XCache plans and studies at LMU Munich

Nikolai Hartmann, G. Duckeck, C. Mitterer, R. Walker

LMU Munich

February 22, 2019

LMU ATLAS group

- Active in ATLAS Computing since the start
  - Operate ATLAS-T2 and HPC resources at LRZ
- Participating in projects A2, A3, B1
  - People: G. Duckeck, N. Hartmann, C. Mitterer, R. Walker
    - N. Hartmann funded by ErUM-Data, 50% A/B, 50% C
- Focus currently on evaluation of caching use-cases in ATLAS
- Substantial past experience on Analysis of ATLAS job- and data-transfer logs:
  - job-parameter (CPU/WallTime, IO Rate IO Volume, ...)
  - dynamic data placement (CHEP-2018 talk/proceedings)
  - caching efficiency studies

Use cases for caching

Caches can supplement manual/automatic data placement. Interesting scenarios:

- Processing on local batch systems without the need to replicate data
- Sites with small storage but available computing resources
- "Hospital" queues where jobs can be processed for which all slots on sites where data is available are full
- Caching for additional data - metadata, containers (http?)

$\rightarrow$ Testing in Munich with an XCache server

XCache

- Data access caching service using xrootd
- Simply prepend xcache server url - e.g.

```
TFile::Open("root:[xcache-server]:[port]//[xrootd-path]")
```

- Works also with rucio data identifiers
  (server looks up replicas via rucio)
- Data is cached in blocks

XCache hardware/software at LRZ-LMU

- 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 4.8.5
- Setup w/ singularity SL6 image following Wei's instructions: https://github.com/wyang007/rucioN2N-for-Xcache/wiki/Deploy-Xcache-via-a-Singularity-Container
- XCache settings:

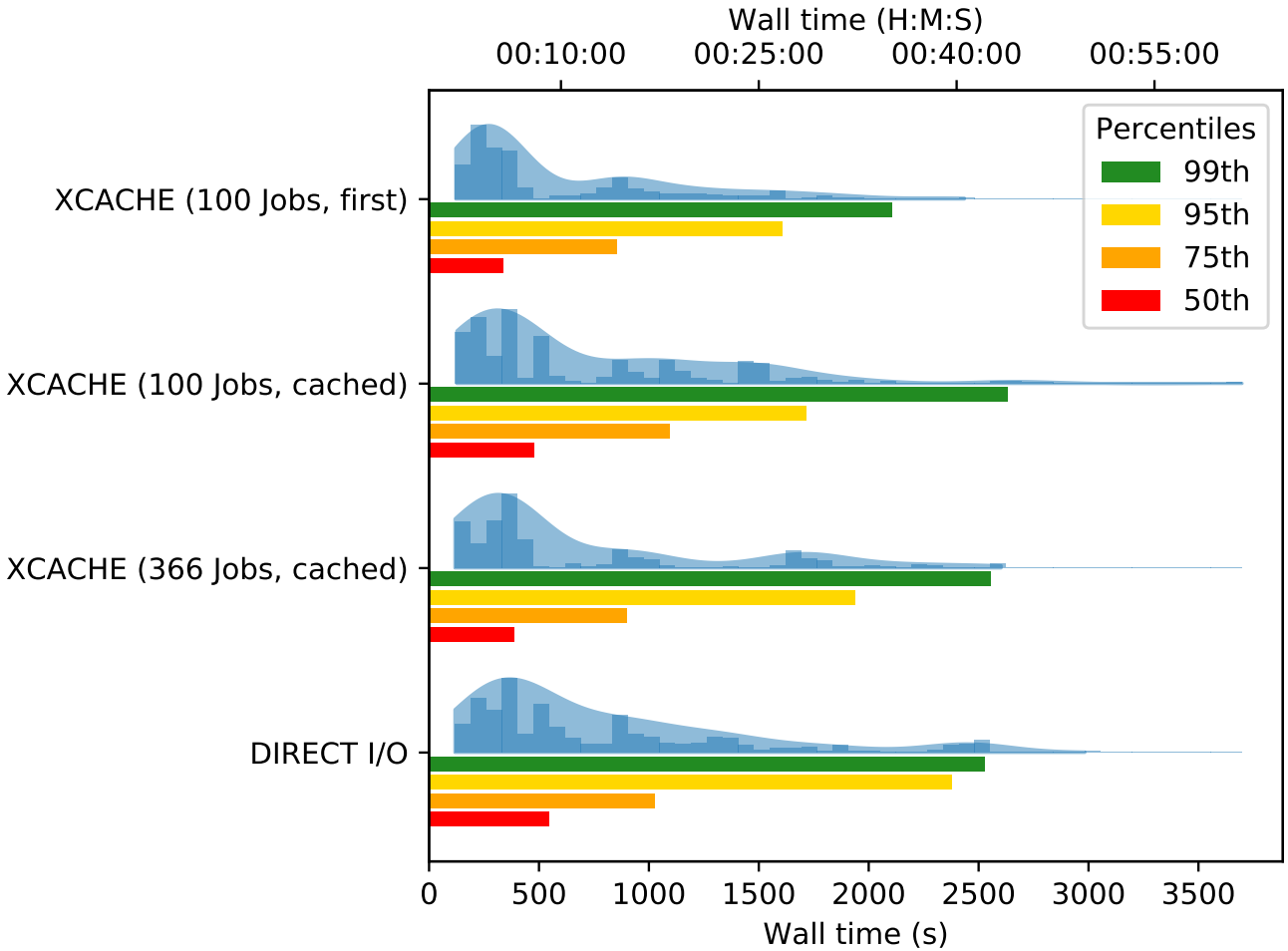
```
pfc.ram 14g
pfc.blocksize 1M
pfc.prefetch 10
```

Testing the XCache server (local batch)

