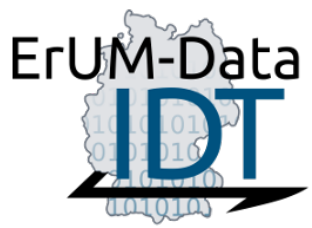
Update on XCache tests at LMU Munich

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

LMU Munich

26th November 2019, DOMA/ACCESS meeting



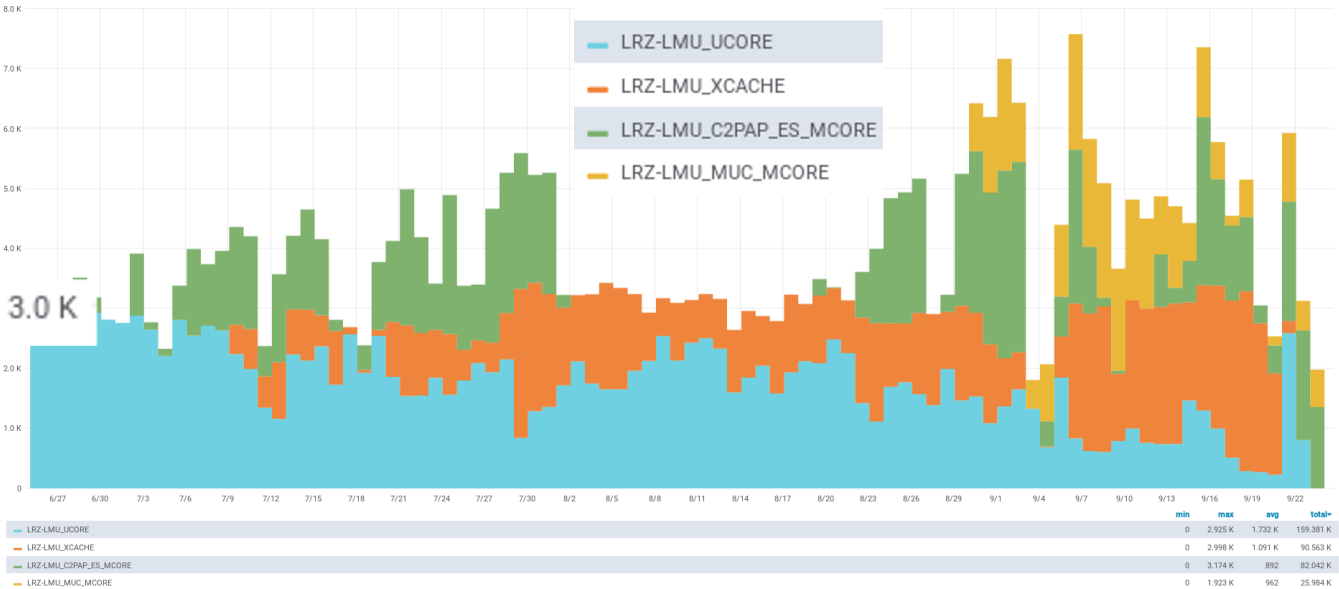
Setup

- 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)
    $\rightarrow$ second node with individual disks since November 2019
- 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

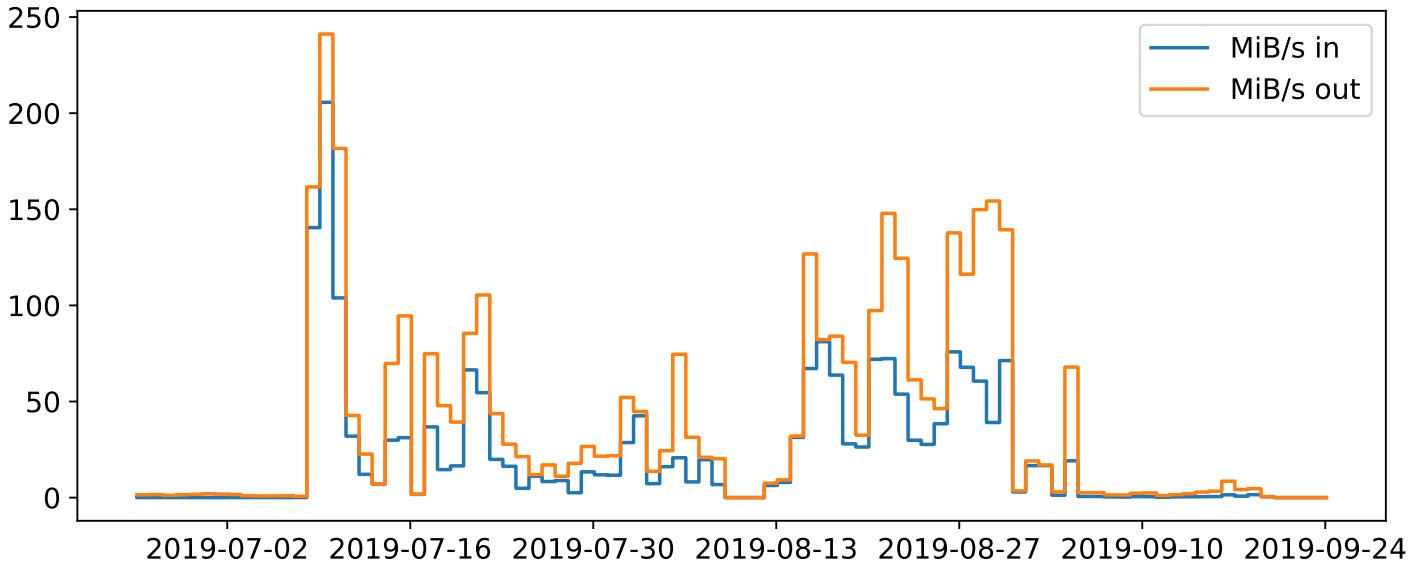
Last 3 months:



