

XCache

Teng Li

University of Edinburgh

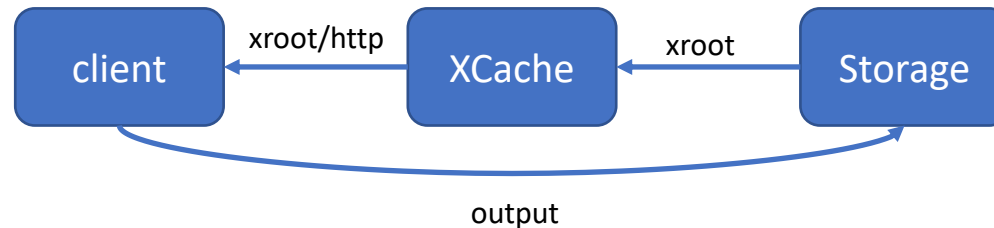
GridPP42, STFC

Outline

- Introduction
- XCache deployment & testing @ Edinburgh
- Simulations
- What's next

Introduction

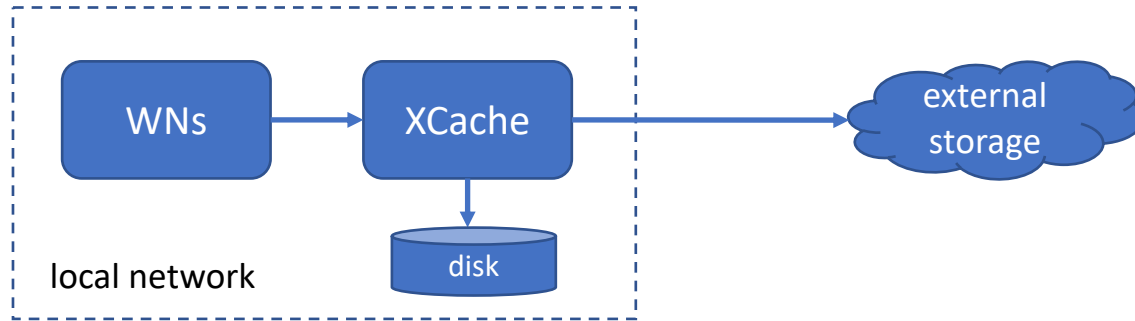
- XCache
 - XRootD server running in proxy mode, with the cache library enabled
 - Every (chunk of) file read through this proxy is saved in the cache disk
 - Speaks xroot protocol (http is also possible between clients and XCache)



- What for?
 - Optimization for bandwidth usage, data access latency, data placement
 - Could be used as volatile storage
- Other features: clusterable, plugin-based, partial file cache

Use cases

- XCache as a transparent cache

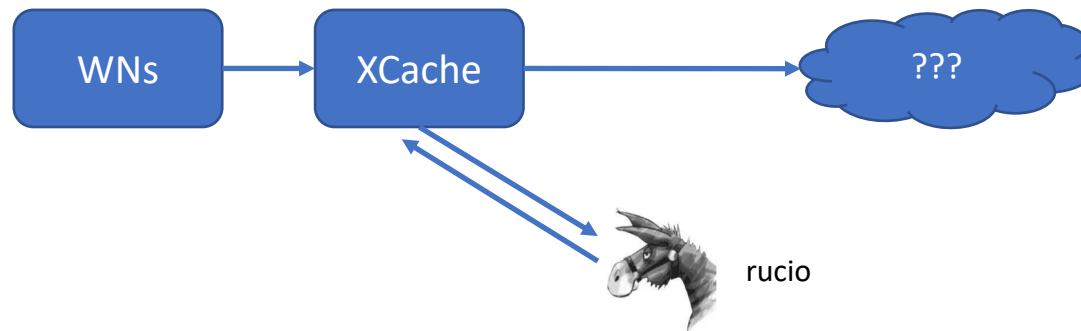


- Useful for site attached to whatever types of external storages (remote SE, storage cloud), which has performance issues of data access. Or if local storage has high latency, e. g. caused by erasure coding
- Other mutants:
 - Server-less cache, XCache running on WNs directly (with a shared filesystem)
 - Memory-based cache, for small sized, rapidly accessed files

Use cases (cont.)