- Test cases: user analysis job and ATLAS derivation
- Analysis job I/O: $\approx 0.5 - 1.5$ MB/s
- Derivation job I/O: ≈ 3 MB/s
- Analysis job test dataset: ≈ 340GB in 366 Files
  (reading via DESY Hamburg)
- Derivation job test dataset: ≈ 33GB in 10 Files

Job run times

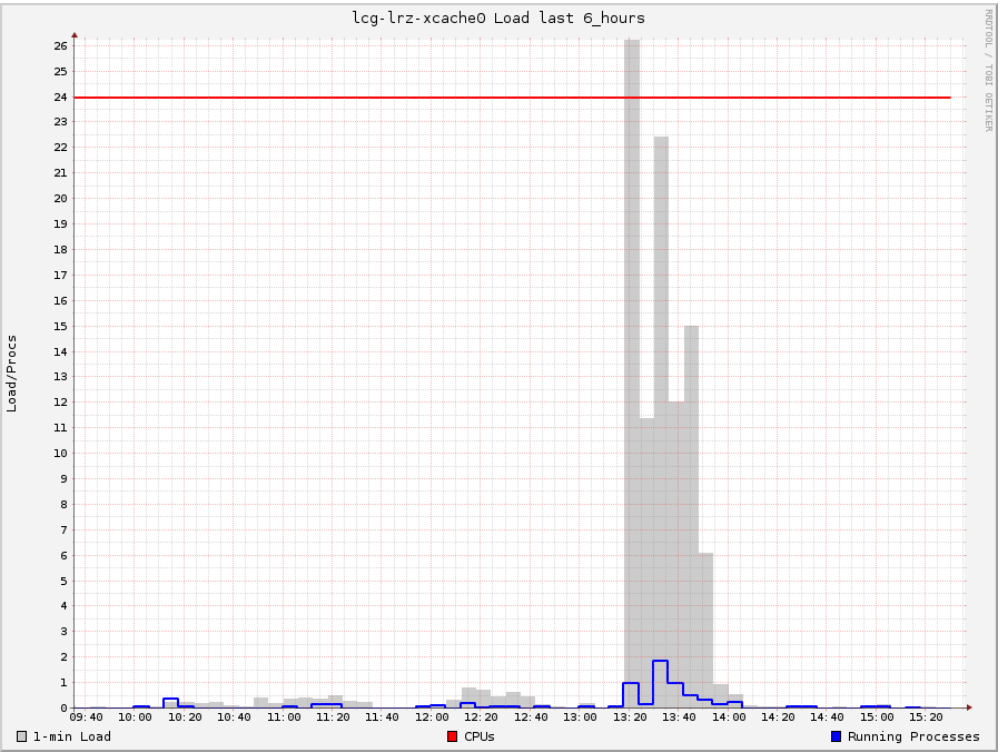
Analysis Job, data stored on well-connected DESY Hamburg site

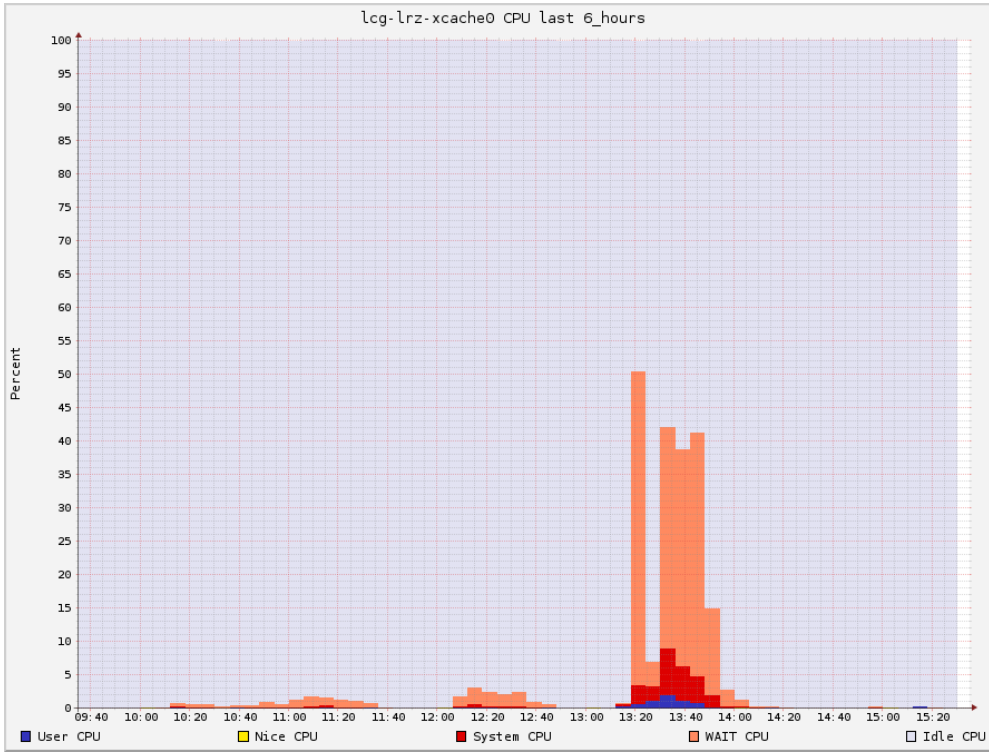


No large difference between first and second processing via cache, direct I/O has similar performance

