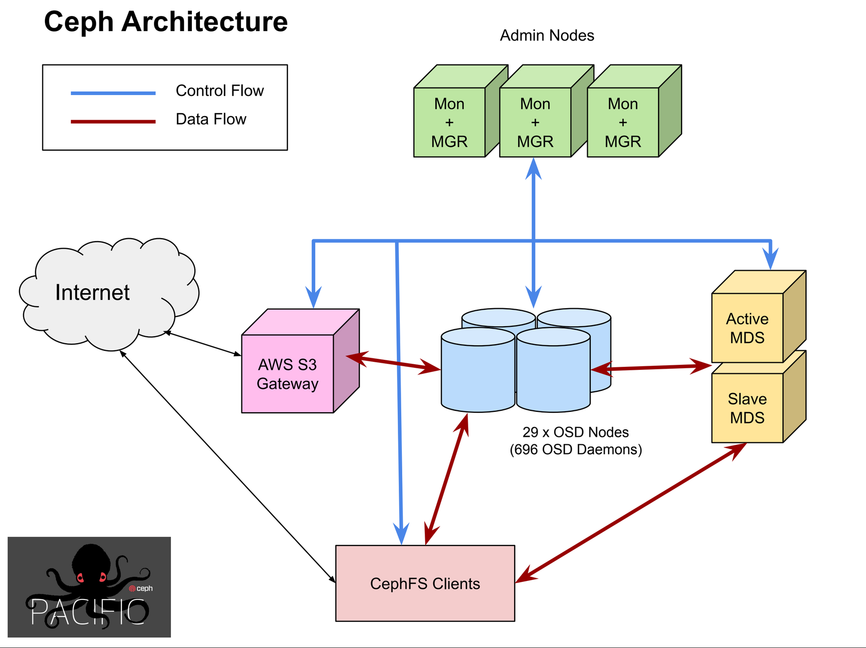
Admin Nodes

- 2 x Intel Xeon Gold 6248 – 40 cores/80 threads
- 192GB RAM
- 2 x 480GB SSD RAID for OS
- 2 x 2TB HDD for logging/Ceph data
- 25GbE Network Interface

AWS S3 Gateway

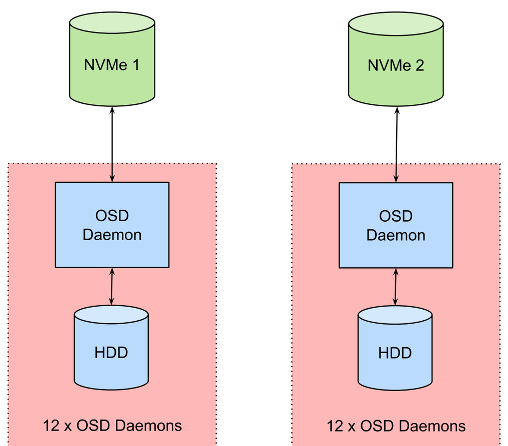
- Virtual Machine, 4 core CPU
- 8GB RAM
- 1 x 80GB HDD

Ceph Architecture



OSD Configuration

Each OSD node has 24 HDDs for object storage and 2 NVMe drives for the OSD daemon DB/WAL. The NVMe drives are used separately rather than being configured using RAID. Each NVMe is configured to store the DB for 12 HDDs. The advantage of this approach compared to RAID is that the amount of data written to each NVMe is reduced by 75% and effectively increasing the life expectancy of the NVMe by 4x. The disadvantage is that when an NVMe drive fails the cluster automatically loses 12 OSD daemons.



CephFS

The existing software (XRootD, Arc, etc) is designed to interact with a POSIX file system. Ceph provides a POSIX-compliant interface via CephFS. Using CephFS has proved to be simple to configure and use on the client machines.

There are two options when using CephFS on the client machines: The FUSE client and the kernel client. We have chosen to use the kernel client as it provides better performance than the FUSE client. We are running CentOS 7 on the client machines which only provides Ceph Octopus and lower version clients. Fortunately, Ceph is designed to allow Octopus clients to work with a Pacific cluster.

Administration

Since going live with the Ceph Cluster, we have had a few hard drive failures. We have also needed to apply security updates and restart all the machines in the cluster. Both of these actions were performed with the cluster remaining up and caused no noticeable impact from the users' perspective.