ATLAS production queue in Munich that retrieves all files via XCache

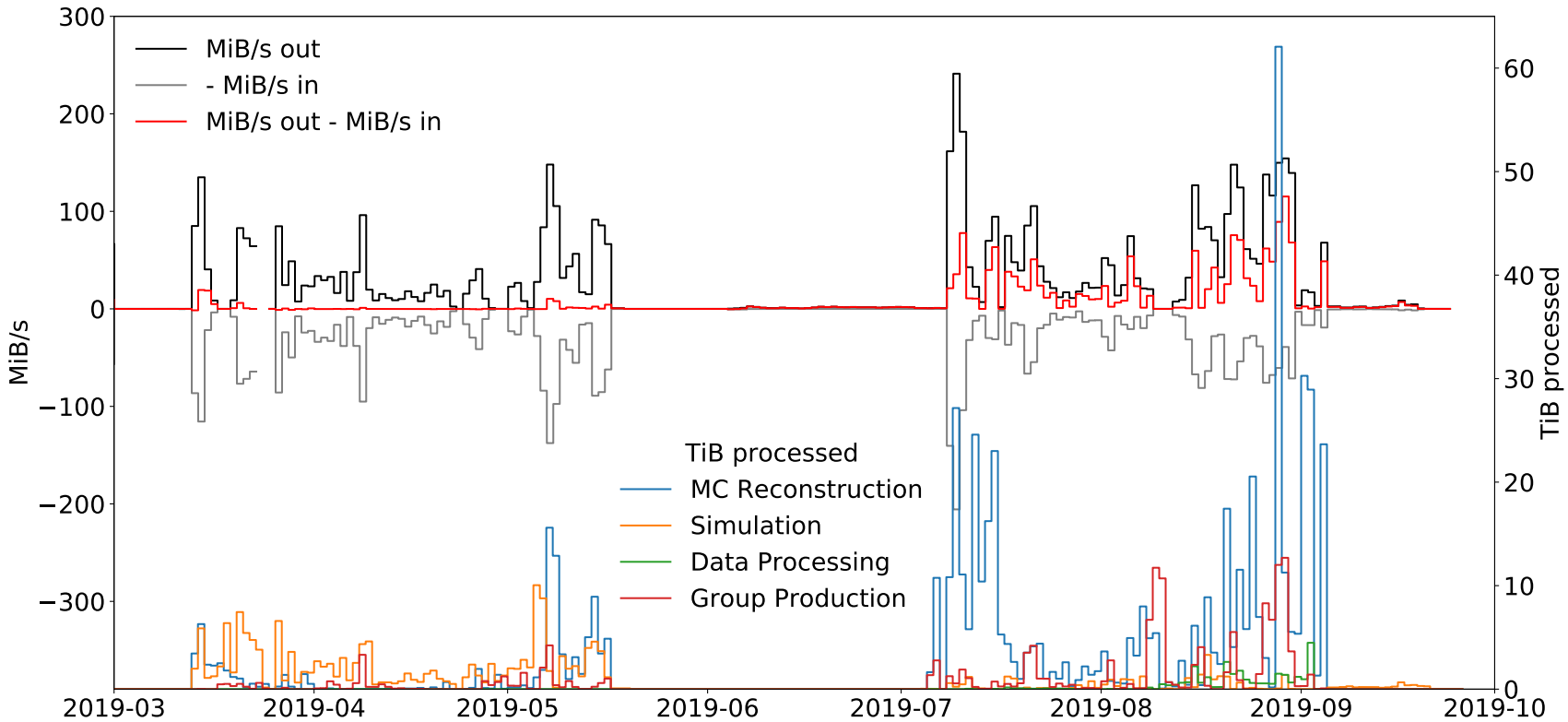
- Remote destination is nearby MPP Munich storage
- Can take a quite significant fraction of the jobs
- Works surprisingly well, given that all traffic goes through a single server

