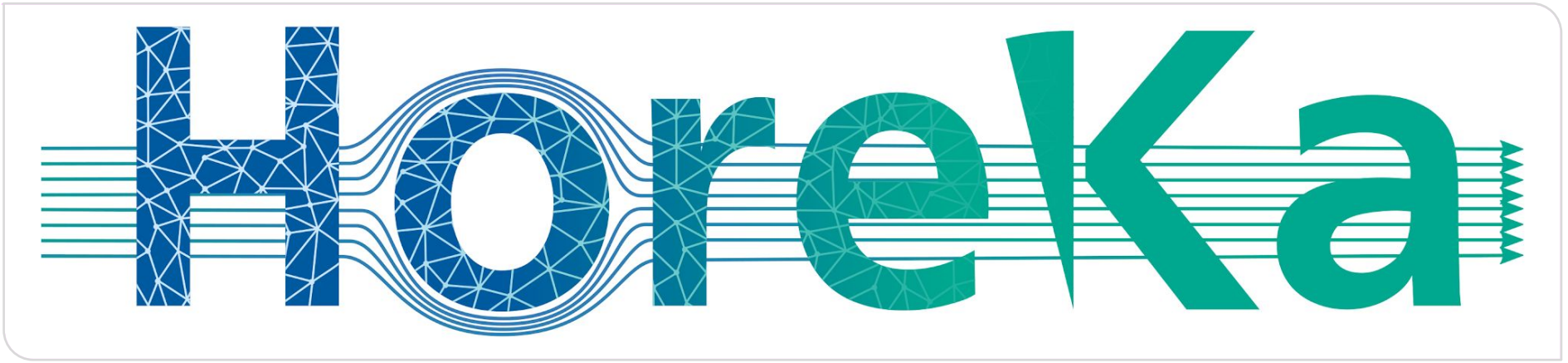
Karlsruhe Institute of Technology

Institute of Experimental Particle Physics

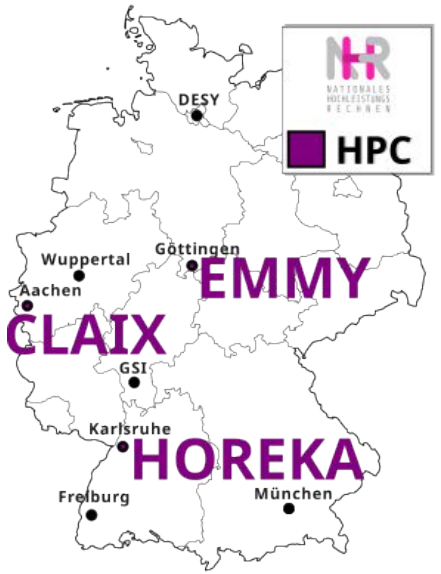
Experience with XCache on HPC in Germany

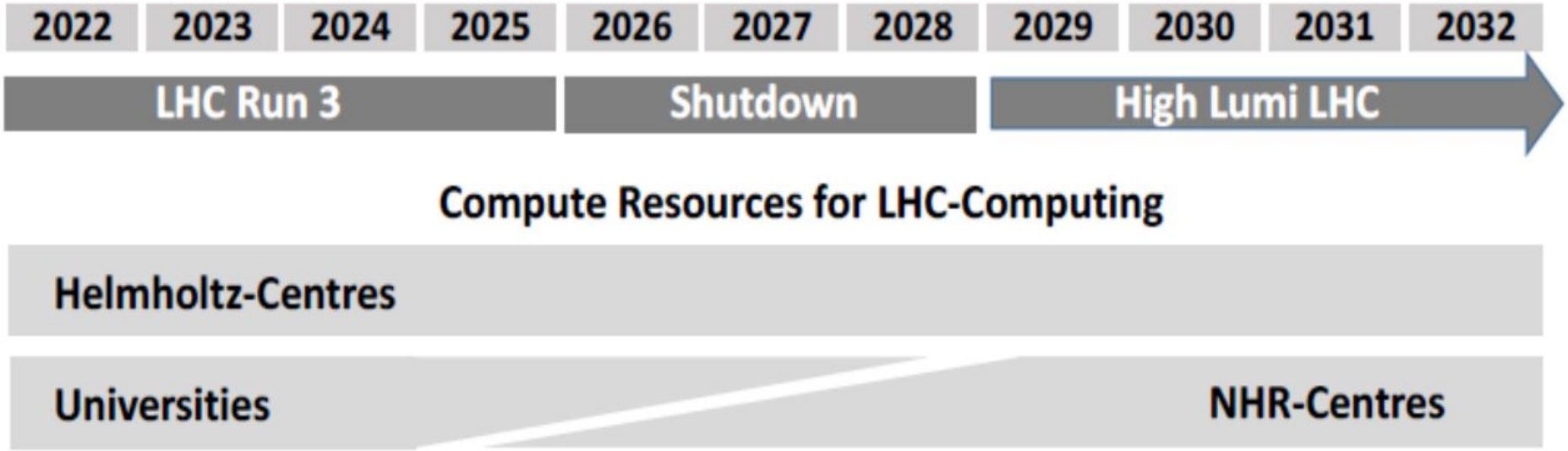
Robin Hofsaess for the GridKa R&D team



NATIONALES HOCHLEISTUNGS RECHNEN

German HEP Computing Strategy





- Transition from dedicated T2 resources at universities to shares on national HPC centers within the NHR computing compound
- Storage at the Helmholtz centers KIT and DESY
- Support from the current T2 groups

strategy paper (german only…)

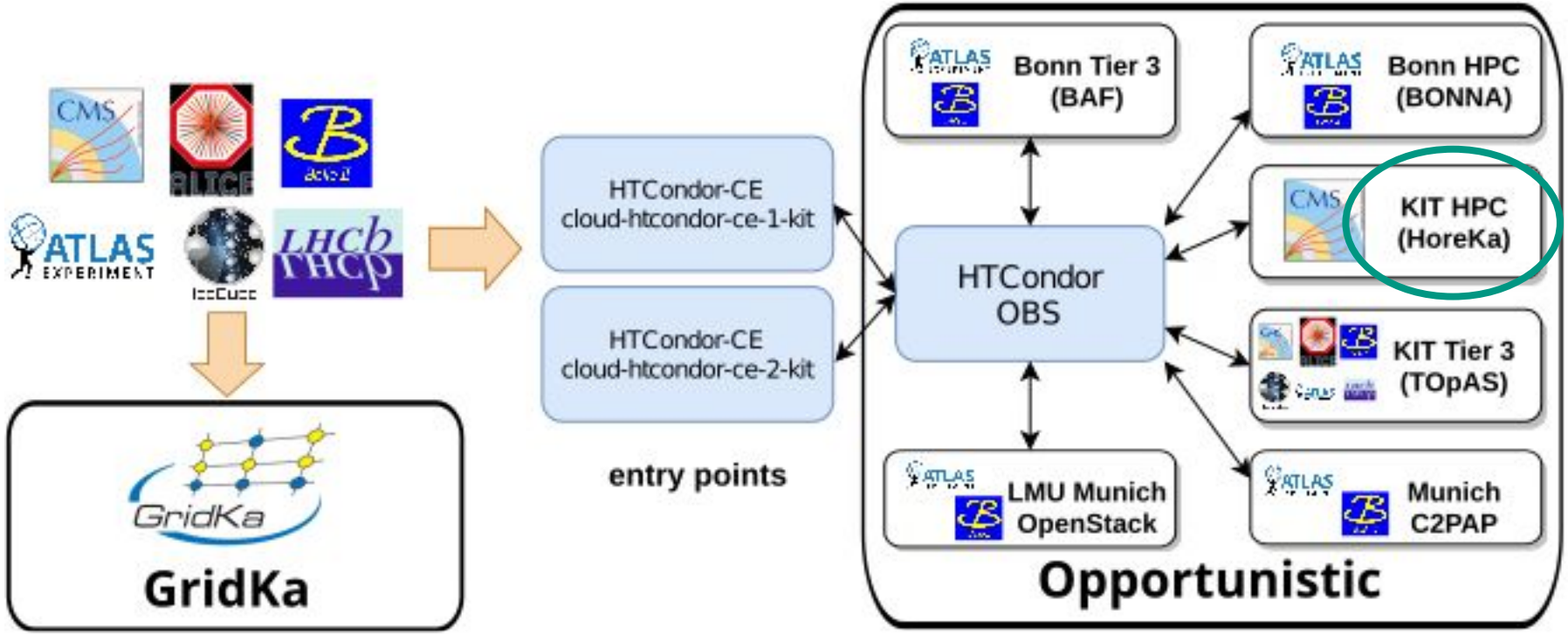
Why and what will change?