- Typically ATLAS use cases (potentially other experiments using rucio)
- XCache works with rucio
 - Volatile RSE

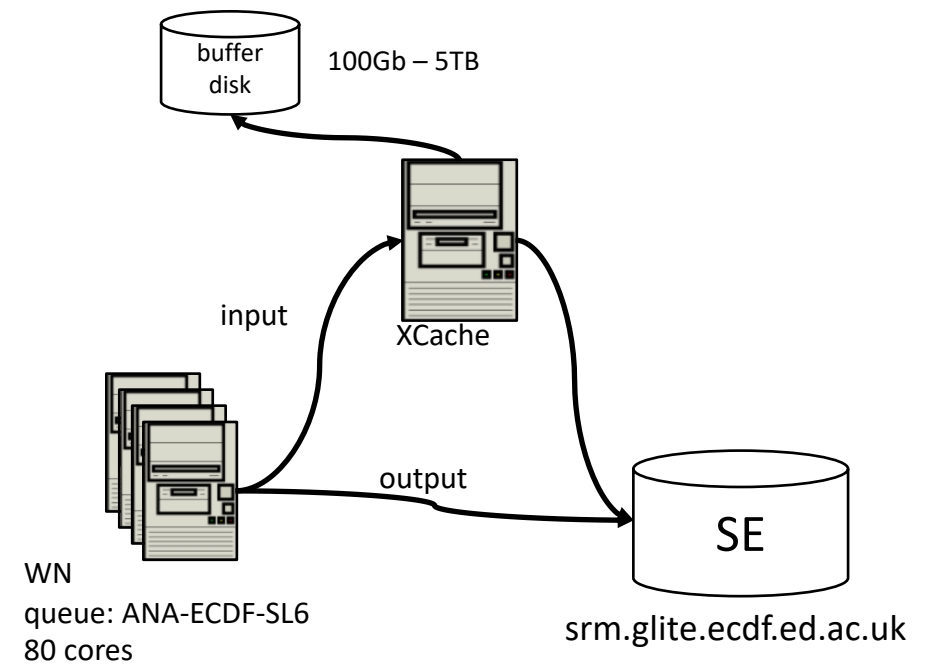
Cached contents are reported back to rucio. Job broker could then send jobs accordingly
 - Rucio Name2Name plugin
 - XCache accepts rucio gLFN and connects to rucio to get the physical location (rucio metalink)
 - Same file with different locations share the same cache entry



- In progress, both relies on the workflow management

XCache deployment & testing

- Aim
 - Testing XCache. Target deployment/ operation issues
 - Study the actual performance and look for optimization methods
- Overview
 - Transparent cache mode
 - Input traffic of WNs is redirected to XCache
 - Output remains unchanged
 - Whole file cache mode is used
 - Tested with ATLAS analysis jobs
 - At small scale (80 cores)
 - Performance is monitored

