Exploring XRootD-PFC Optimizations Using Advanced Monitoring in the UK

Robert Currie (**rob.currie@ed.ac.uk**), Wenlong Yuan (**wenlong.yuan@ed.ac.uk**)

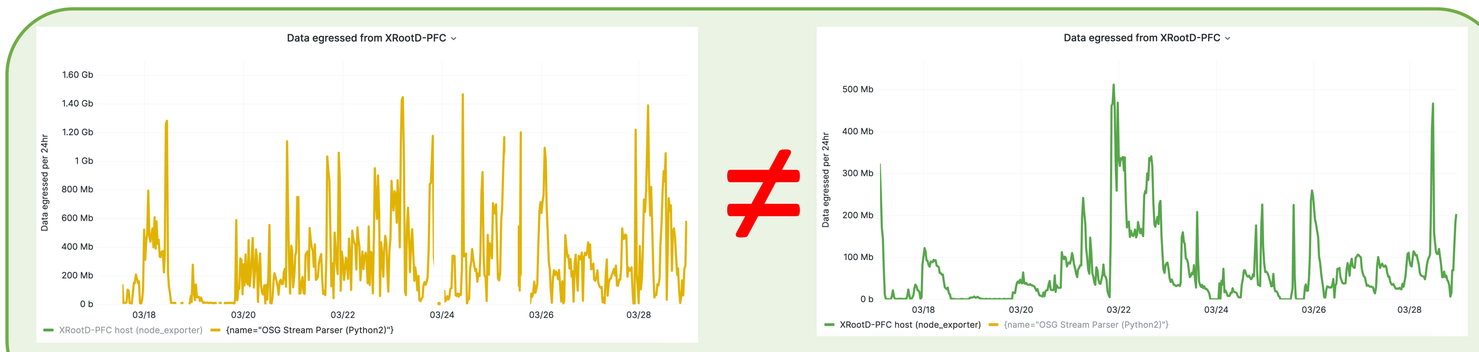
THE UNIVERSITY of EDINBURGH

Background

As presented at previous CHEP conferences there is a disjoint between egress rates **broadcast** from XRootD and raw monitoring metrics extracted from **host monitoring**.

To determine the source of these differences additional studies were identified as needed.

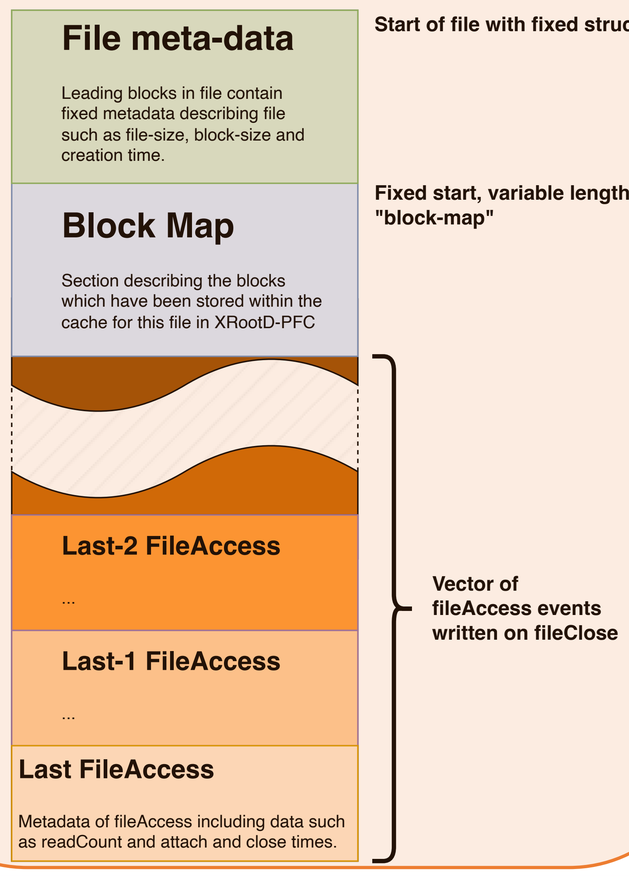
As well as working to understand these differences we have also explored using this data to optimize and tune our service configurations.