- Increased efficiency and sustainability:
  - Use resources that are available anyway
  - Newer, more efficient hardware + better sustainability/scalability
  - Higher utilization of national HPC centers
- Enhanced collaboration between communities

Changes and Challenges:

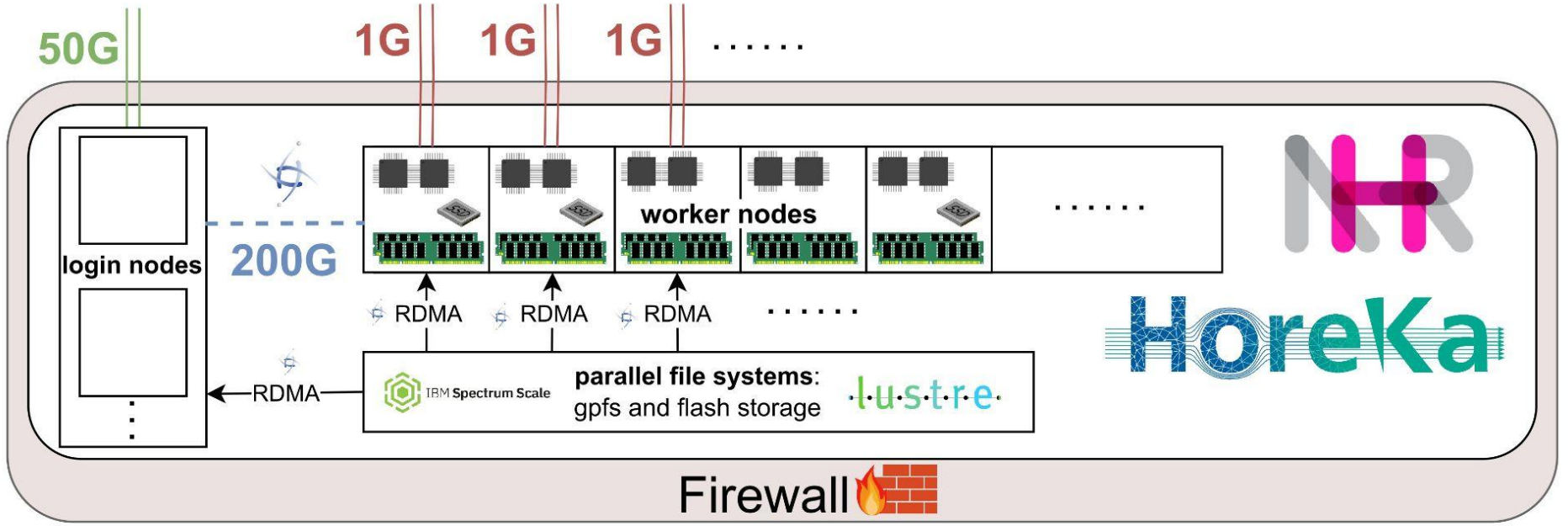
- Shift from mainly maintenance effort to more R&D and support in the T2 groups
- From admin to a user among many
- HPC is not inherently suited for HEP workflows compared to our dedicated WLCG (HTC) sites
- WLCG pledges: A reliable and efficient operation must be guaranteed!
- Political implications

Situation at KIT + HoreKa



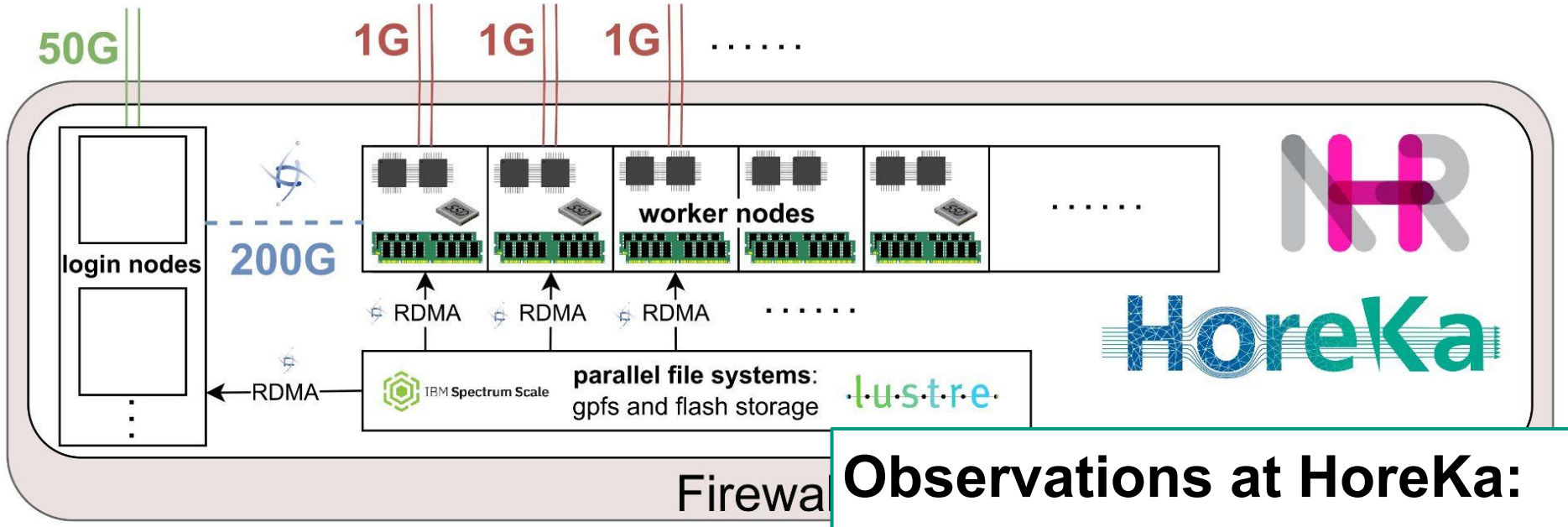
- Several opportunistic resources integrated into the Tier 1 with [COBaID/TARDIS]
- **HoreKa** is integrated **opportunistically** since more than three years