Caching works



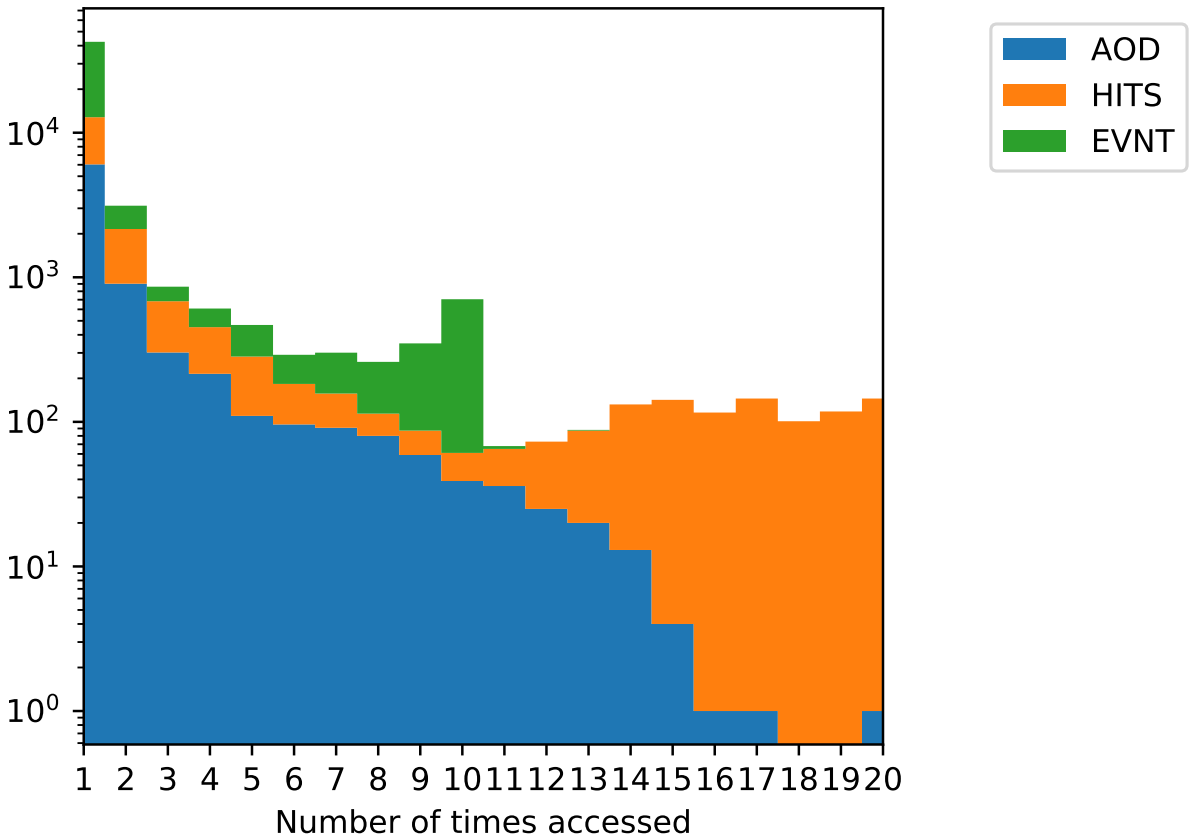
$\rightarrow$ Output volume already larger than input volume (≈ 1.8)