XCache server monitoring

366 Analysis Jobs at once - “stress test”

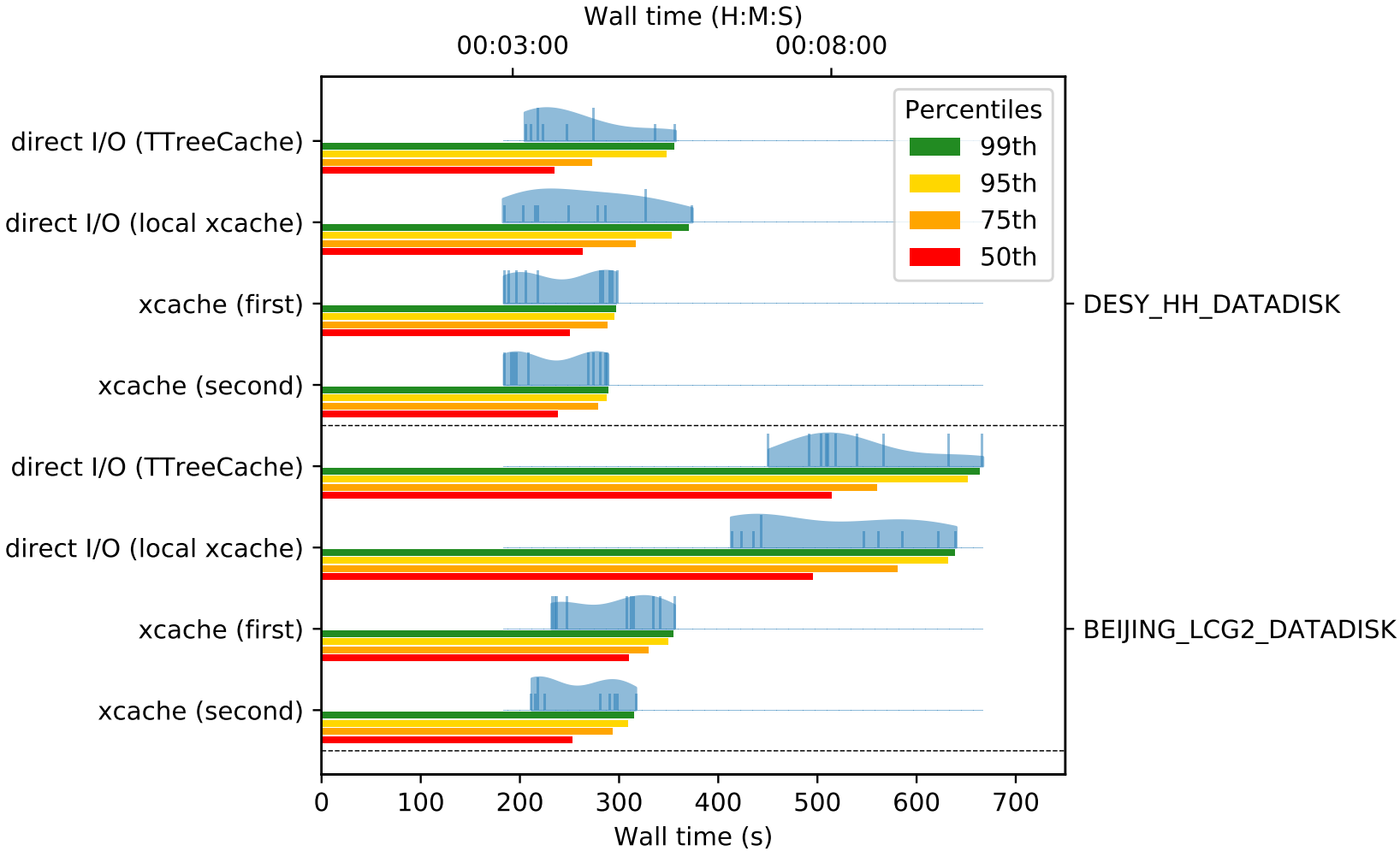




Server got busy for a short period of time, but analysis job runtimes basically unaffected