Situation at KIT + HoreKa



- Whole node scheduling
- 1G WAN per WN, 50G for login nodes
- Currently: backfilling of CMS production jobs

Situation at KIT + HoreKa

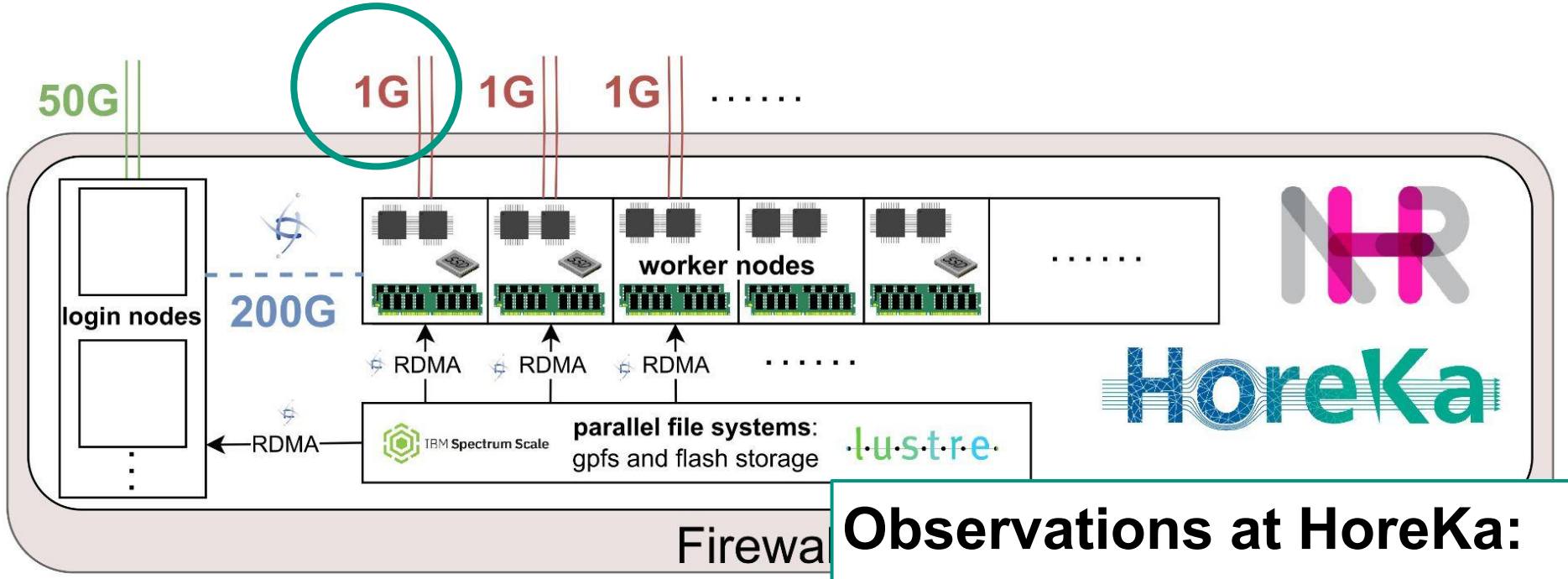


- Whole node scheduling
- 1G WAN per WN, 50G for login nodes
- Currently: backfilling of CMS production jobs

Observations at HoreKa:

- Compared to T1/T3:
  - Lower cpu efficiency
  - Higher failure rate
- Debugging and optimizations are complicated…

Situation at KIT + HoreKa

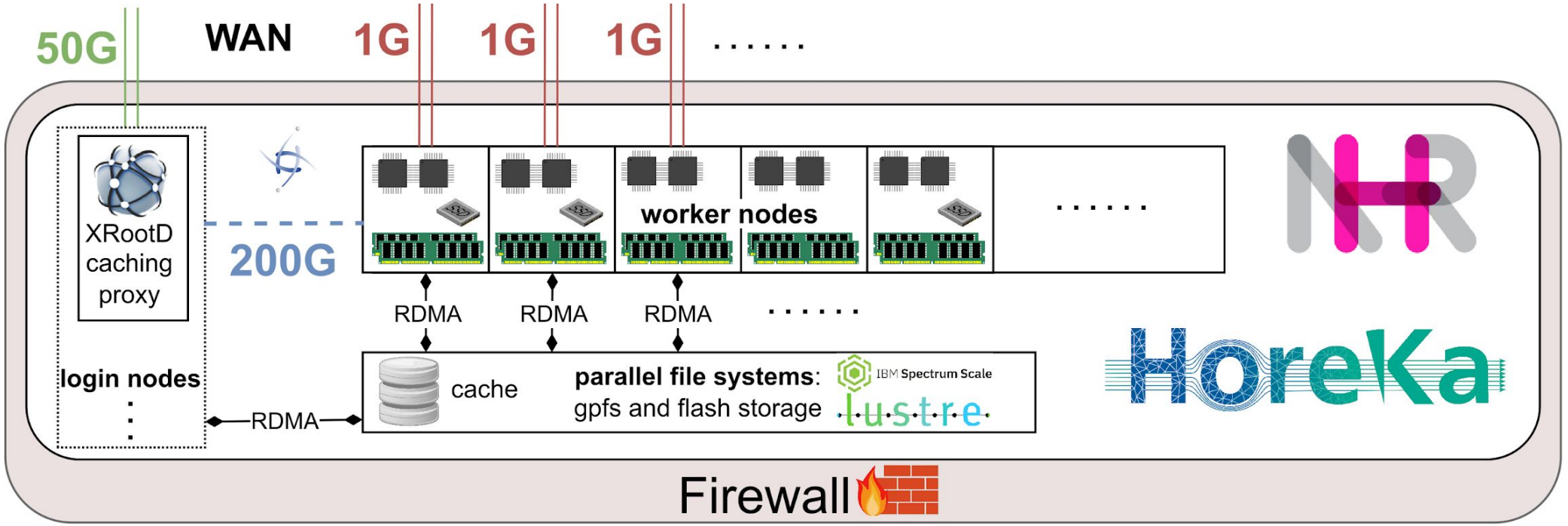


- Whole node scheduling
- 1G WAN per WN, 50G for login nodes
- Currently: backfilling of CMS production jobs