But hit rate depends on type of job



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

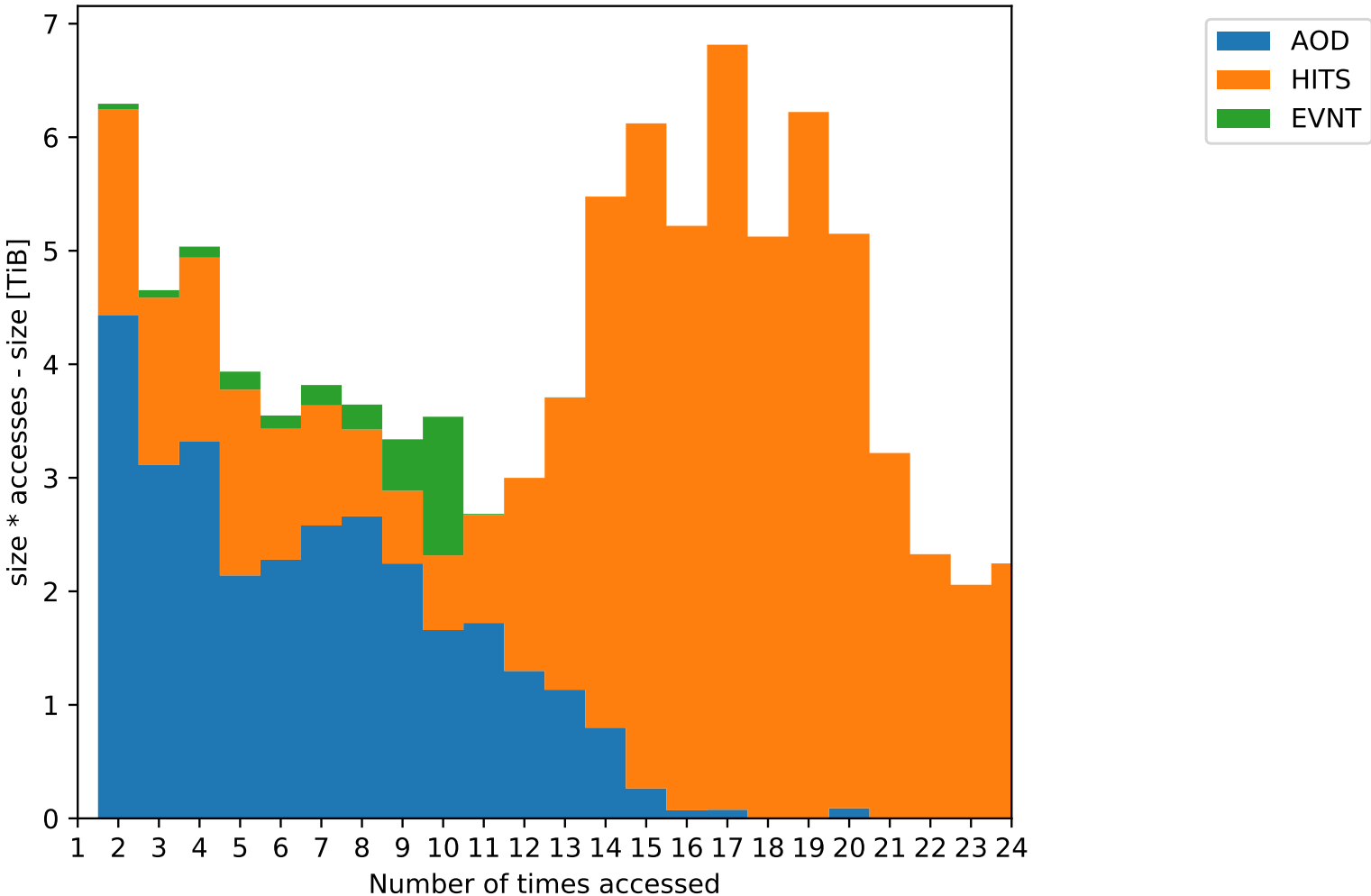
Access statistics from cinfo files



- Most reused files are HITS (pileup)
- EVNT files get reused when one file is processed via multiple jobs
- AOD files get reused for DAOD production (?)

Weighted by `size * accesses - size`

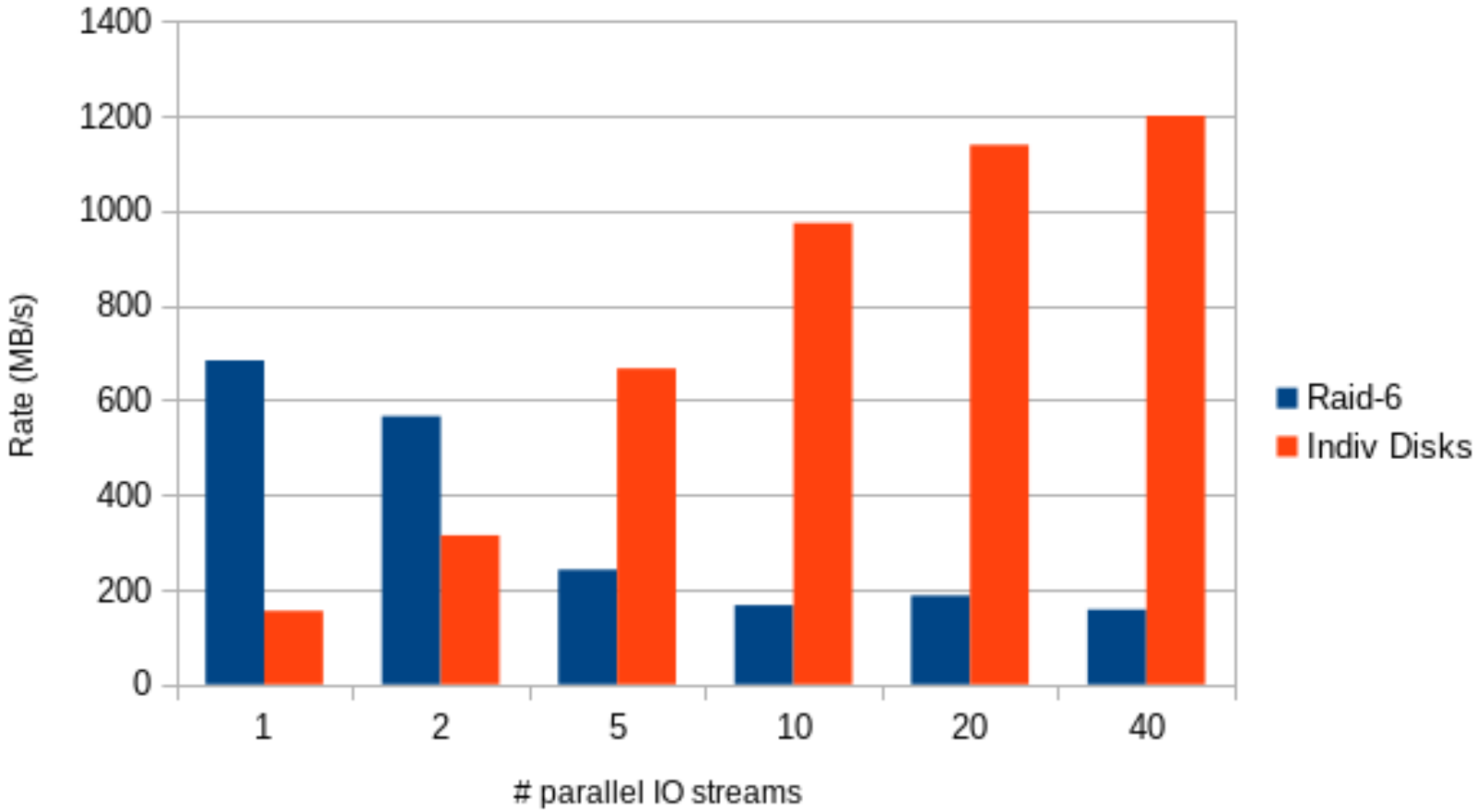
Corresponding reduction in WAN traffic
(w.r.t reading everything from remote without cache)