Why Choose XRootD?

Our data server was required to have the following minimal functionality

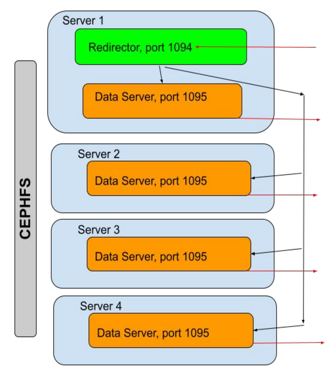
- https/davs access
- TPC support
- VOMS authentication
- Flexible scalability
- (For future-proofing) Token authentication support

The data server solution that filled all these criteria was XRootD [2].

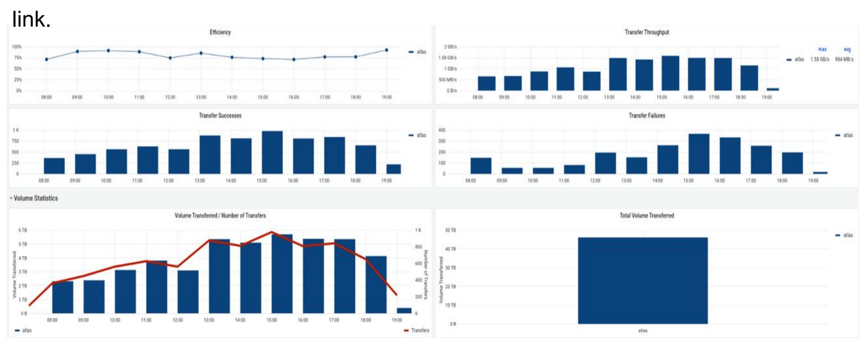
XRootD is a well known and well established software framework for scalable and fast data access, originating from SLAC. There is a large amount of expertise and experience in deploying, tuning and operating XRootD instances in the UK and the WLCG communities. In addition, XRootD is being adopted by communities and fields beyond High Energy Physics - such as astronomy.

Redirector Setup

A single XRoot host alone cannot provide the throughput to utilise the available site bandwidth of 25Gb/s. Transfers, and in particular the post-transfer file integrity check, are resource-intensive - particularly using a lot of RAM. This was confirmed in a test involving the rapid transfer of ATLAS data to the site [3].



In order to overcome this, we deployed XRootD in a redirector/cluster fashion. At the time of writing this was 4 nodes - but as noted in the HEPiX study, the improvements to throughput performance were almost linear in proportion to the number of deployed nodes, and would be possible to scale up until the bottleneck was moved to the site network link.



In the above plots you can see the point where the addition of an extra redirector linearly increases the throughput.

XRootD/CephFS integration - Authentication and ACLs

The XRootD services all write and read files under the "xrootd" user. Access control for xroot or https/davs is controlled through the xrootd authdb.

For users accessing through the CephFS mount, this posed additional challenges. One was how to prevent unrecorded writes, or even more worrying, deletions. The best solution to this problem was straight forward - the CephFS mount across the WNs was made to be read only.
To prevent users reading from other directories we turned to extended ACLs, to provide group level control over reading of directory trees.

This posed one more problem - DAV directory creation (MKCOL) doesn't allow for setting ACLs or permissions, and directories created via DAVs didn't inherit from their parent directories.
The solution to this final problem was to use the xrootd "ofs.notify" functionality to call a "fix permissions" script on each mkdir event.

In future releases of xrootd, there are plans to switch to a multiuser capacity - we look forward to integrating any developments in this area

XRootD Server Hardware Requirements

Our servers are across a diverse site of physical nodes, some purchased for the role, others repurposed.

The experience gained suggests for a high-performance XRootD installation the most important hardware consideration is RAM. Our most performant (least loaded) servers are repurposed ZFS servers with 192GB of memory. Conversely, the redirector service is incredibly lightweight.

We have near-term plans to move our redirector onto our institution's resilient virtualisation platform.

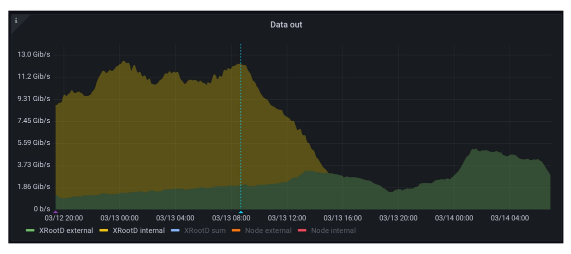
Plot showing the load for all XRoot Server (xrootd) processes
Note that the redirector load is so low as to be barely visible.