Processing from different sites

Derivation Jobs (≈ 3MB/s) - process 500 Events

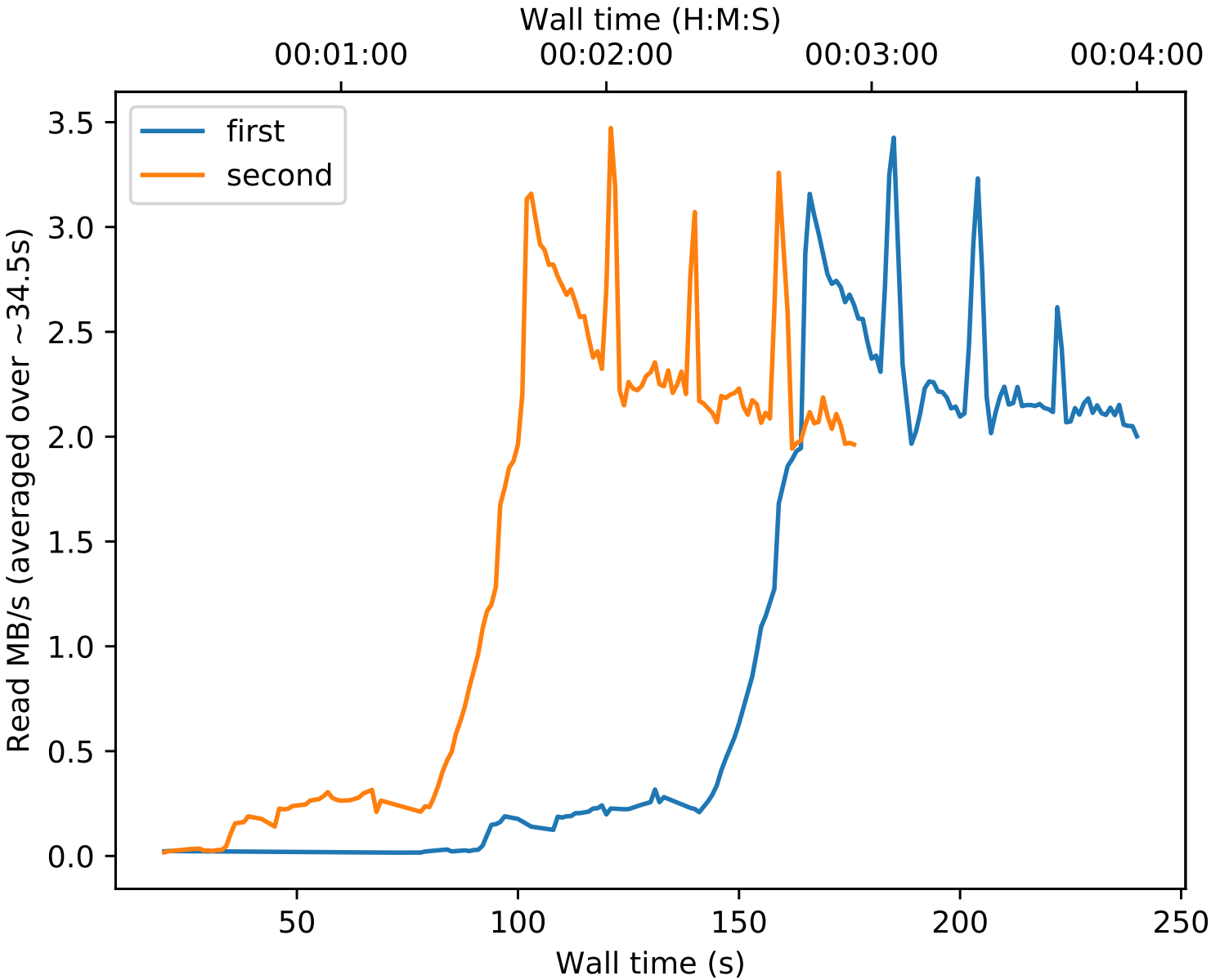


→ Differences for direct I/O and cached visible for far away sites

→ Local XCache (on each node) can serve as alternative to TTreeCache

First and second processing via Cache

Difference visible only for far away sites - Example: BEIJING



Integration into ATLAS grid production and analysis

- Set up test queues for analysis and production
- Including a “hospital” queue
- So far reading from a limited set of nearby storage elements (MPP Munich, DESY Hamburg)
- First tests successful - jobs automatically read files via XCache

Test during scheduled downtime of SE

- Setup production queue that read via XCache from neighbour site MPPMU
- Regular queue went into downtime
  → all ∼3200 cores/slots used by XCache queue
- By chance mostly pileup jobs requiring large input volume
  → ∼25 GB per 8 core job, running for ∼6 hours, avg read rate 300 MB/s via XCache
- XCache server heavily overloaded, but didn't break
- Subsequent transition to simulation jobs
  → no problem to serve all 3k slots with single XCache node

Job statistics compared to nominal running

XCache for Hospital queue

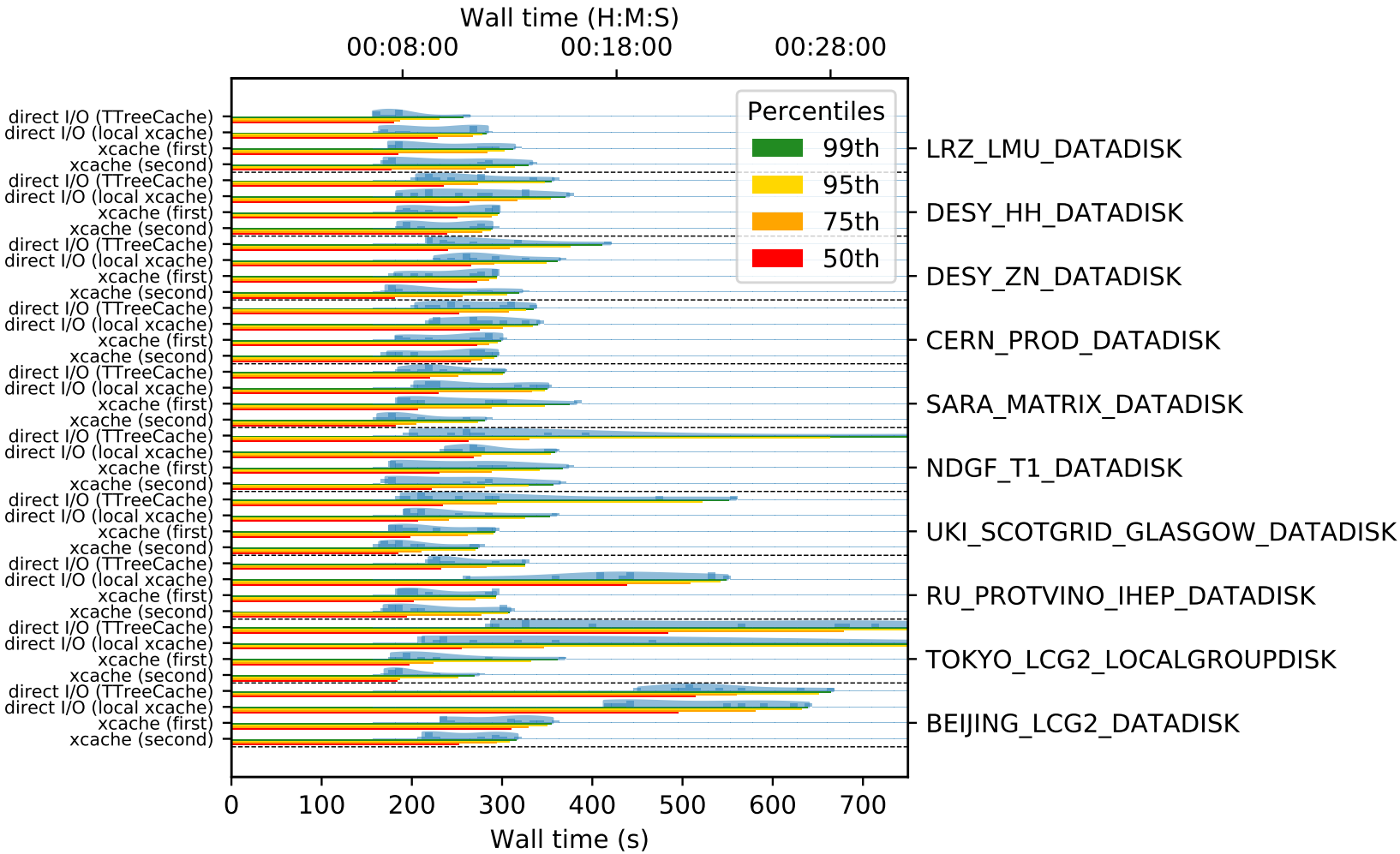
- Sometimes problems to schedule user-analysis jobs at site where data is located
  - Often T1 sites: big storage but low analysis share
- Concept of "Hospital" queues at other sites:
  - Analysis jobs which time out waiting for a free slot can get re-scheduled to remote site which is configured access data remotely from original site
- Good use-case for XCache
  - Setup at LRZ since $\sim$1 week, configured to pick-up jobs for Taiwan T1 site
  - Rather varying load
  - XCache issues with open-file limit