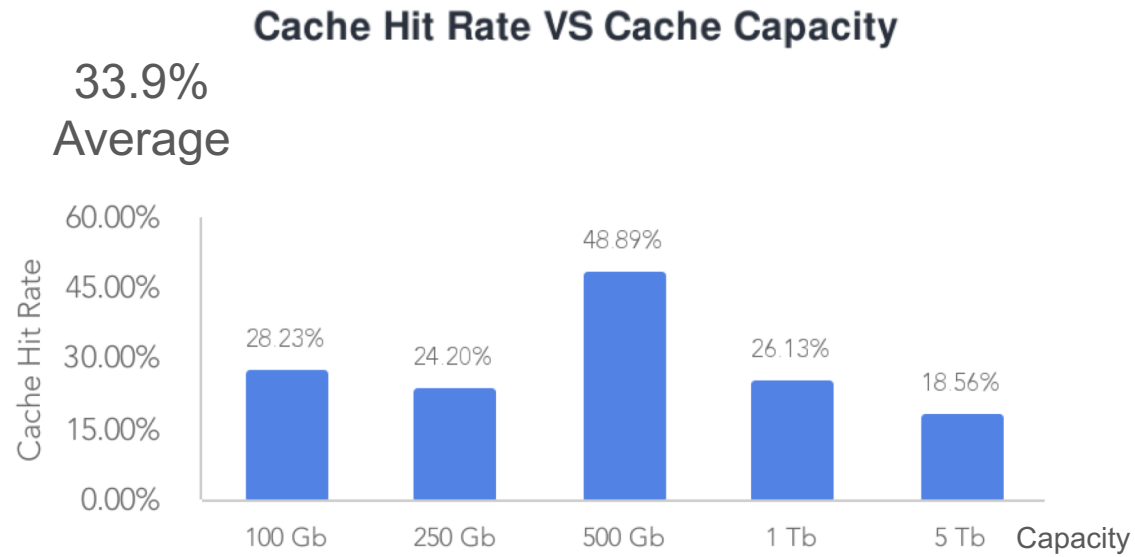


XCache deployment & testing

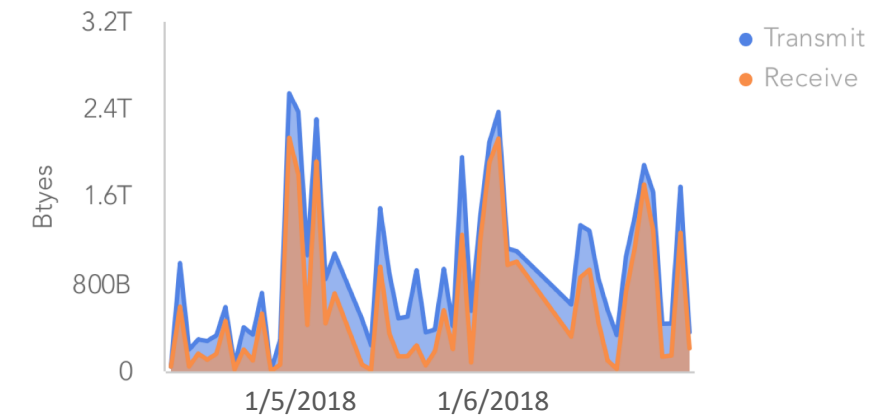
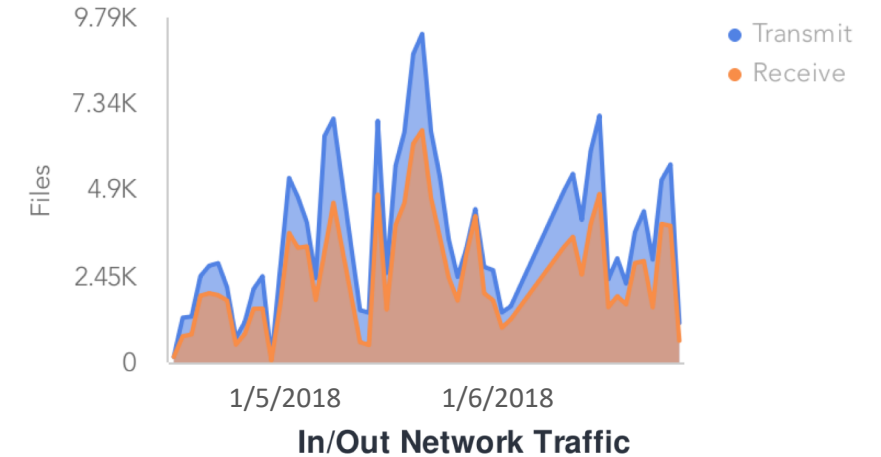
- Summary of the deployment and operation
 - Easy to deploy
 - Light weighted. Could be deployed as container
 - 2 services (xrootd, cmsd), one filesystem, and several cronjobs are enough
 - Easy to maintain
 - Crash twice during 4 months. No actions more than restarting services.
 - Not worried at all about data loss
- 2 major issues
 - Integrate XCache transparently into ATLAS workflow
 - Not fully defined by ATLAS
 - Could be done with the help of AGIS and XRootD name plugin
 - A valid certificate
 - XCache robot certificate is available

Performance

- 4 months of data is taken to measure the cache performance
- Average cache hit rate is 33.9%
(bytes read from cache/all bytes read by WNs)
- Different cache capacities are tested. Peak value reached ~50%.



