XCache studies and Logfile analytics

G. Duckeck, N. Hartmann, C. A. Mitterer, R. Walker, E. Schanet

LMU Munich

October 1, 2019

XCache

What is XCache?

- Disk caching proxy using xrootd (`libXrdFileCache.so`)
- Data is cached in blocks
- Simply prepend xcache server url - e.g.
  `TFile::Open("root:[xcache-server]:[port]//[xrootd-path]")`
- Optionally use rucio DIDs via N2N plugin:
  `https://github.com/wyang007/rucioN2N-for-Xcache`
  → allows usage of rucio DIDs instead of xrootd path
  → tracks identical files distributed at different locations
  (internal symlink `.../scope/XX/YY/filename`)

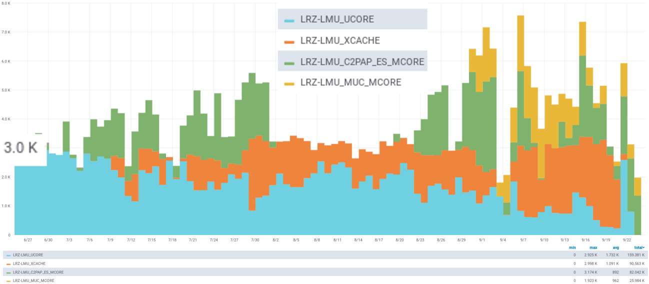
Setup

- Hardware: Old dCache pool node (from 2012):
  - Dell R710, 2x6 core Xeon L5640, 32 GB RAM, 10 Gb Ethernet
  - 60 TB Raid-6 (2x12x3TB HDD)
- Xrootd version 4.10.0
- Setup w/ singularity SL6 image. Full configuration: https://gitlab.physik.uni-muenchen.de/Nikolai.Hartmann/xcache-singularity-lrz/
- XCache settings:

```
pfc.ram 14g
pfc.blocksize 1M
pfc.prefetch 10
```

Test XCache in ATLAS production queue

Last 3 months:

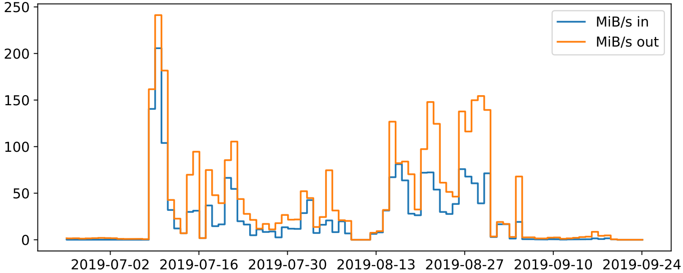


	min	max	avg	total▾
LRZ-LMU_UCORE	0	2.925 K	1.732 K	159.381 K
LRZ-LMU_XCACHE	0	2.998 K	1.091 K	90.563 K
LRZ-LMU_C2PAP_ES_MCORE	0	3.174 K	892	82.042 K
LRZ-LMU_MUC_MCORE	0	1.923 K	962	25.984 K

ATLAS production queue in Munich that retrieves all files via XCache

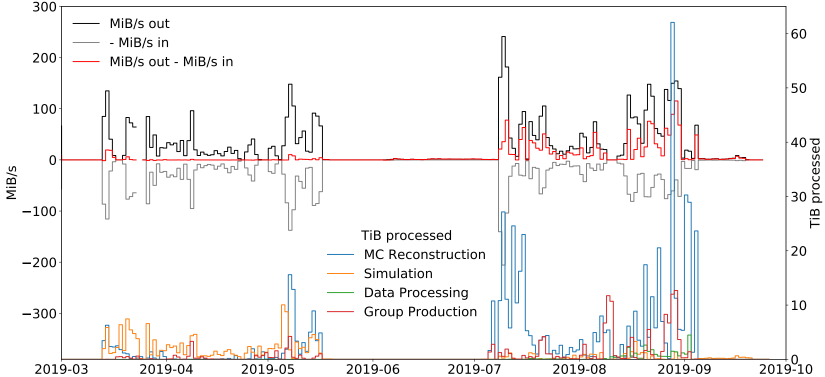
- Remote destination is nearby MPP Munich storage
- Can take a quite significant fraction of the jobs
- Works surprisingly well, given that all trafic goes throuh a single server