XCache experience: problems and issues

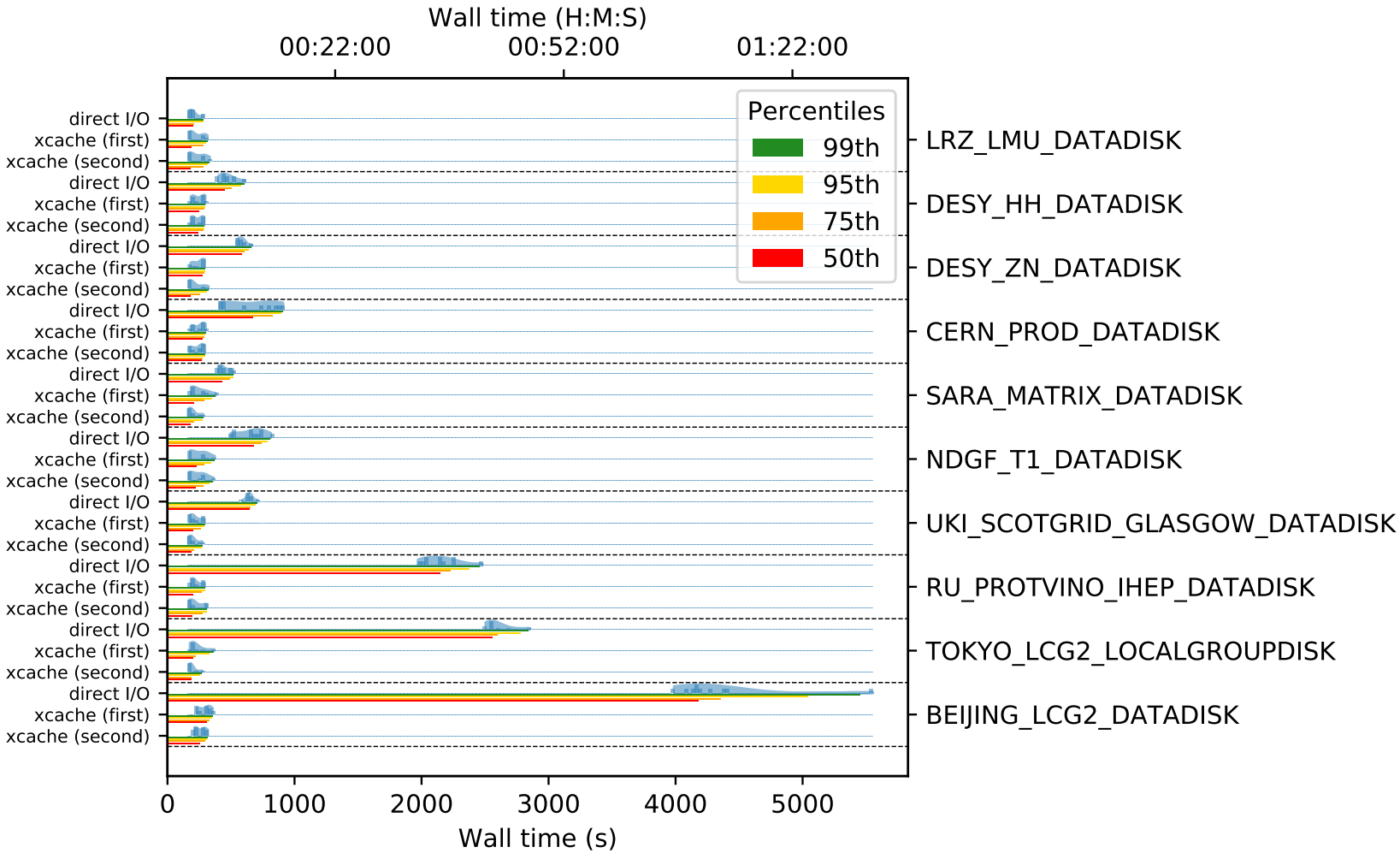
- Setup & deployment rather straightforward, except for some tricky details ...
- Checking cache status and cleanup for tests rather cumbersome
- Found weird bug: xcache crashes when reading same file simulateneously by different clients from different locations (fix in v4.9)
- Read failures due to open-file system limit (16k on our system)
  (Investigating, not clear if xcache or client problem)
- Overall positive experience: robust and performant