Broadcast XRootD data egress

Egress data from node_exporter

.cinfo File Structure



Understanding Monitoring Data

To verify monitoring metrics broadcast from XRootD, a new logging approach was developed at Edinburgh.
This made use of the **on-disk .cinfo metadata** files stored within the XRootD-PFC.
Capturing changes to these files, when the metadata is written to disk, provides a monitoring stream constructed as a cross-check of the messages broadcast by XRootD streams and read by multiple parsers.

Analyzing this data revealed unique insights over earlier studies:

- Granularity for site-specific XRootD-PFC is extremely coarse.
  Grid jobs keep files open for long periods with fileClose events occurring 24hr+ after data transfers.

- Granularity isn't enough to explain differences between node_exporter and XRootD monitoring data.
- Both XRootD and node_exporter can allow for finding and fixing service mis-configurations.

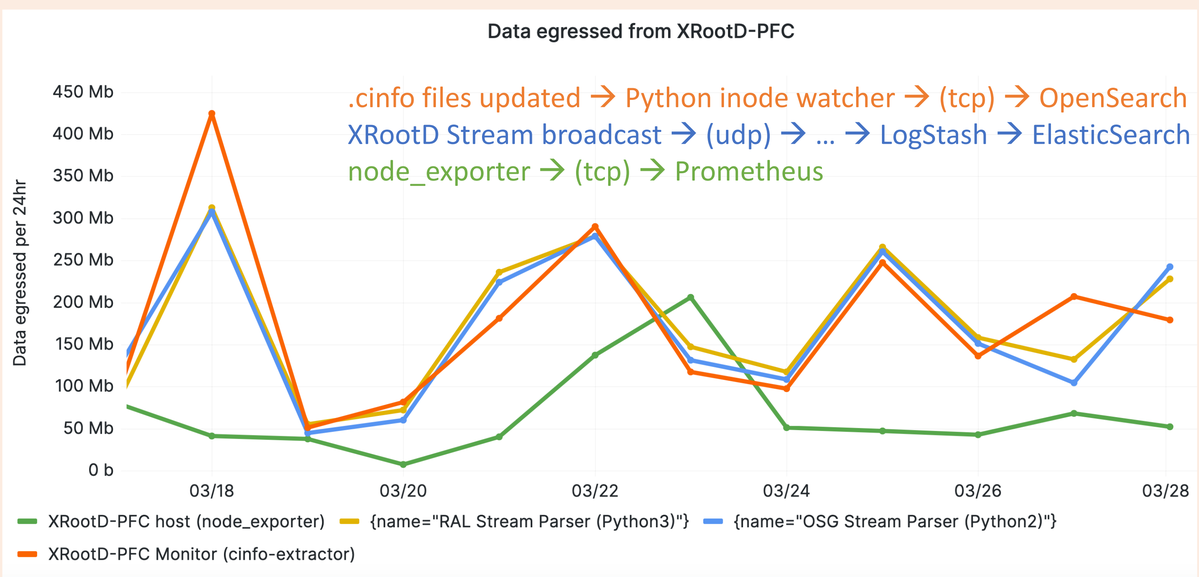
If either the PFC blocksize and prefetch settings were mis-configured both XRootD and node_exporter show the XRootD-PFC service leads to an amplification of data required to be transferred from site storage.

Combining monitoring metrics from multiple sources was possible using the Edinburgh Tier2 monitoring stack as presented elsewhere.
Using this, we have shown there is a consistent difference between XRootD and node_exporter metrics.