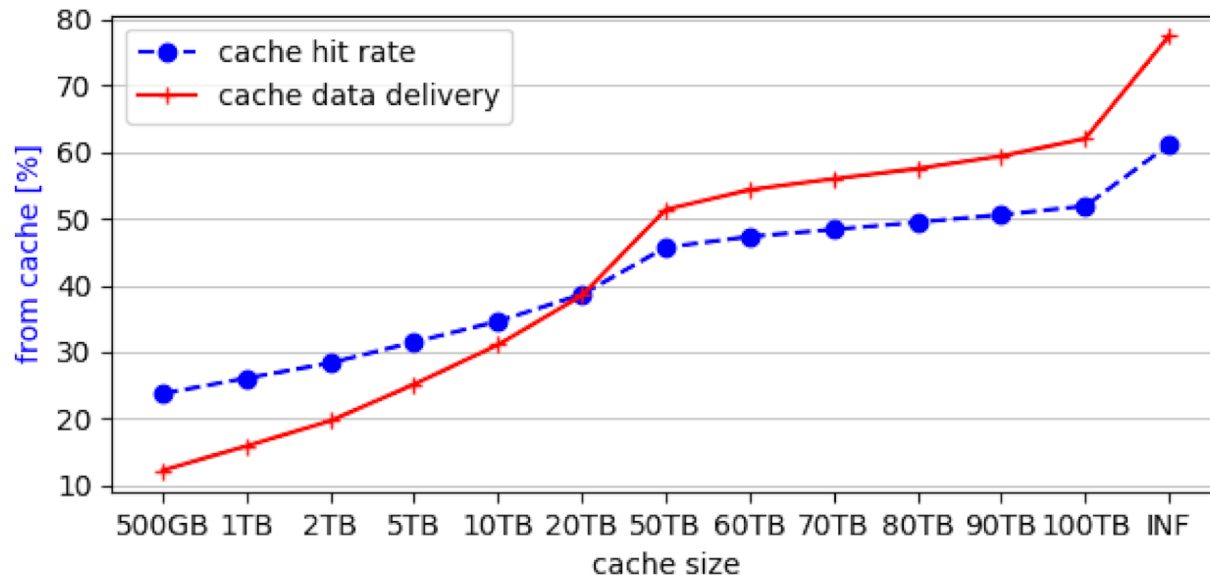
Number of Transferred/Received Files



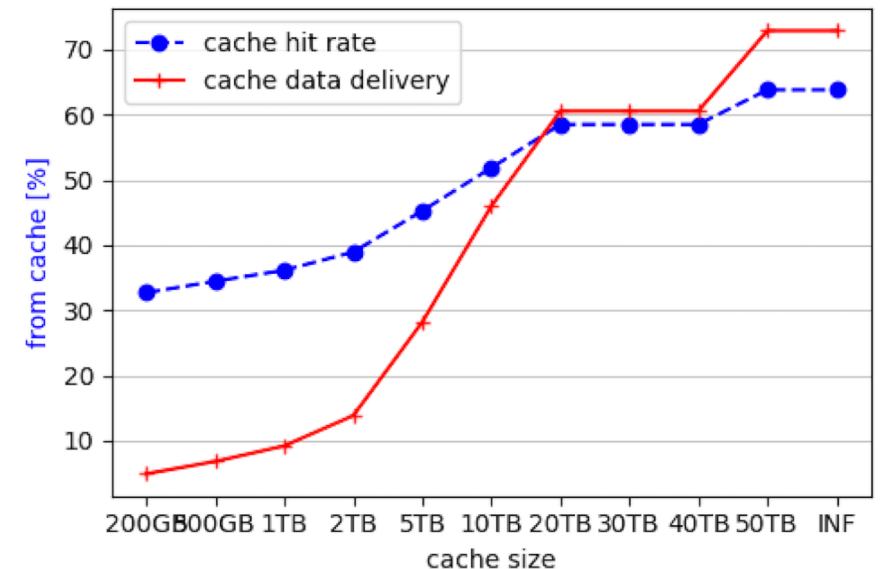
Performance

- XCache simulation as a cross validation
 - Based on the rucio trace data for 6 months
 - Simulate real behaviors of XCache to see the performance
 - Comparison of ECDF analysis queue and BHAM production queue