Backup

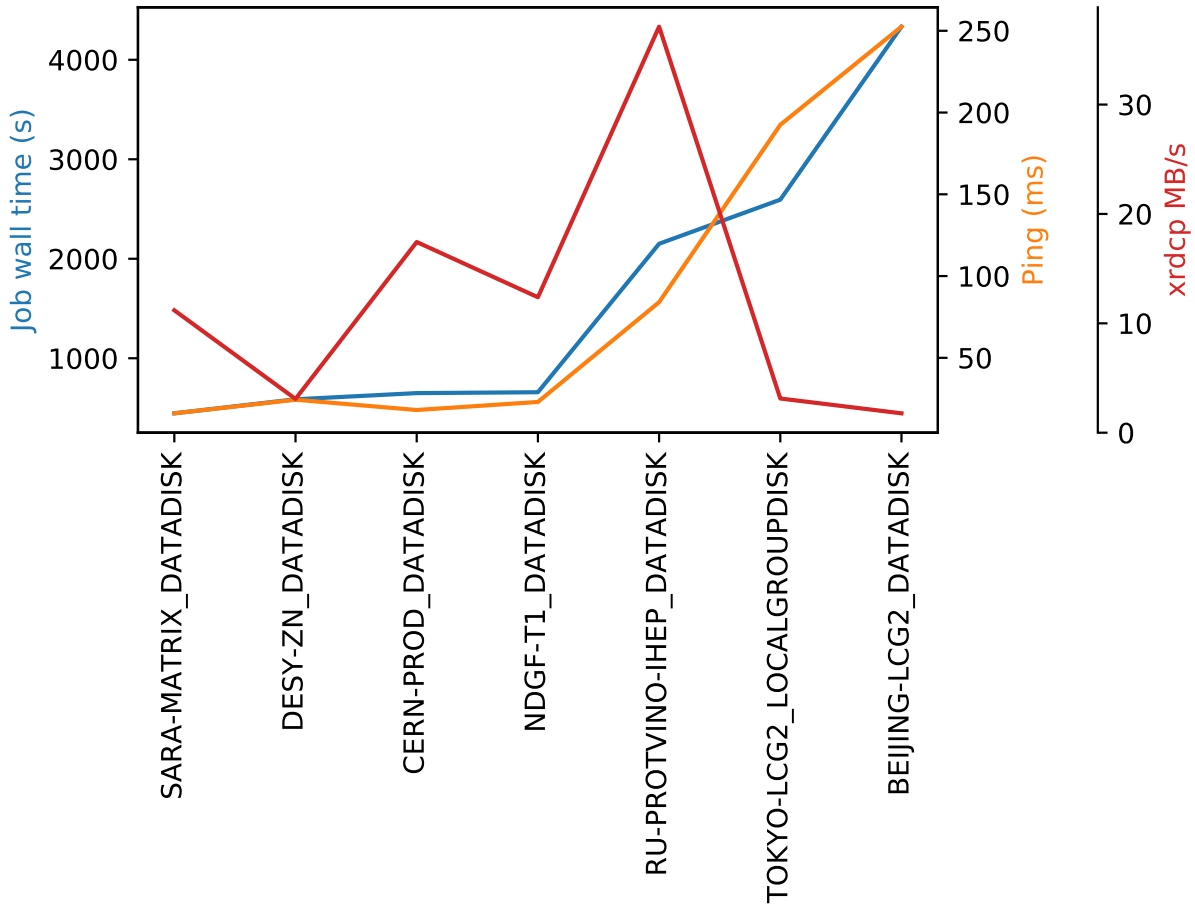
All sites


Wall time (H:M:S)
00:08:00
00:18:00
00:28:00
Percentiles
99th
95th
75th
50th
direct I/O (TTreeCache)
direct I/O (local xcache)
xcache (first)
xcache (second)
LRZ_LMU_DATADISK
DESY_HH_DATADISK
DESY_ZN_DATADISK
CERN_PROD_DATADISK
SARA_MATRIX_DATADISK
NDGF_T1_DATADISK
UKI_SCOTGRID_GLASGOW_DATADISK
RU_PROTVINO_IHEP_DATADISK
TOKYO_LCG2_LOCALGROUPDISK
BEIJING_LCG2_DATADISK
0
100
200
300
400
500
600
700
Wall time (s)