XRootD/CephFS integration - Rucio Symlink

Most of the data at Lancaster is for ingesting by atlas jobs running on our WNs. During the initial deployment, these jobs just accessed their files over the internal xroot. As the files were available through posix access on the WNs cephfs mount this was incredibly inefficient!

In response, the UK Atlas Cloud Manager James (re-)implemented a rucio module to simply symlink to the file's location within the cephfs mount, rather then copy the file locally. This greatly reduces the internal traffic accessing our xrootd servers.

For future development, native xrootd "redirect to file location" functionality would allow more efficient access for non-rucio using VOs.



Plots of internal network accesses to the XRoot Server, showing the clear dropoff when atlas jobs were switched to using symlink access.

Monitoring stack

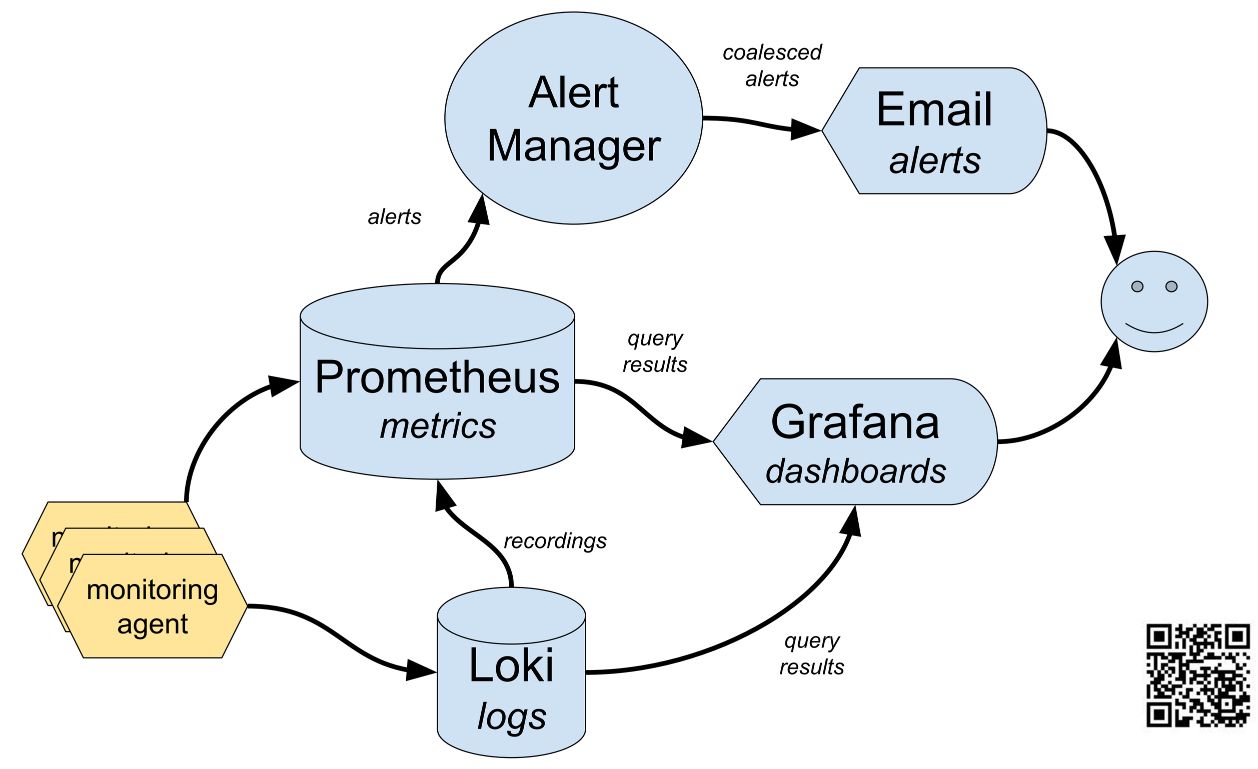
The installation is monitored using a Prometheus time-series DB and a Loki log aggregator, populated using various passive and active monitoring agents co-located with storage and gateway components. These include the Prometheus Node Exporter, Loki PromTail, Ceph's in-built exporter, and several custom scripts [5] to work with XRootD's f-streams and summary reporting, and with Ceph's presentation of SMART disc health metrics. Grafana provides the visualization of live and historical data to the operator, while AlertManager manages live delivery of Prometheus-generated alerts via email and potentially other notification systems.