ECDF Analysis queue



BHAM Production queue

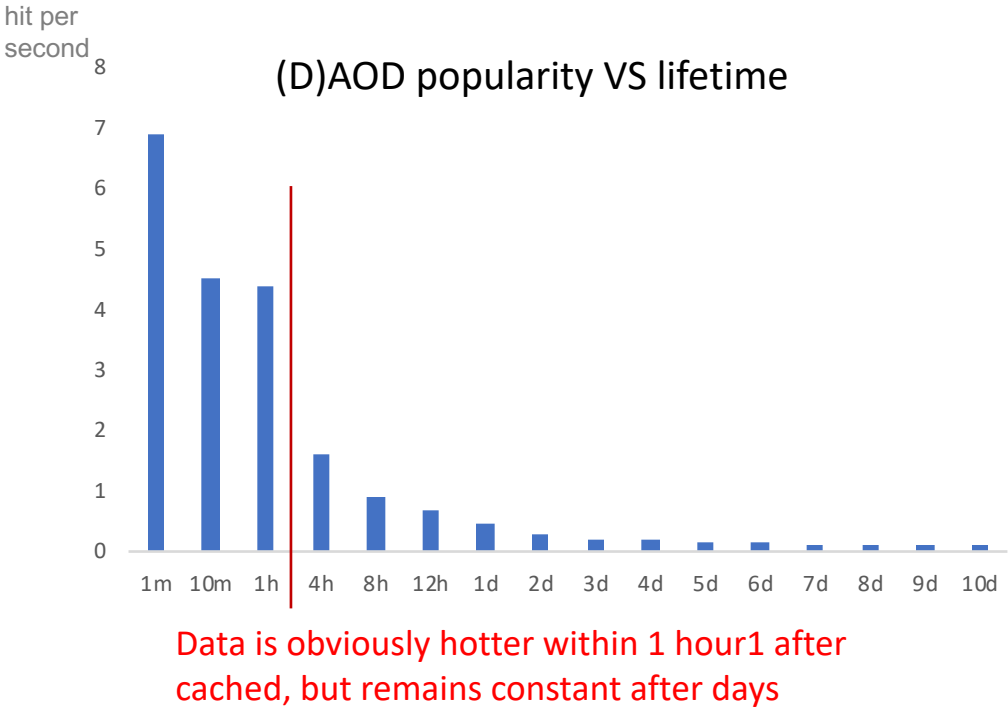
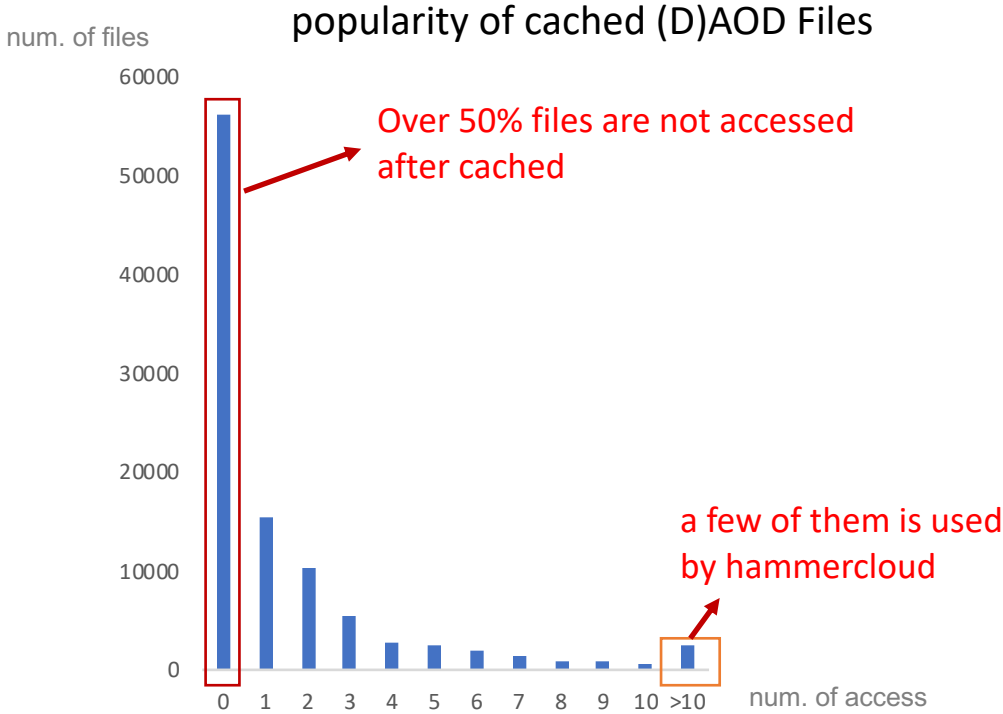


Performance

4 kinds of files are cached:

- **input:** input data files (AOD, DAOD, ...)
- **output:** user output
- **library:** user library files (dispatched by panda)
- **log:** job log files

	Write into cache	Read from cache	Cache hit	Cache hit rate
AOD	110957	343629	232671	67.7%
library	173	5052	4879	96.6%
log	7.7	7.8	0.07	~0
output	1275	1371.7	96.6	7%