Performance for parallel reads - Raid6 vs single disks

Feedback from xrootd developers: Use multidisk-mode instead of Raid
(see slides from Matevž at XRootD workshop)

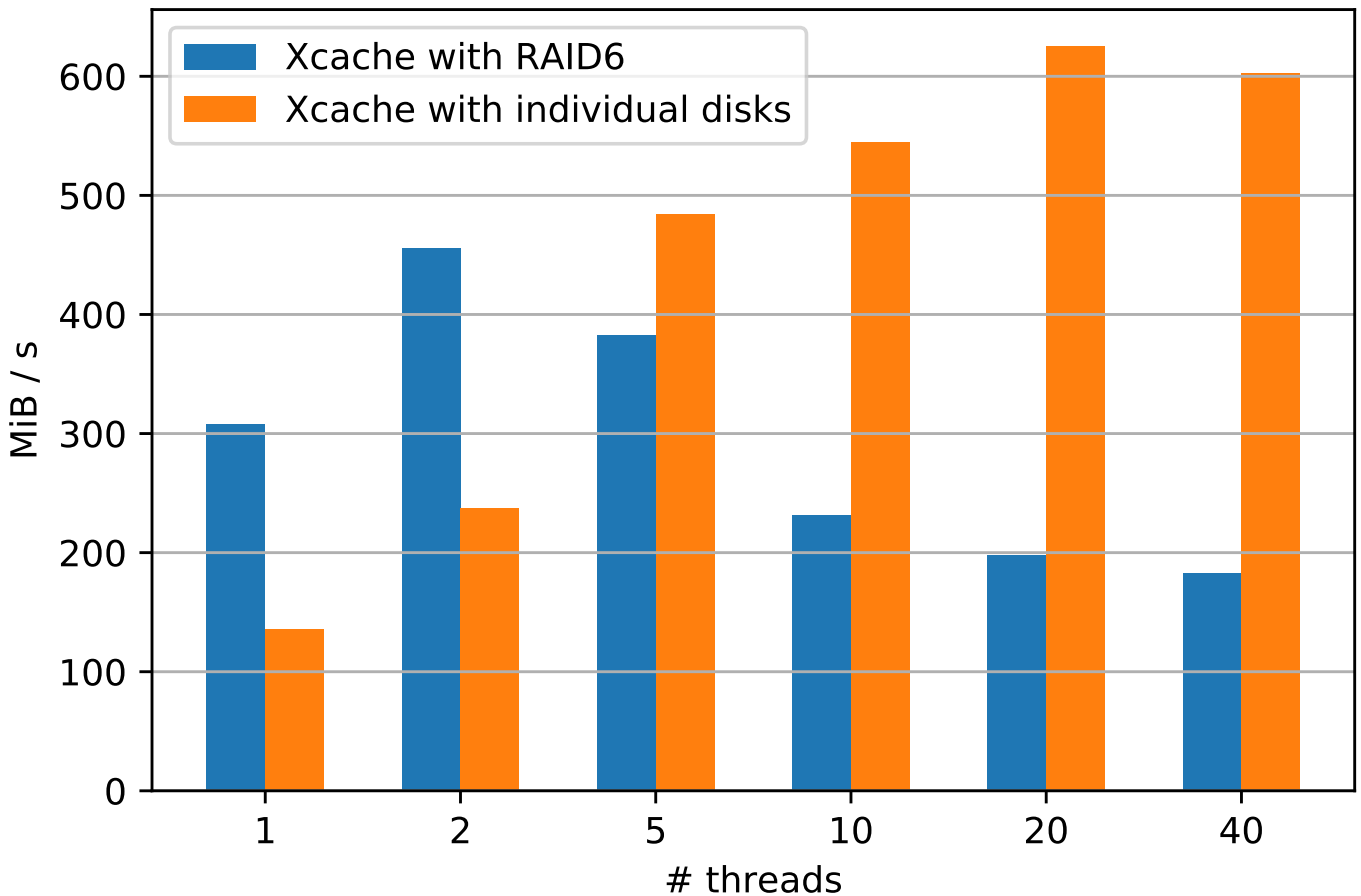
Raw reading tests at LRZ:



→ multi-disk mode might perform better than Raid for caching system

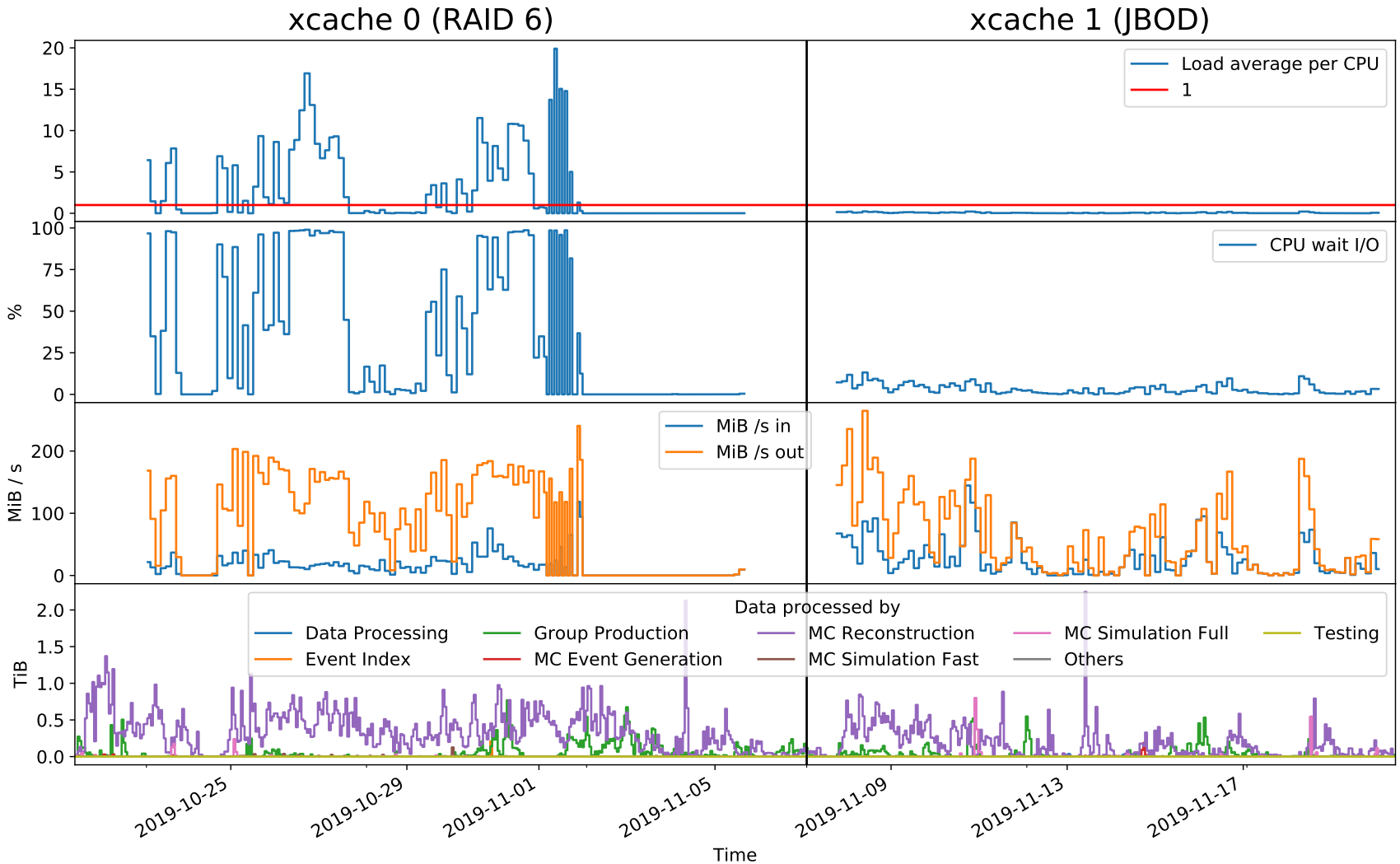
Performance for parallel reads - Raid6 vs single disks

Now similar test with an actual xcache setup:
(read random cached files through xcache, read from server)



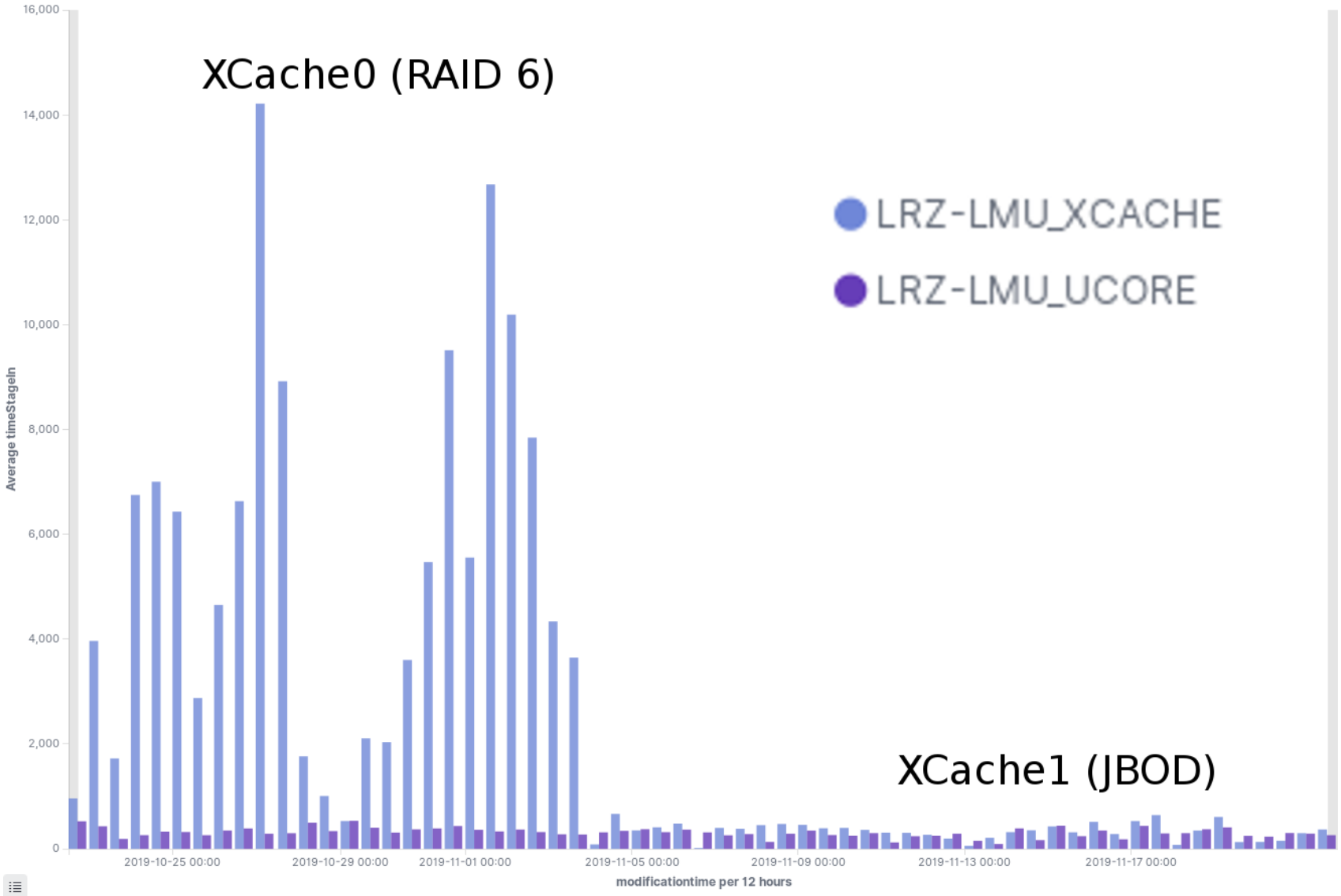
→ same conclusion - individual disks outperform RAID for parallel reads