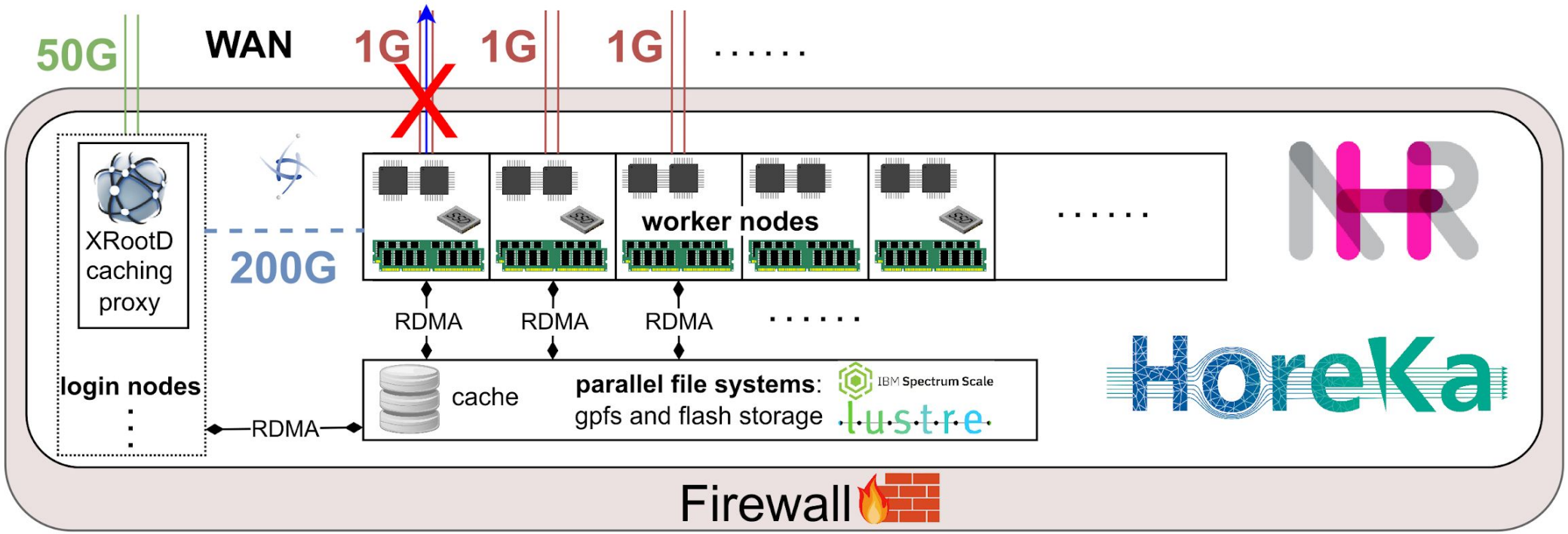
Observations at HoreKa:

- Compared to T1/T3:
  - Lower cpu efficiency
  - Higher failure rate
- Debugging and optimizations are complicated…

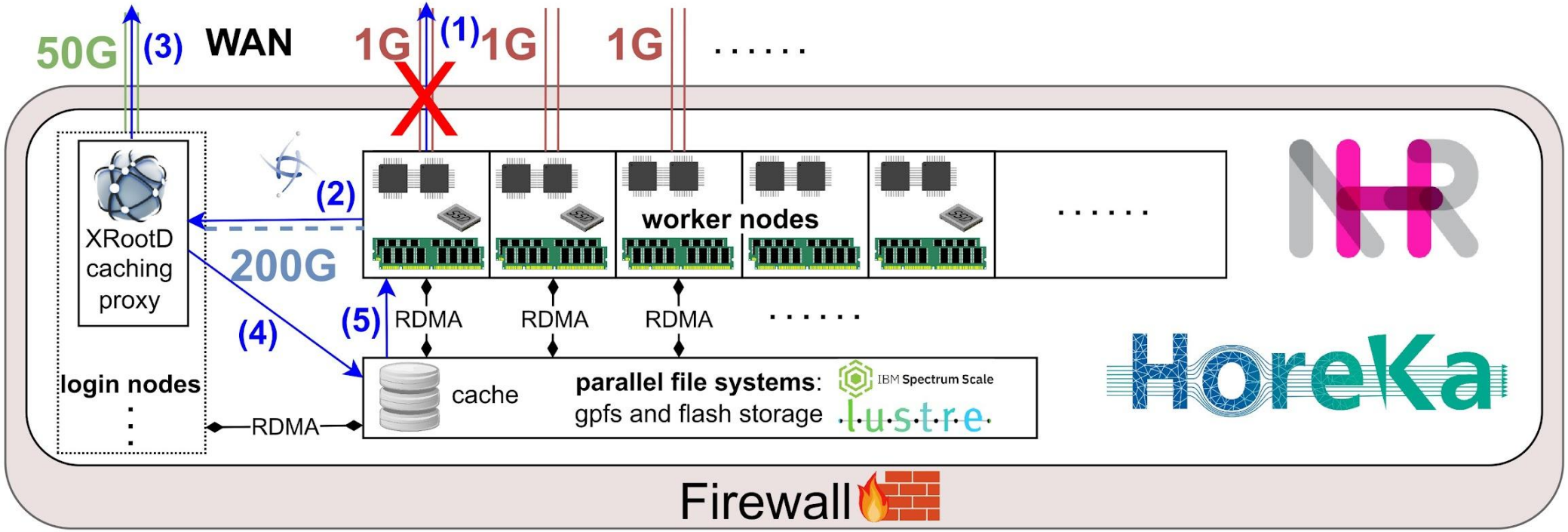
Data-Access Bottleneck Mitigation with XRootD



Data-Access Bottleneck Mitigation with XRootD



Data-Access Bottleneck Mitigation with XRootD



Data-Access Bottleneck Mitigation with XRootD

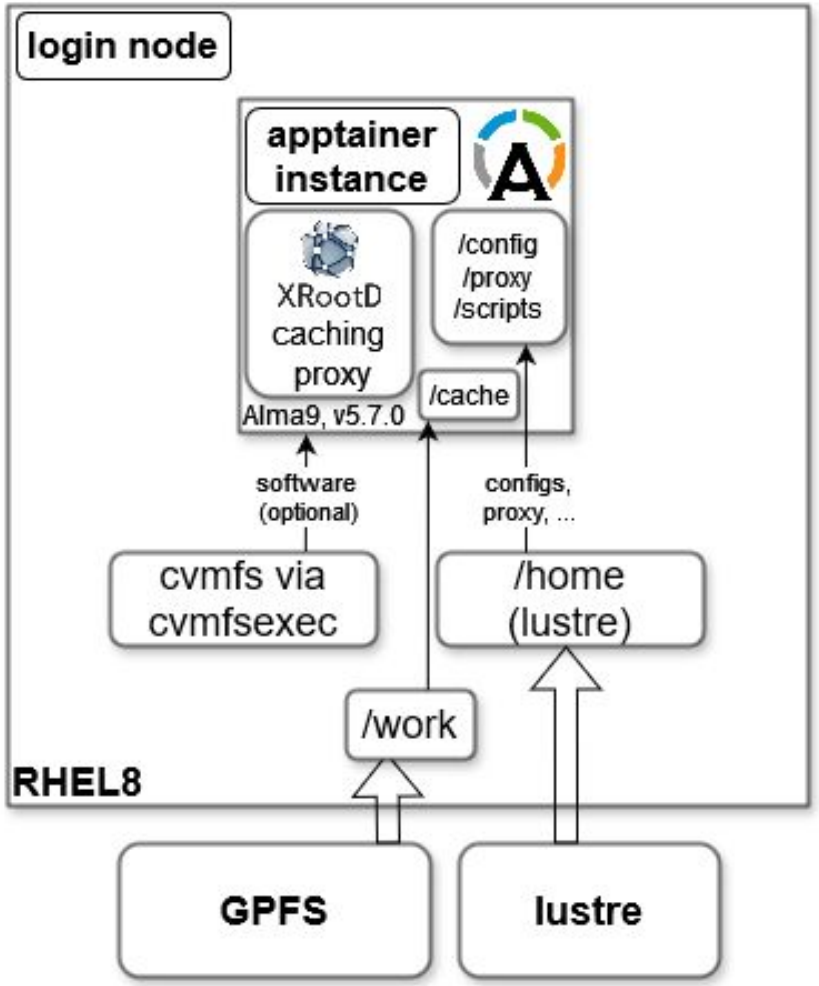
1. Instead of using the 1G external connection of the worker node, we run a (caching) proxy at one of the cluster's login nodes (50G)
2. In the siteconf, we add the proxy as a prefix
3. The transfer is executed over the login node with faster connectivity
4. and cached on the fly (if enabled).
5. The job gets the data via the internal IPoIB network from the login node (or directly from the cache).