Without TTreeCache



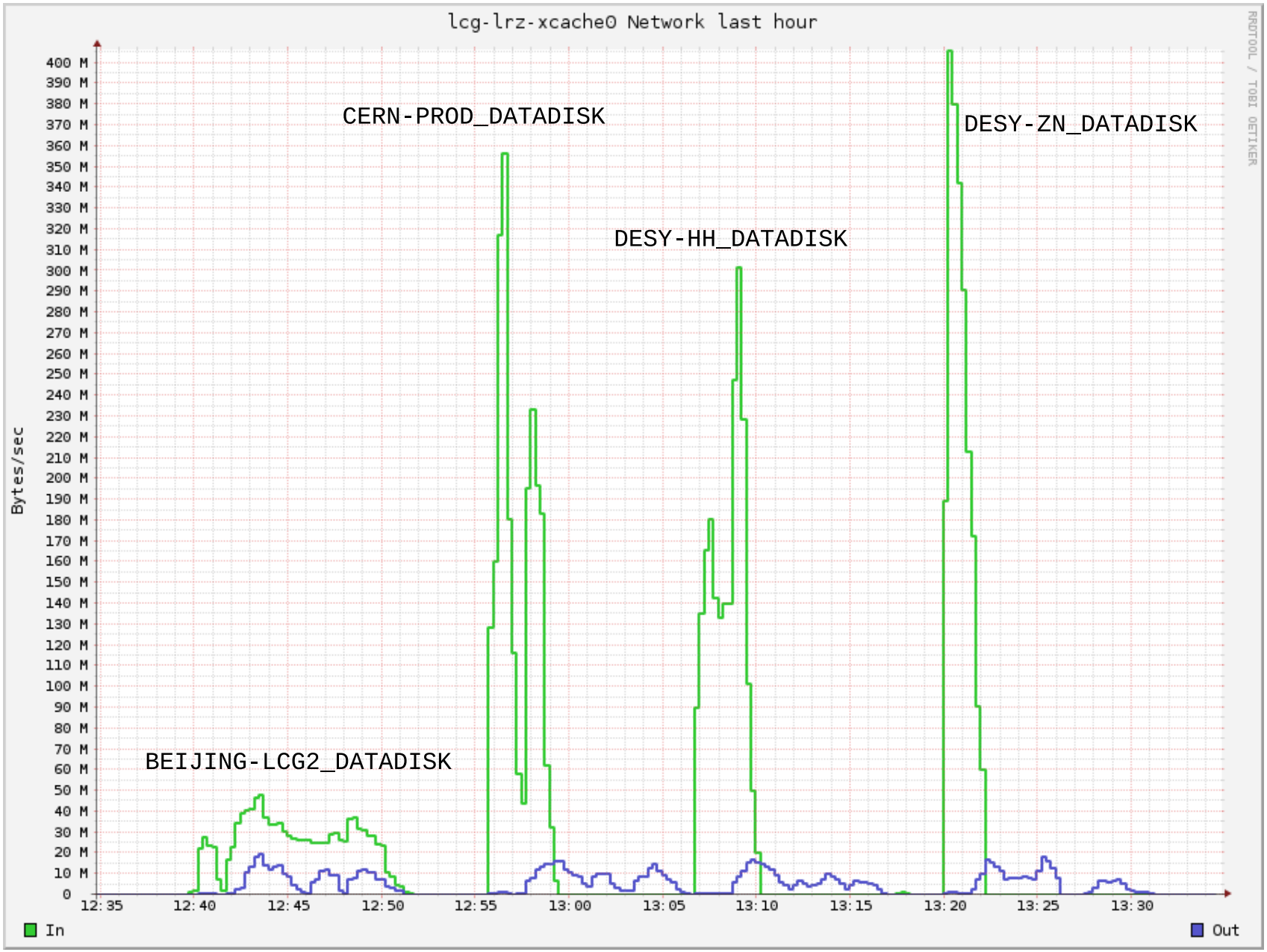
Run times without TTreeCache vs ping vs speed



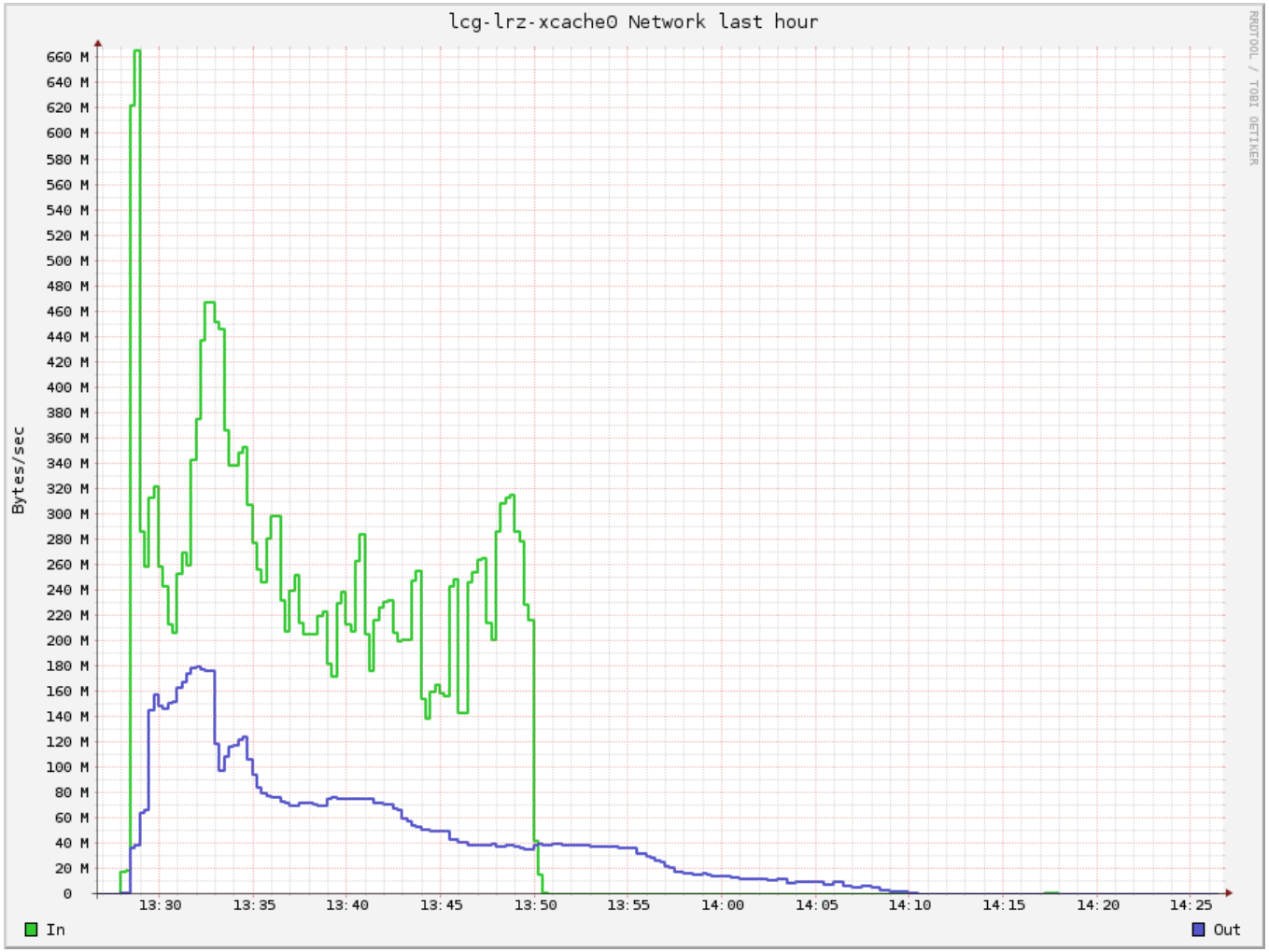
→ processing time correlates with ping, not (xrdcp) transfer speed