Multidisk XCache in ATLAS production queue



→ load and wait CPU drastically reduced for multidisk mode setup!

Stage-in times



→ comparable stage-in times (with JBOD) as for non-xcache queue

Summary

- Successful running of xcache in ATLAS production environment
- Most reused files in current workflow from pileup overlay jobs
- Running XCache with individual disks beneficial (compared to RAID6)
  - significantly reduces load and wait times
  - peak I/O also increased for parallel disk reads/writes

Next plans

- Further stress testing:
  - Remove I/O limit on xcache queue
  - Run all jobs through xcache
- Combine the 2 xcache servers to a cluster
- Implement checksum test for fully cached files
  $\rightarrow$ long-term plan of developers: have blockwise checksums
- Continue tests with analysis jobs
- Test remote processing in practice
  (currently reading from neighbor site)

Backup

Bugs/Issues

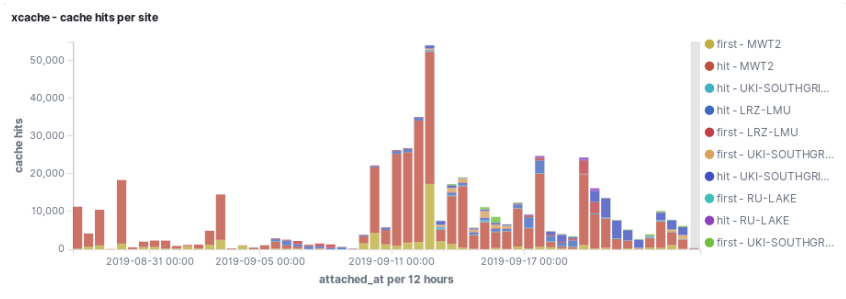
Found 2 Problems when XCache is under high load:

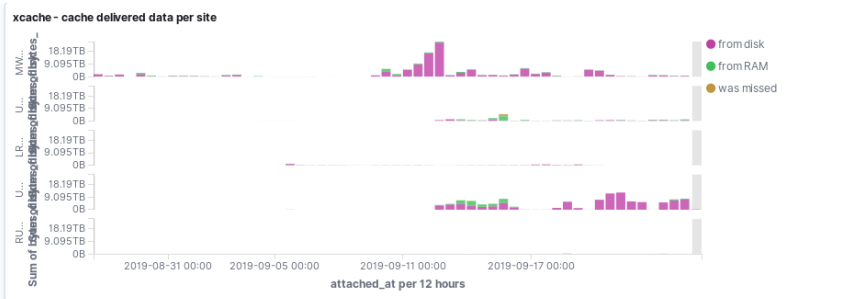
- Number of open files increasing until system limit is hit (https://github.com/xrootd/xrootd/issues/975) → fix in work
  → partially mitigated by settings: `pss.ciosync 60 900`
- Segfaults/Crashes (https://github.com/xrootd/xrootd/issues/1026)
  → mostly fixed in xrootd 4.10, but occasionally still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