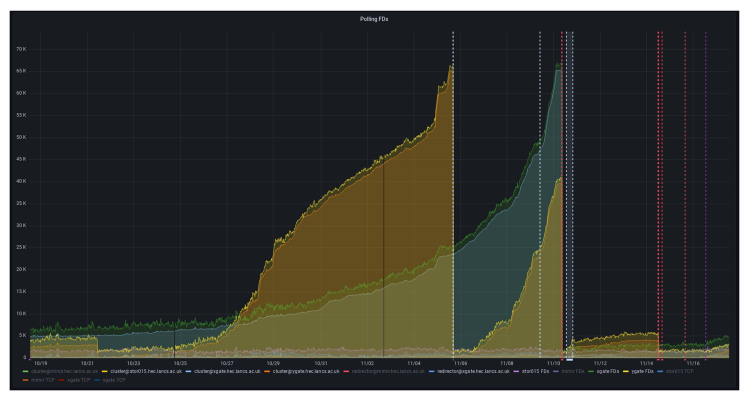
Challenges were addressed mostly by custom monitoring agents:

- Translating XRootD and SMART metrics for Prometheus
- Matching XRootD's push of metrics to the traditional pull/scrape of Prometheus—Uses Prometheus remote-write interface, which Loki also uses for recordings
- Representing static expectations and relations—YAML configuration describes network/interface topology, services and certificates expected, number of Ceph OSDs expected per host, names of expected XRootD instances, etc
- Correlation of metrics from different subsystems—Renamed/eliminated some labels in Prometheus configuration
- Timely acquisition of SMART metrics—696 requests via Ceph, each taking 1-2s; remote-write means each can be recorded individually, as soon as available

Advantages:

- Detection of impending resource exhaustion on XRootD nodes
- Detection of growing defect list and write errors of Ceph OSD discs
- Correlation of disc failures to physical location
    - for replacement
    - for impact from environment (heat, vibration?)