Advantages

- Faster external **connectivity**:
  - We use the caching proxy as a sort of “buffer” with **prefetching** enabled
  - The data is **cached** on the parallel FS on the fly, if enabled
- Potential **cache hits** (ideally with dca enabled) increase the transfer speed extremely (several GB/s)
- More info on problems/job failures from the logs
- Useful, even if there is **no data access bottleneck**:
  - Throttling for multi-user scenarios
  - Additional **monitoring**:
    - Apptainer instance stats (RAM, CPU, IO; if enabled)
    - XrootD monitoring (summary + detail)
    - Own stuff: io, iops, cache hits, …

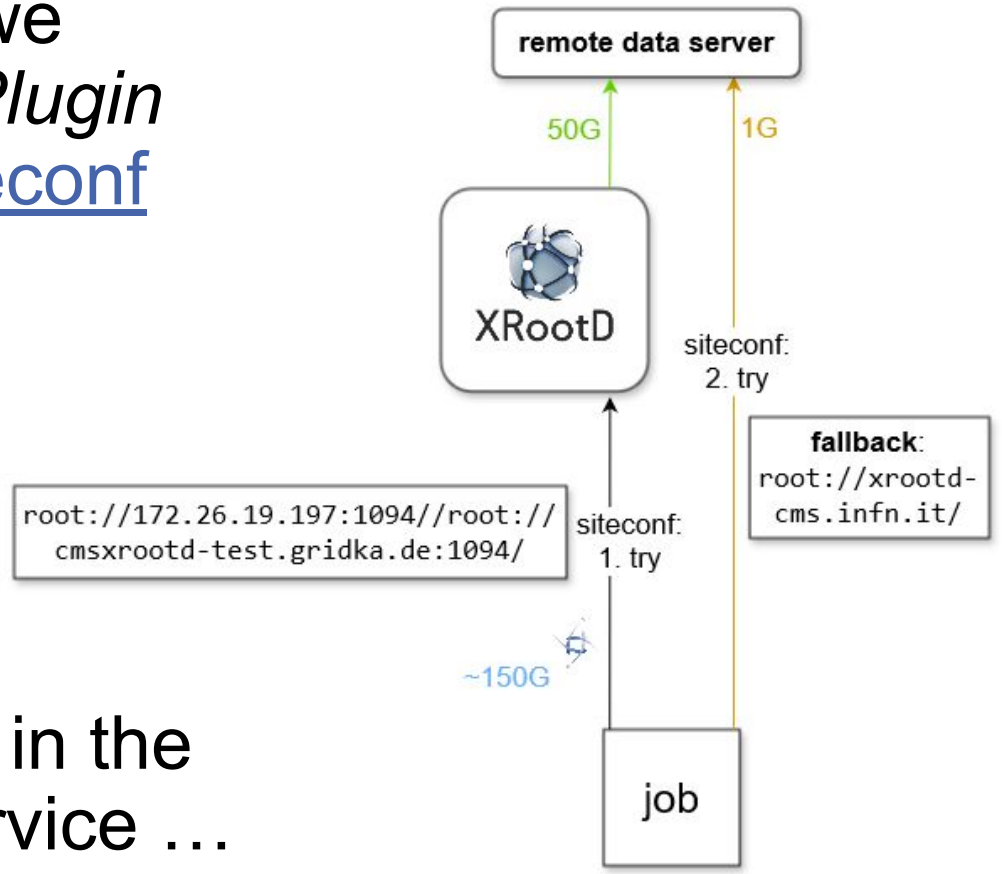
Setup and Configuration: Overview

- Host: RHEL8
- usernamespaces, CGroups v2, systemd user services
- Currently running: XRootD v5.7.0 as Alma9 apptainer instance (image bootstrapped from docker)
- In principle up to 76c and 500GB RAM, 50G WAN
  - But shared with other users (limitation via apptainer instance with CGv2 possible)
  - Usage: 32t, 64GB memory
- 250T quota on gpfs (via IB)



Setup and Configuration: Connection

- For enabling the proxy for transfers, we currently do **not** use the *XrdClProxyPlugin*
- Instead, the proxy is added to the siteconf directly
- Advantages:
  - We can use the intended fallback mechanism if smth fails
  - It is only enabled for file reads
- Disadvantages:
  - Changes require always a change in the repo and a full shutdown of the service …



Setup and Configuration: Caching?

- Caching: Yes or No is a **tough** decision
  - Can be useful:
    - Depends on data sets, cache size, etc etc -> expected rotations
    - E.g. I heard from very positive experiences with caching of user analysis data
    - Overall: currently hard to decide
  - But, from our experience on HPC: very error-prone