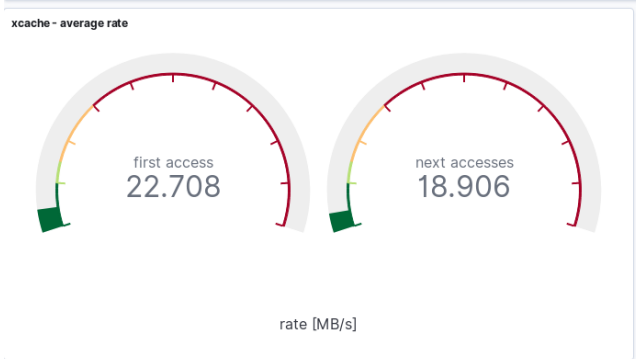
Lead to corrupted files: wrong checksum for file in cache, $\approx$ 90 out of 200k files
→ not observed any more after fixes/mitigations
→ still, we want to have a check for corrupted files in the future

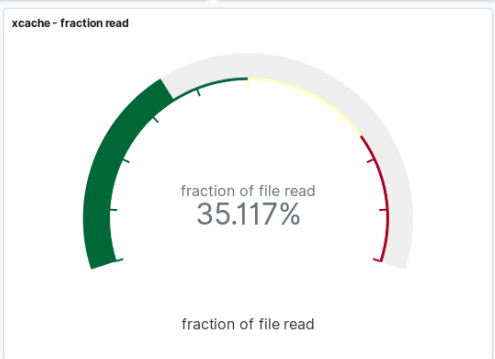
Central monitoring for ATLAS XCaches

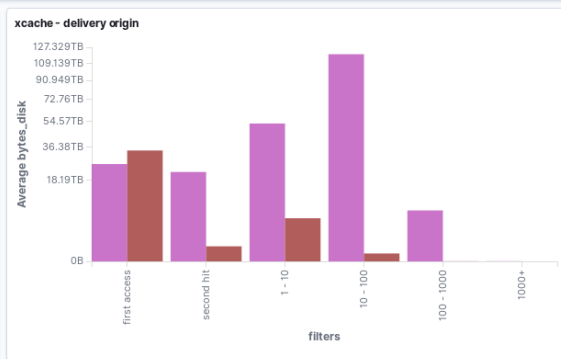
Since a few weeks we are (together with other ATLAS XCaches) monitoring file access statistics to an ElasticSearch instance in Chicago









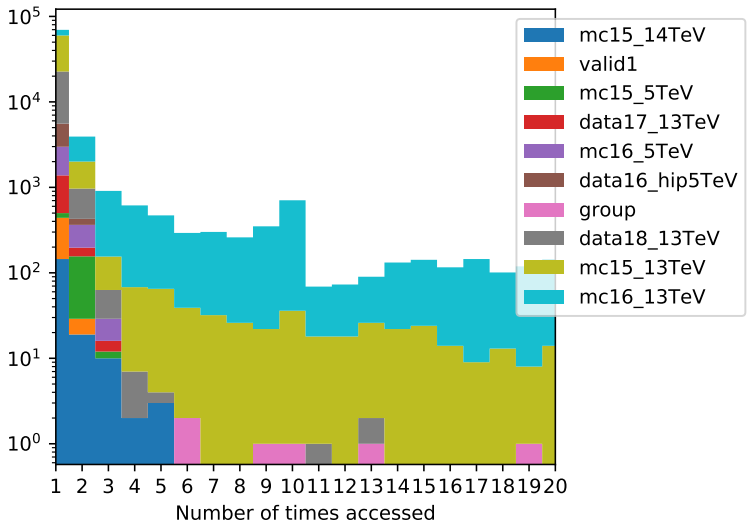


xcache - details

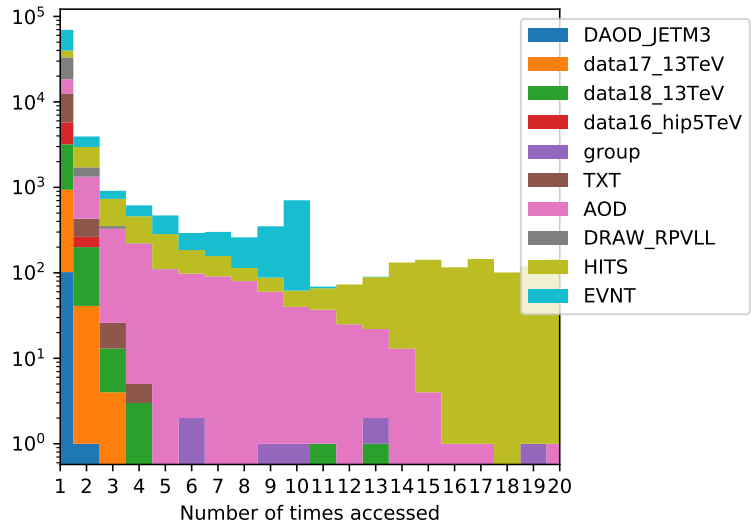
site: Descending	files accessed	Delivered from disk	Delivered from RAM	Avg. rate [MB/s]	Avg. read sparseness
MWT2	404,339	111.103TB	14.797TB	1.452	21.525%
UKI-SOUTHGRID-BHAM-HEP	40,171	96.185TB	15.493TB	51.496	92.227%
LRZ-LMU	38,423	6.925TB	819.666GB	149.786	102.092%
UKI-SOUTHGRID-CAM-HEP	9,901	11.575TB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

Access statistics from cinfo files - detailed

By scope



By data type



Which HITS?

Add info from rucio (parent DID name)