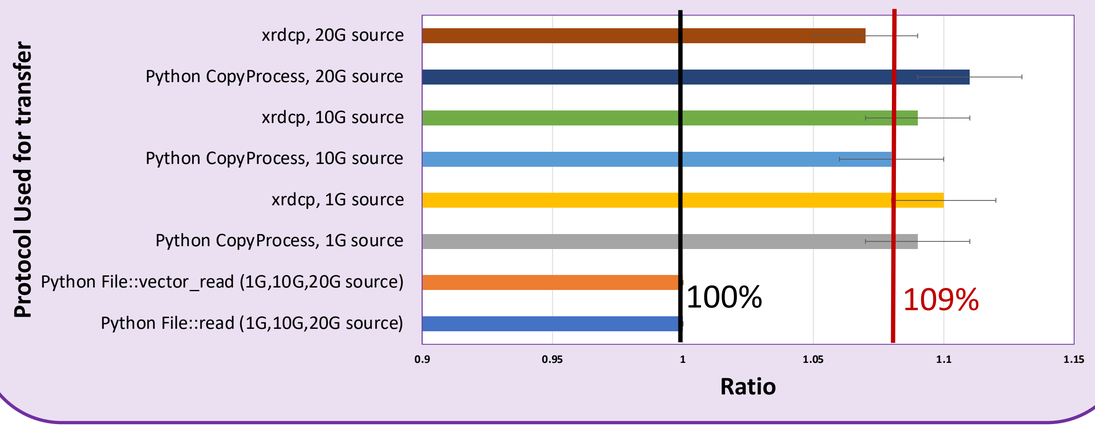
Isolated testing

Isolated testing of an XRootD PFC instance using containers allowed us to analyze the behavior of the monitoring under different situations.
This has revealed that when copying a file using **xrdcp** or **CopyProcess** there is an over-reporting of the amount of data transferred by ~10%.
This appears to match with what we see in production.
This is not observed when reading data using simple read functions in PyXRootD.

Further investigations are needed to understand this in more detail.



Ratio of Data Transferred (cinfo metadata) vs fileSize



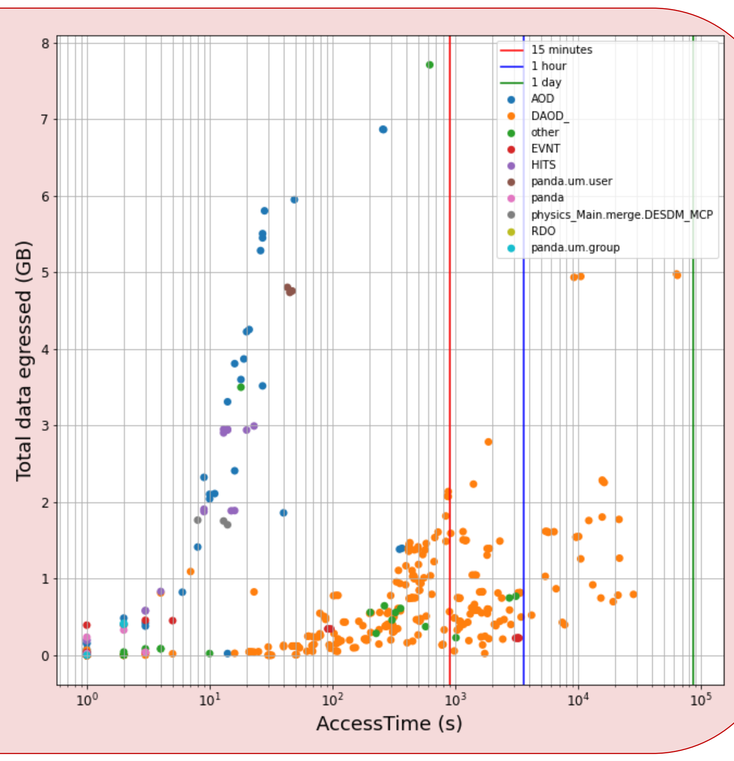
Further Analysis

After verifying the performance of different monitoring methods, we started to begin to analyze the performance of our XRootD-PFC service.

Using tooling built for verifying monitoring metrics, we discovered it is possible to categorize the behavior and performance of our XRootD-PFC in production based upon the filetype information alone extracted from the filename as shown when we examine the accessTime vs fileSize.

Using this we know we can improve the cache performance by not caching files which are just transferred through the cache and that are not repeatedly accessed.

Files accessed over long periods from the cache also tend to have more complex access patterns whilst the connection is open for the full lifetime of a grid job.



Conclusions

Through analyzing the data from our XRootD-PFC instance we have understood some of the major differences between our monitoring metrics and our network monitoring.

This has shown our monitoring of PFC performance in with production jobs has to be performed with the granularity of whole job lifetimes.

Further work is needed to explore why data transfers are being over-reported within XRootD-PFC in our containerized isolated test setup.

We have shown that we are able to tune and improve the performance of our cache by understanding the behaviors of different access patterns relating to different file types.