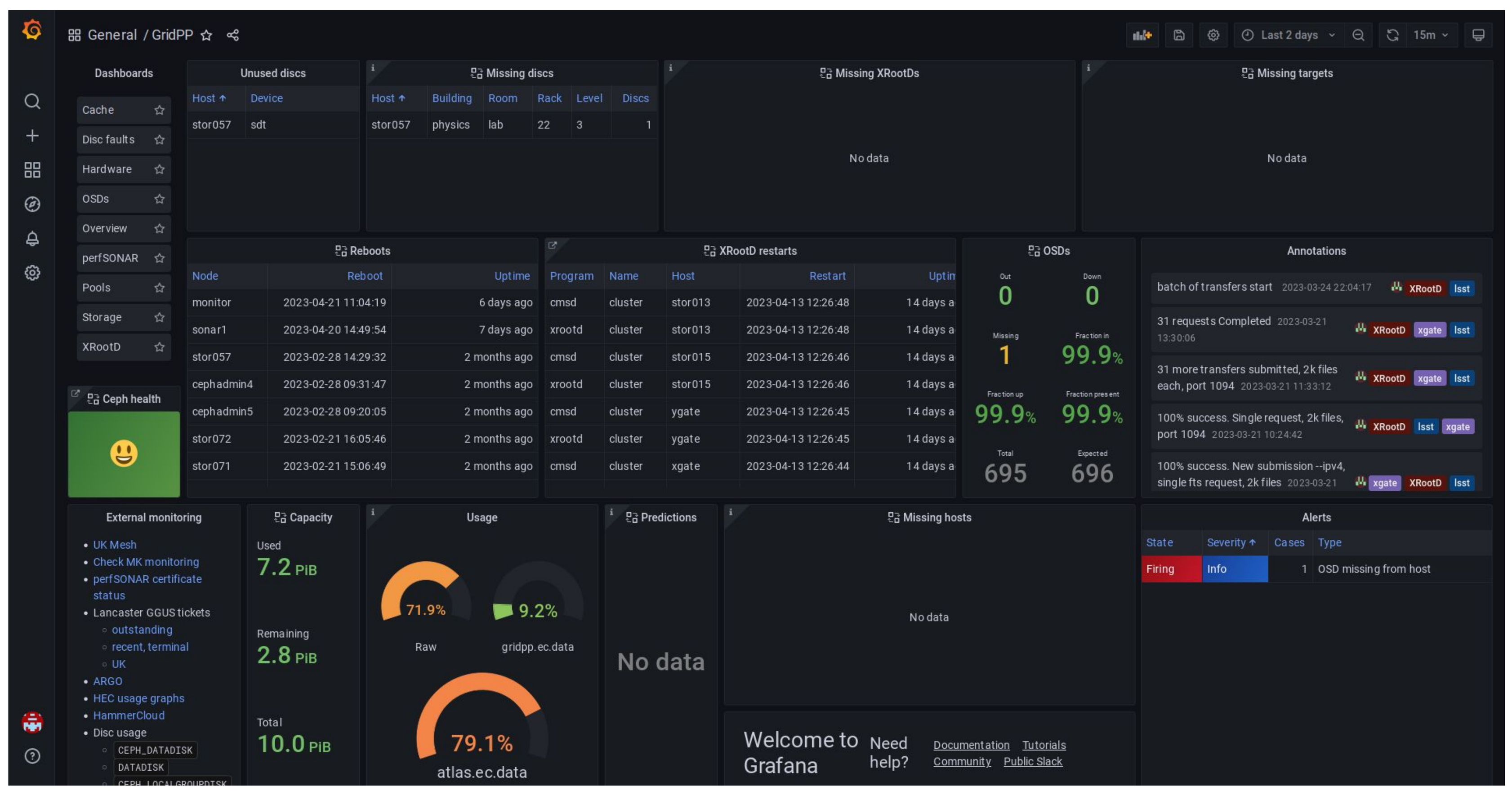
Detection of FD exhaustion

Monitoring helped to spot an irregularly recurring issue with XRootD, whereby it would appear to reach a cap of around 65k file descriptors attached for polling. In the first case, this correlated to a GGUS ticket on many connection timeouts. Restarting the problematic instance mitigated the problem. Monitoring identified which instances manifested the problem, and when, so only faulty instances needed a restart. It also showed that the redirector was unaffected, and that instances running on newer OSes (Rocky8 vs CentOS7) were also immune.