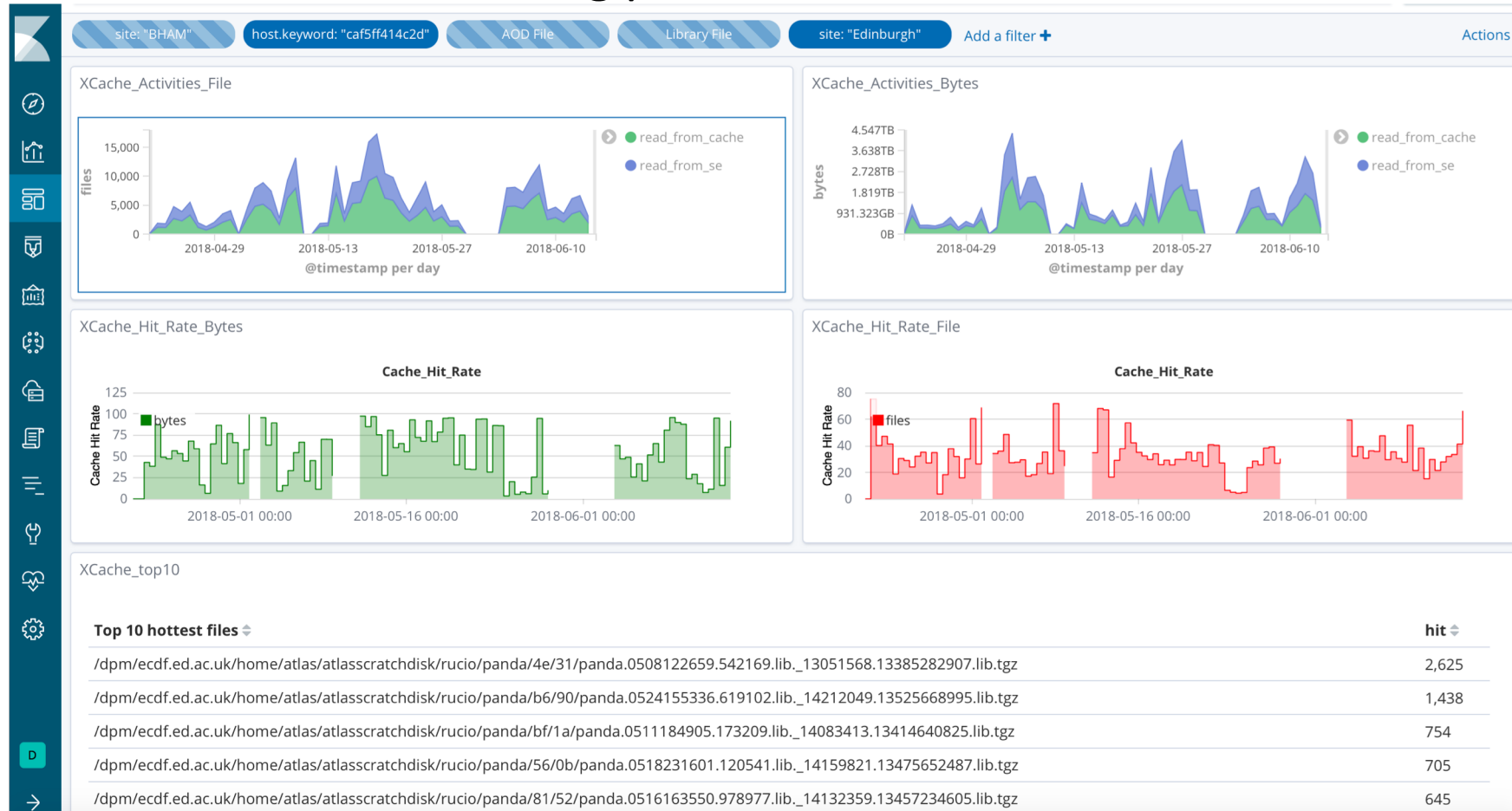


Performance

- Summary on performance study
 - Cache “hit” rate up to ~50% for ATLAS analysis jobs, 60% for ATLAS production jobs
 - Production queues are more “cache-able” and doesn’t need much optimization
 - For analysis queues:
 - Library files are extremely hot
 - Over half AOD input files are cold
 - Files are usually hot only for the first 4 hours
 - Some optimization methods may help
 - A decision library to filter out cold files
 - More aggressive purging frequency is preferred
 - Memory based cache for library files would work well

What's interesting to do next?

- See the actual performance for more sites
 - ELK stack built at Edinburgh for XCache monitoring and analytics
 - Monitor the real-time caching performance



What's interesting to do next?

- Partial file cache
 - Configure AGIS settings of the queue to directly access storage
 - Configure pre-fetch blocks number and block size in XCache to tune performance
 - Be careful of the access latency
- XCache and rucio
 - Fancy part to exploit functionalities and performance
 - Discussion, R&D and testing needed
- Optimizations
 - Performance study shows large optimization potential
- Contribute to the doma-access group

Thanks
&
Q & A