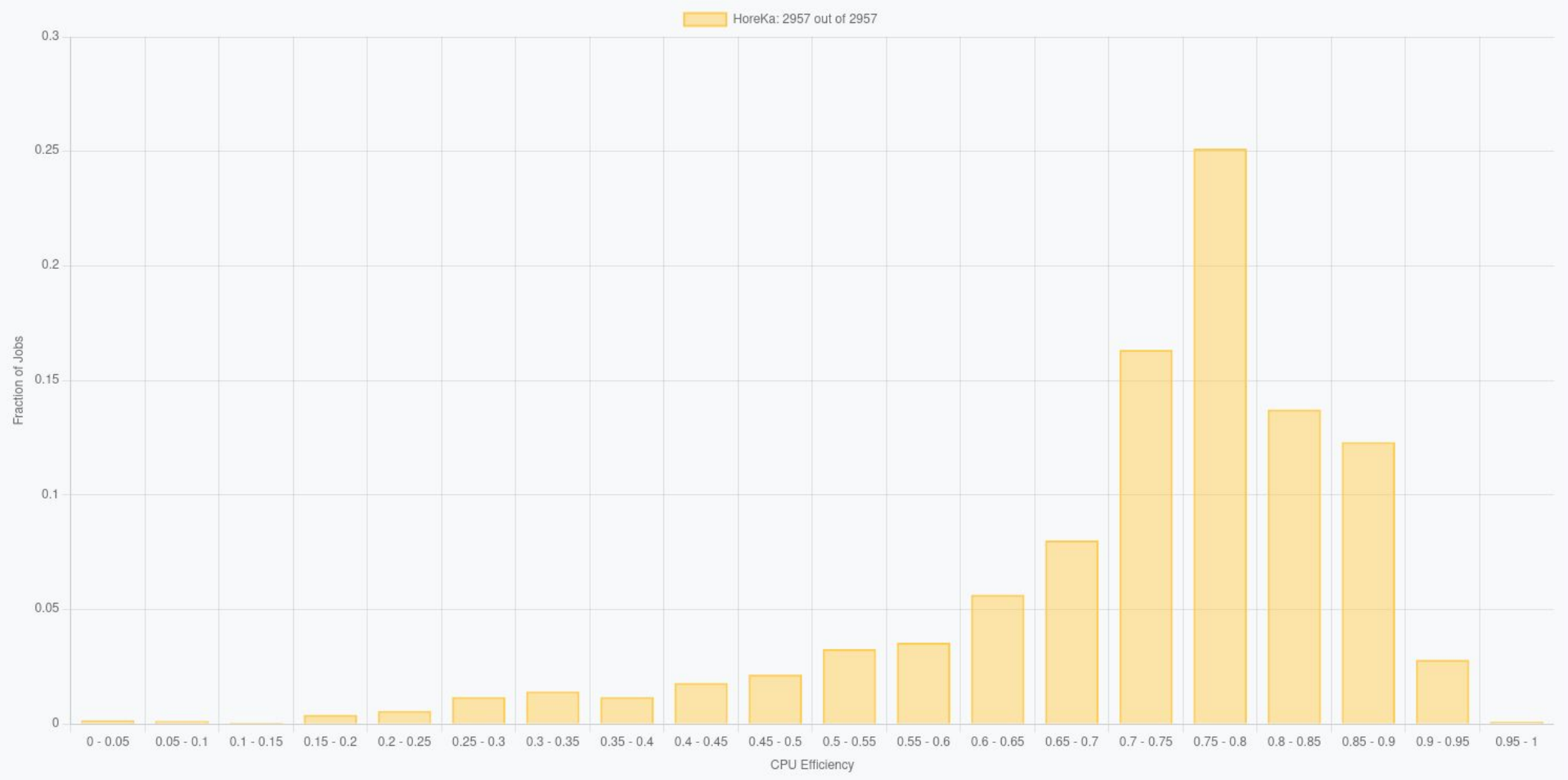
    *(Remark: **NOT** only because of XRootD, but together with CMSSW)*
    - Another example: partly cached files lead to errors with no upward redirection
- We plan to do some studies with and without caching in the future while tracking cache hits

Setup and Configuration: RDMA

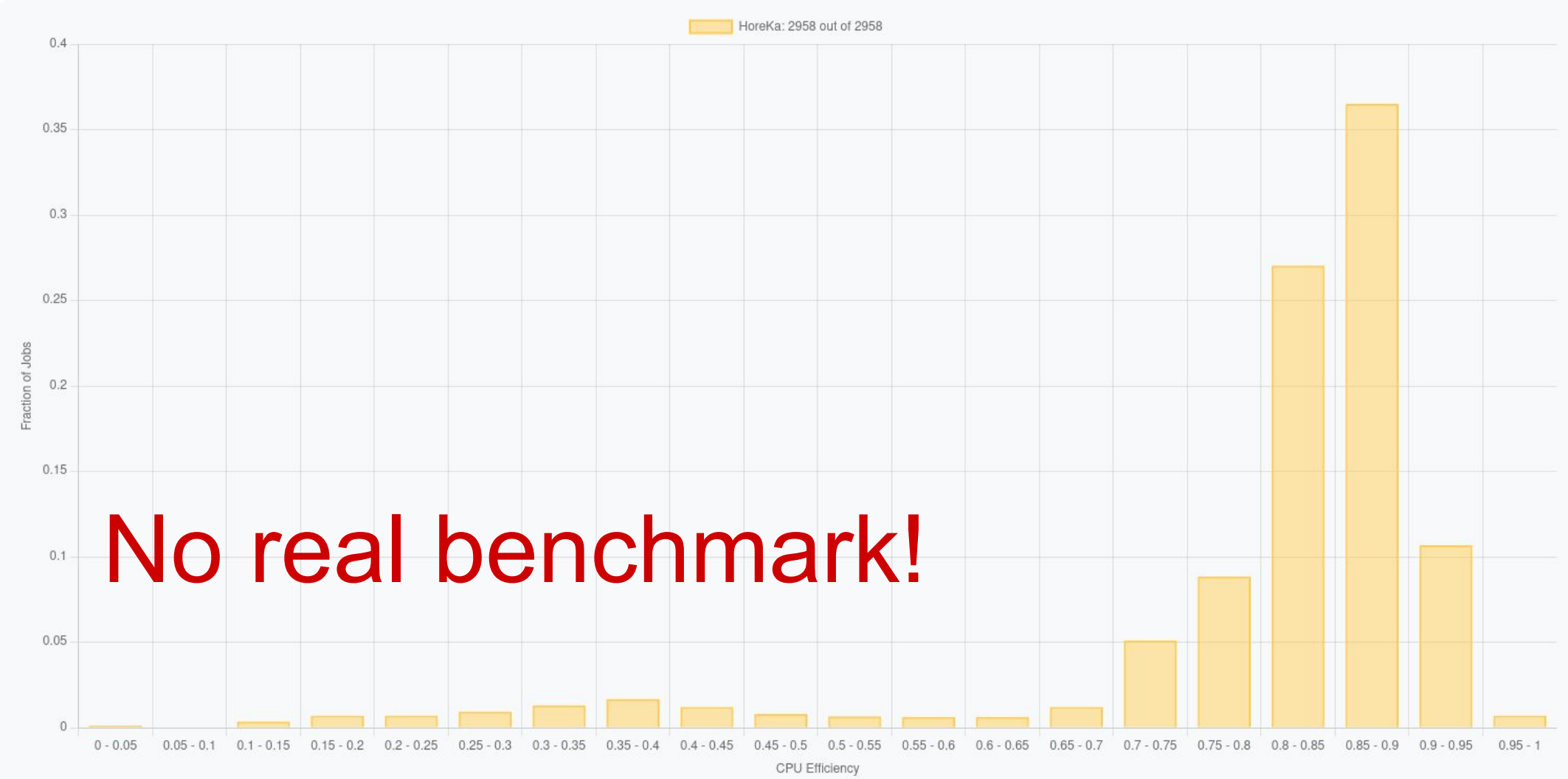
- We currently don't use RDMA natively, but IPoIB for the transfers
- Currently, only IPv4 is possible (no link local v6 addresses in xrootd)
- The cache is also mounted via IPoIB (RDMA)
- We got some complaints from the GPFS team because of the many, many IOPS
- Reason: small blocksizes when caching is enabled
- ideally (FS PoV): pfc.blocksize == FS blocksize, or in general: as big as possible

- dca:
  - Tested, but problematic with containerization
  - Dependent of the campaigns/datasets, e.g. premix rarely completely cached -> in production rather pointless
  - *Would be very useful, if made possible for partially cached files – if possible.*