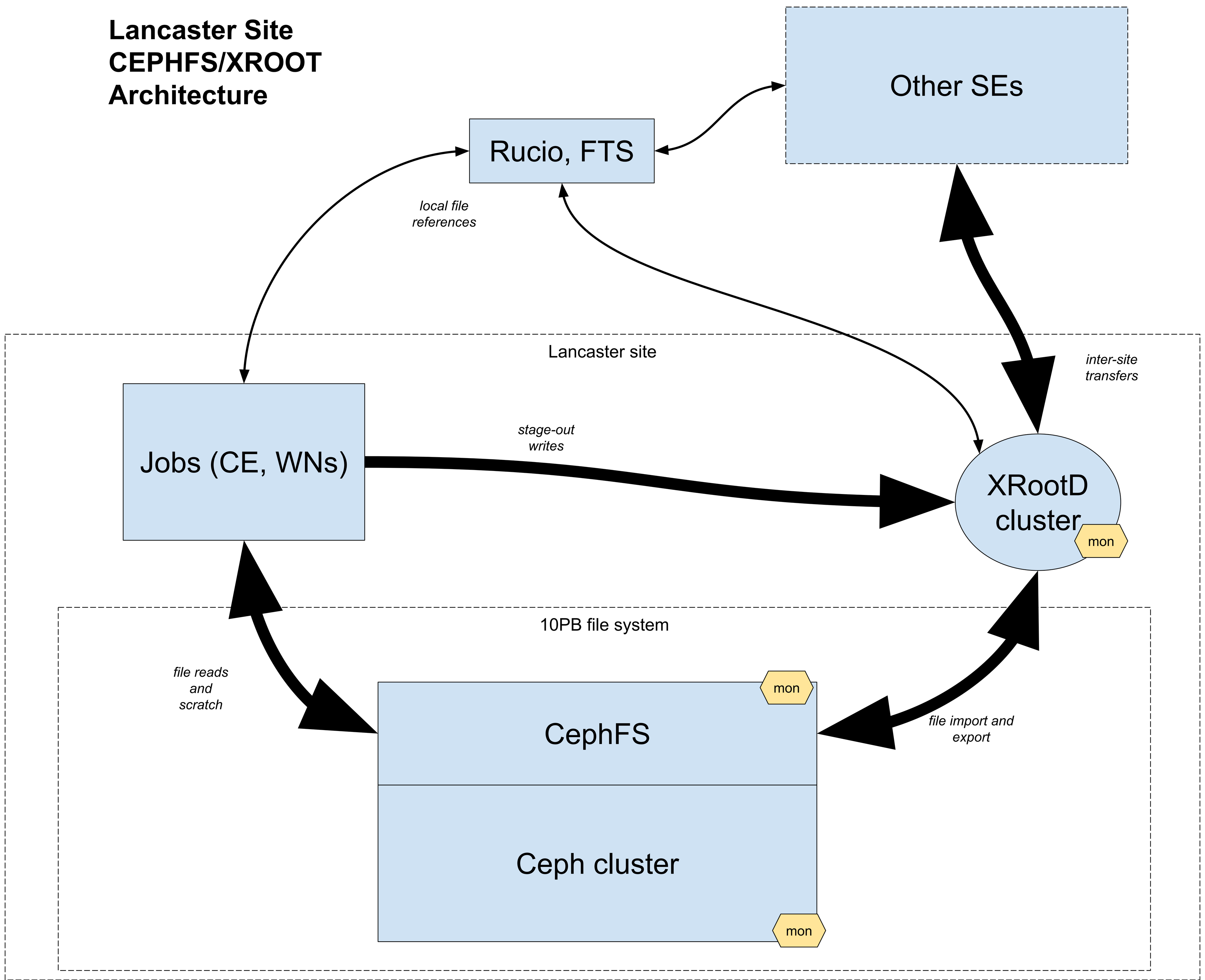
Monitoring console

System health is observable through Grafana dashboards. **Above** shows four OSD hosts, indicating that one of them is missing an OSD. **Below** is the main dashboard, including:

- Over-all Ceph health
- Lists of unused discs/counts of missing OSDs per host
- Unresponsive hosts, XRootD instances and monitoring agents
- Host reboots and service restarts
- Used and available storage capacity
- Recent annotations and current alerts
- Links to other dashboards and external monitoring



Lancaster Site CEPHFS/XROOT Architecture



Architecture

The installation is built around a Ceph cluster providing 10PB of object store, presented through CephFS as a POSIX file system to worker nodes and gateways.

A cluster of XRootD gateways present remote authenticated access to this file system, and through these Rucio and FTS populate it with transfers from other sites. This serves as preparation for jobs on the worker nodes, which consult Rucio to locate local copies of files. Rucio's structural awareness of the site allows it to point jobs at the local CephFS mount, and have them directly read from the file.

The XRootD gateways also perform administration tasks, such as producing lists of stored files or the Storage Resource Reporting json.

Jobs with bulk output write back via the local gateways to enforce the more sophisticated access control that XRootD can enforce for writes. This output can then be delivered under the control of Rucio/FTS to other sites.

CephFS also provides a job data scratch area, replacing NFS in our cluster. This is a separate small pool, mounted r/w.

Gateways and storage are monitored continuously with Prometheus and Loki.

Benefits:

- Reliable/failure-tolerant storage supporting on-line hardware replacement without file loss
- Authorization integrity maintained both internally and externally
- Inter-site transfers exploiting available bandwidth/capacity
- Efficient read access to files
- Identification of failing discs, disconnected hosts, exhausted resources
- Using CephFS allowed XRootD machines and Compute Nodes to continue using same software.

References

[1] Ceph
[2] XRootD
[3] HEPiX talk
[4] Lancaster Gridsite xroot configs
[5] Custom monitoring scripts
[6] Storage Resource Reporting Script

Physics | Lancaster University | GridPP UK Computing for Particle Physics

Presented by Roger Jones at CHEP2023. Produced by Steven Simpson, Peter Love, Gerard Hand and Matt Doidge. With thanks to Sam Skipsey (Glasgow University) and James Walder (RAL STFC).