→ reading in larger blocks better (use TTreeCache)

XCache server I/O monitoring

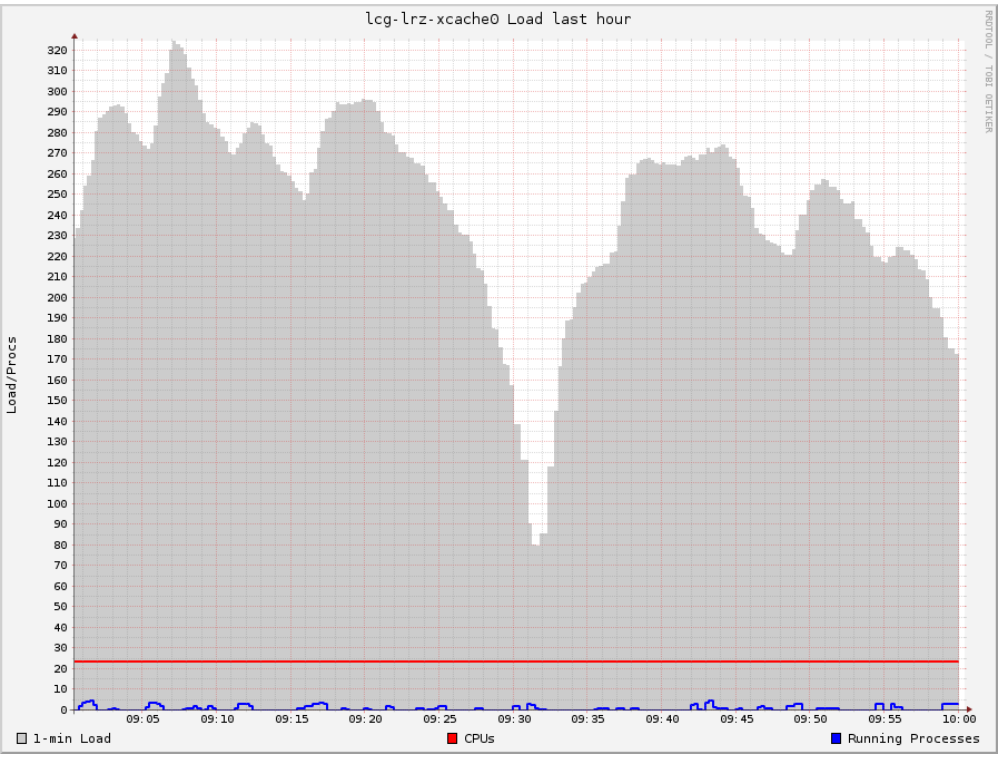


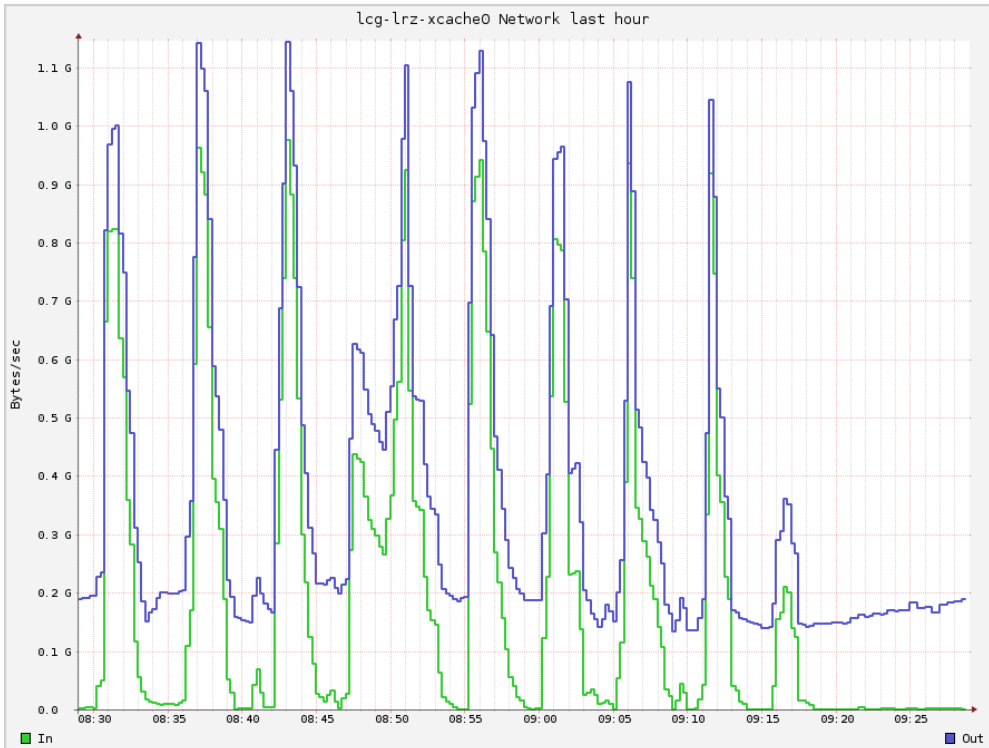
Analysis job “stress test” via lrz batch system



366 Jobs (one per file) at once still work fine
(total dataset size: ≈ 340 GB)

Load and IO on XCACHE server during pileup run





CPU on XCACHE server during pileup run