Results: No Caching Proxy



Results: With Caching Proxy



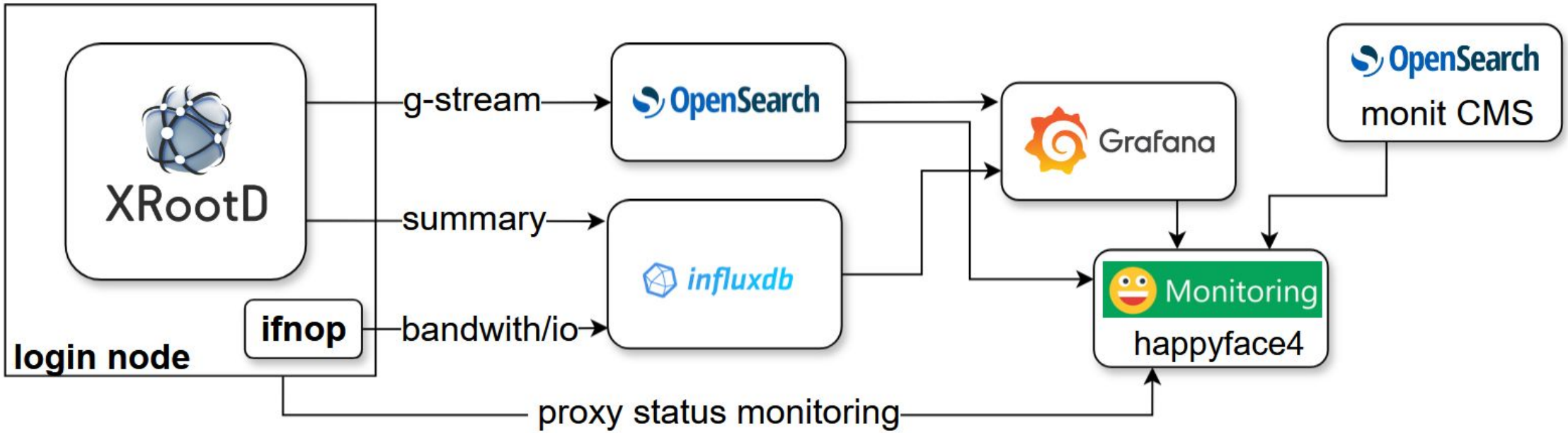
No real benchmark!

Job Mix

- Currently, we only run a subset of the full job mix
- This works well, even for more data intensive WFs
- A full replacement will require the full mix, including *Analysis*
- Caching may be useful for a subset of the job mix (e.g. not RAW)
  - TBD with more monit/benchmarks

- Our proposal:
  - Even the full replacement should not run jobs like *Merge*, as they are just too inefficient on the expensive HPC hardware
  - To achieve the best possible efficiency, it is crucial to consider that

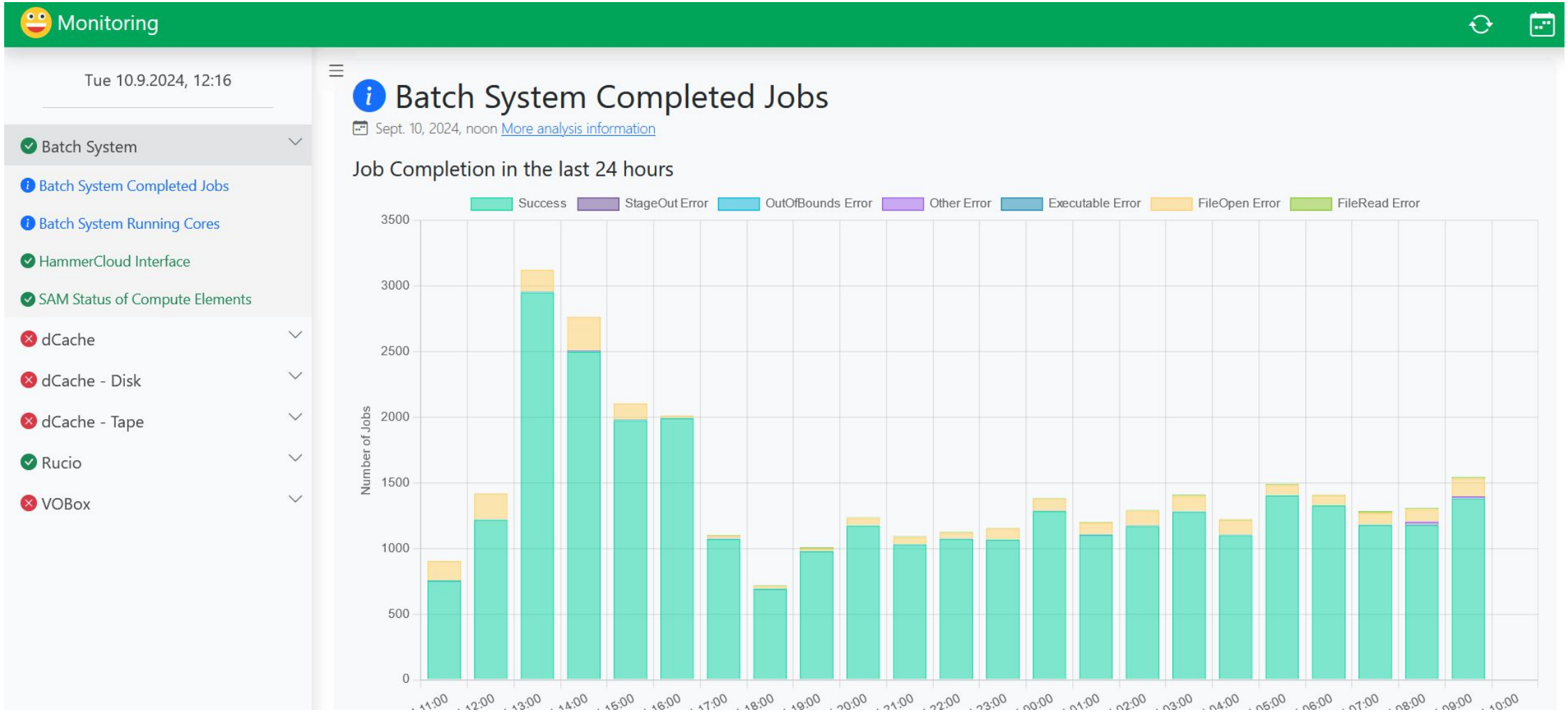
Monitoring



- Many additional monitoring capabilities thanks to XRootD
- Especially cache summary monitoring would be helpful (*on the way*)
- We collect everything and unify it in our meta-monitoring: HappyFace4
- I learned a lot about CMS and created some tools: I, II

Note: not yet fully in production/public

Monitoring


Monitoring
Tue 10.9.2024, 12:16
Batch System
Batch System Completed Jobs
Batch System Running Cores
HammerCloud Interface
SAM Status of Compute Elements
dCache
dCache - Disk
dCache - Tape
Rucio
VOBox
Batch System Completed Jobs
Sept. 10, 2024, noon More analysis information
Job Completion in the last 24 hours
Success
StageOut Error
OutOfBounds Error
Other Error
Executable Error
FileOpen Error
FileRead Error
Number of Jobs
3500
3000
2500
2000
1500
1000
500
0
11:00
12:00
13:00
14:00
15:00
16:00
17:00
18:00
19:00
20:00
21:00
22:00
23:00
00:00
01:00
02:00
03:00
04:00
05:00
06:00
07:00
08:00
09:00
10:00