Caching works



→ Output volume already larger than input volume (≈ 1.8)

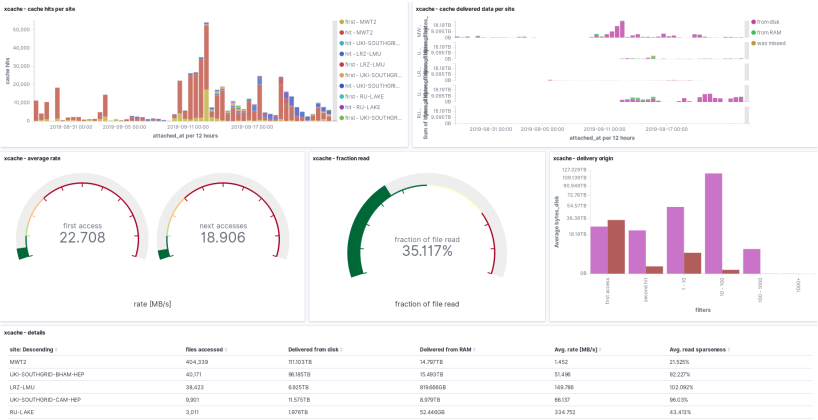
But hit rate depends on type of job



→ largest hit rate for MC Reconstruction (here mainly pileup overlay)

Central monitoring for ATLAS XCaches

Since a few weeks we are (together with other ATLAS XCaches) monitoring file access statistics to an ElasticSearch instance in Chicago



xcache - details

site: Descending	files accessed	Delivered from disk	Delivered from RAM	Avg. rate [MB/s]	Avg. read sparseness
MWT2	404,339	111.103TB	14.797TB	1.452	21.525%
UKI-SOUTHGRID-BHAM-HEP	40,171	96.185TB	15.493TB	51.496	92.227%
LRZ-LMU	38,423	6.925TB	819.666GB	149.786	102.092%
UKI-SOUTHGRID-CAM-HEP	9,901	11.575TB	8.979TB	66.137	96.03%
RU-LAKE	3,011	1.976TB	52.446GB	334.752	43.413%

Bugs/Issues

Found 2 Problems when XCache is under high load:

- Number of open files increasing until system limit is hit (https://github.com/xrootd/xrootd/issues/975) → fix in work
  → partially mitigated by settings: `pss.ciosync 60 900`
- Segfaults/Crashes (https://github.com/xrootd/xrootd/issues/1026)
  → mostly fixed in xrootd 4.10, but occasionally still seen for very high load (pileup jobs)

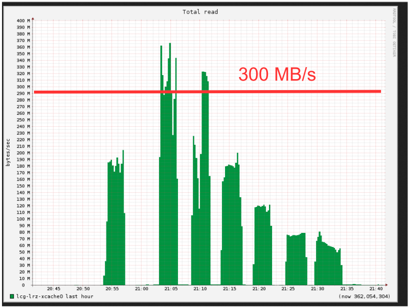
Lead to corrupted files: wrong checksum for file in cache, ≈ 90 out of 200k files
→ not observed any more after fixes/mitigations
→ still, we want to have a check for corrupted files in the future

Performance for parallel reads - Raid6 vs single disks

Feedback from xrootd developers: Use multidisk-mode instead of Raid

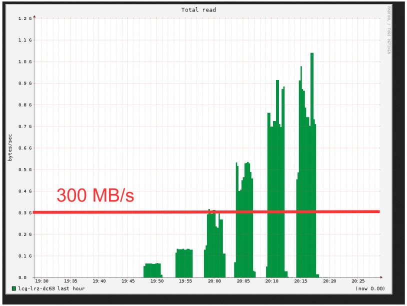
Performance test at LRZ:

Raid 6



1 2 5 10 20 40
simultaneous reads

individual disks



1 2 5 10 20 40
simultaneous reads

→ multi-disk mode might perform better than Raid for caching system
→ similar test with additional 50% writes gives the same picture

Outlook

- Test Multi-disk mode instead of Raid system
  → claimed to perform better, tests seem to confirm
- Investigate more use cases for caching:
  - Analysis jobs/Direct read instead of copy-to-scratch
    → continue tests, saw issues with long running jobs
  - Test XCache in columnar data analysis (e.g. with Pandas/Dask)