Outlook and Plans

- Further improvements:
  - Improve the caching: only cache certain sites (e.g. NOT GirdKa)
  - Introduce **dedicated** transfer nodes:
    - Instead of using the login node, move the caching proxy to the T1
    - A direct connection from HPC to the T1 will increase the transfer speed a lot

- Plans:
  - Make the most out of all monitoring
  - Switch to tokens (yikes!)
  - Develop a benchmark mechanism
  - Investigate feasibility and caching efficiency
  - Further data analysis for comparing HPC with “normal” grid sites
  - Documentation and publishing (probably CHEP proceedings)

Conclusion

- The setup runs very smooth and is easily adaptable
- Failure rate and CPU efficiency overall improved – jobmix still important
- XRootD helps a lot in terms of site operation by providing log/monit data
  - We are pretty good in identifying problematic WFs and sites

- @CMS: requestable benchmark WFs for site comparison would be nice

further details: ACAT poster

Karlsruhe Institute of Technology

THANKS!

Requests

- More info/docu/examples would be nice
  - What configurations are reasonable for what?
  - What has which impact on performance?
  - Some examples for “default” stuff would be great.
- More monitoring and statistics in the summary monit would be nice (especially for the evaluation of caching), e.g.:
  - transfer speed
  - cache rate

- CMSSW source disabling issue
- link local IPv6 support
- native RDMA
- credential forwarding for XRootD Proxies

BU: link local addresses

```
7: ib0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 2044 qdisc mq state UP group default qlen 256
    link/infiniband 00:00:10:29:fe:80:00:00:00:00:00:00:04:3f:72:03:00:e8:d1:6a brd 00:ff:ff
    inet 172.26.19.197/17 brd 172.26.127.255 scope global noprefixroute ib0
       valid_lft forever preferred_lft forever
    inet6 fe80::63f:7203:e8:d16a/64 scope link
```

```
[scc-sdm-hep-0001@hkn0003 ~]$ ping fe80::63f:7203:e8:d16a
PING fe80::63f:7203:e8:d16a(fe80::63f:7203:e8:d16a) 56 data bytes
ping: sendmsg: Invalid argument
^C
--- fe80::63f:7203:e8:d16a ping statistics ---
1 packets transmitted, 0 received, 100% packet loss, time 0ms

[scc-sdm-hep-0001@hkn0003 ~]$ ping fe80::63f:7203:e8:d16a%ib0
PING fe80::63f:7203:e8:d16a%ib0(fe80::63f:7203:e8:d16a%ib0) 56 data bytes
64 bytes from fe80::63f:7203:e8:d16a%ib0: icmp_seq=1 ttl=64 time=1.46 ms
64 bytes from fe80::63f:7203:e8:d16a%ib0: icmp_seq=2 ttl=64 time=0.179 ms
64 bytes from fe80::63f:7203:e8:d16a%ib0: icmp_seq=3 ttl=64 time=0.099 ms
64 bytes from fe80::63f:7203:e8:d16a%ib0: icmp_seq=4 ttl=64 time=0.285 ms
64 bytes from fe80::63f:7203:e8:d16a%ib0: icmp_seq=5 ttl=64 time=0.162 ms
^C
```

```
Apptainer> xrdcp root://[fe80::63f:7203:e8:d16a%ib0]//root://xrootd-cms.infn.it:1094//store/mc/RunIISummer20ULPrePremix/Neutrino_E-10_gun/PREMIX/UL17_106X_mc2017_realistic_v6-v3/210005/0DA9B681-83C2-9442-89B8-52E7053B50E0.root .
[0B/0B][100%][==================================================][0B/s]
Run: [FATAL] Invalid address:  (source)
```

BU: fallback + disable redirector

```
09-Sep-2024 05:22:21 UTC  Closed file file:../cmsRun2/RAWSIMoutput.root
----- Begin Fatal Exception 09-Sep-2024 05:22:21 UTC-----------------------
An exception of category 'FallbackFileOpenError' occurred while
   [0] Constructing the EventProcessor
   [1] Constructing module: class=PreMixingModule label='mixData'
   [2] Calling RootFileSequenceBase::initTheFile()
   [3] Calling StorageFactory::open()
   [4] Calling XrdFile::open()
Exception Message:
Failed to open the file 'root://xrootd-cms.infn.it//store/mc/RunIISummer20ULPrePremix/Neutrino_E-10_gun/PREMIX/UL17_106X_mc2017_realistic_v6-v3/210005/0DA9B
681-83C2-9442-89B8-52E7053B50E0.root'
   Additional Info:
      [a] XrdCl::File::Open(name='root://172.26.19.197//root:/cmsxrootd-test.gridka.de:1094/store/mc/RunIISummer20ULPrePremix/Neutrino_E-10_gun/PREMIX/UL17_
106X_mc2017_realistic_v6-v3/210005/0DA9B681-83C2-9442-89B8-52E7053B50E0.root', flags=0x10, permissions=0660) => error '[FATAL] Connection error' (errno=0, c
ode=108)
      [b] Remote server already encountered a fatal error; no redirections were performed.
      [c] Input file root://172.26.19.197:1094//root://cmsxrootd-test.gridka.de:1094//store/mc/RunIISummer20ULPrePremix/Neutrino_E-10_gun/PREMIX/UL17_106X_m
c2017_realistic_v6-v3/210005/0DA9B681-83C2-9442-89B8-52E7053B50E0.root could not be opened.
Fallback Input file root://xrootd-cms.infn.it//store/mc/RunIISummer20ULPrePremix/Neutrino_E-10_gun/PREMIX/UL17_106X_mc2017_realistic_v6-v3/210005/0DA9B681-8
3C2-9442-89B8-52E7053B50E0.root also could not be opened.
Original exception info is above; fallback exception info is below.
      [d] XrdCl::File::Open(name='root://xrootd-cms.infn.it//store/mc/RunIISummer20ULPrePremix/Neutrino_E-10_gun/PREMIX/UL17_106X_mc2017_realistic_v6-v3/210
005/0DA9B681-83C2-9442-89B8-52E7053B50E0.root', flags=0x10, permissions=0660) => error '[ERROR] Server responded with an error: [3011] No servers are availa
ble to read the file.
' (errno=3011, code=400). No additional data servers were found.
      [e] Last URL tried: root://xrootd-cms.infn.it:1094//store/mc/RunIISummer20ULPrePremix/Neutrino_E-10_gun/PREMIX/UL17_106X_mc2017_realistic_v6-v3/210005
/0DA9B681-83C2-9442-89B8-52E7053B50E0.root?tried=
      [f] Problematic data server: xrootd-cms.infn.it:1094
      [g] Disabled source: xrootd-cms.infn.it:1094
----- End Fatal Exception -------------------------------------------------
Complete
```

/eos/cms/store/logs/prod/recent/PRODUCTION/cmsunified_task_GEN-RunIISummer20UL17wmLHEGEN-00023__v1_T_240510_062351_1095/GEN-RunIISummer20UL17wmLHEGEN-00023_0/vocms0281.cern.ch-6370591-0-log.tar.gz

BU: COBaID/TARDIS