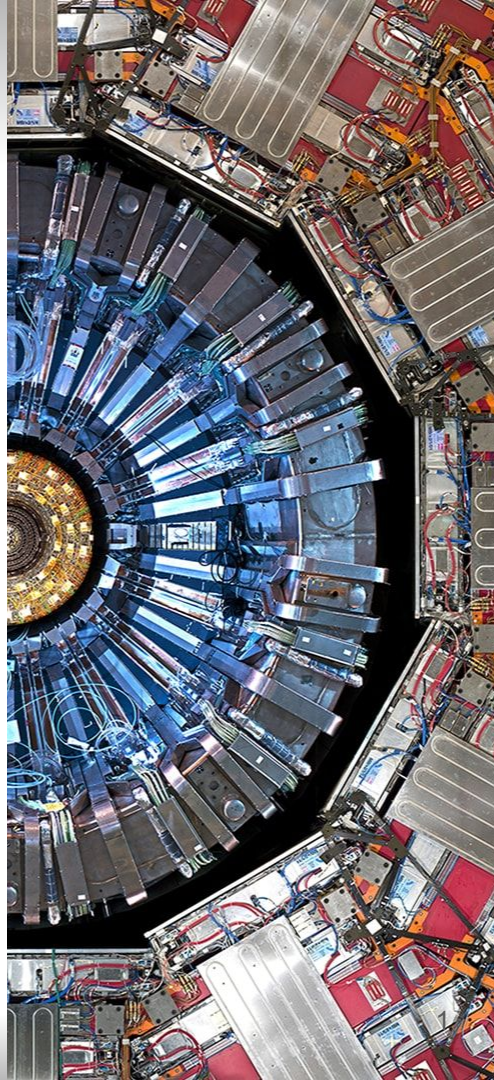
# XRootD Remote-Reads for WLCG Data Challenge

Rahul Chauhan
XRootD and FTS Workshop
September 14 - 18, 2026
CNRS Délégation Rhône Auvergne



# We need XRootD Remote Reads in DC27

Previous data challenges compensated for CMS Remote read traffic by compensating load with FTS

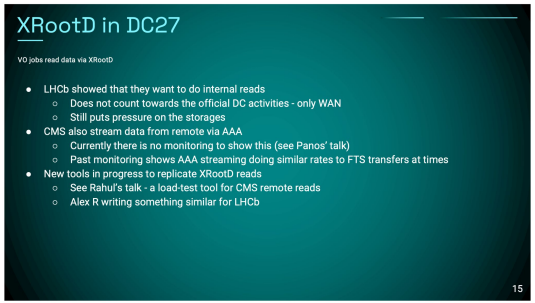
- This only added more HTTP traffic
- Only exercised the full download mode
- Source and destinations involved storage elements and not worker nodes

We need to simulate the worker node traffic

- Hold the rates at planned rate to conform to Data Challenge requirements
- Involve vector reads and test various read patterns
- Mimic premixing jobs

[WLCG/DOMA Data Challenge 2024: Final Report]

[Data Challenge 2024 – CMS Activities | EPJ Web of Conferences]




[XRootD and FTS Workshop @ CNRS Rhône Auvergne Delegation Lyon, France: Data Challenge 2027]

# Target to achieve: 25% share of the total CMS traffic

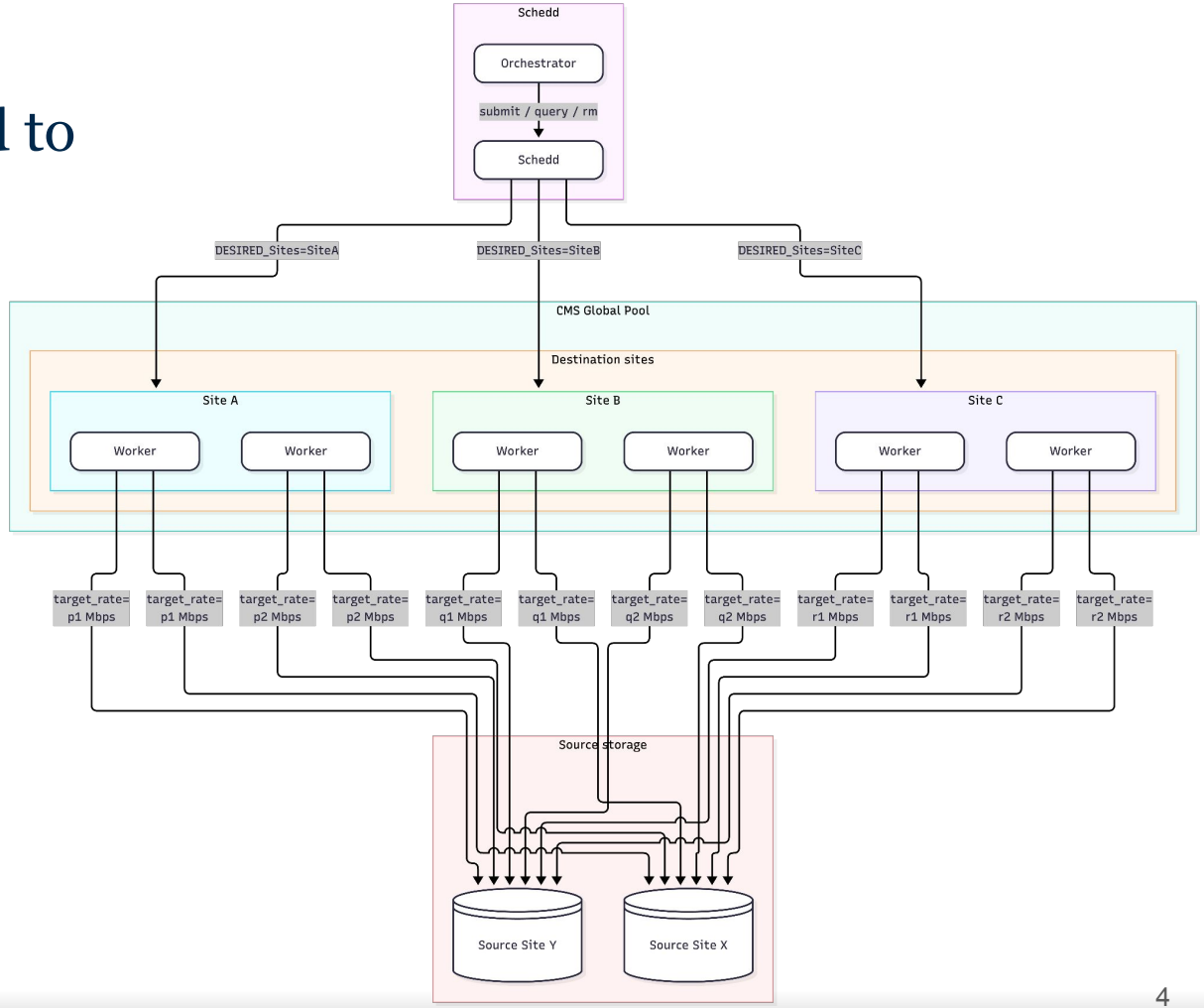
My current understanding of the targets based on previous papers and presentations.

Actual numbers would be decided in upcoming DOMA meetings and WLCG workshop.

| | **DC24 (25%)** | | **DC27 (50%)** | | **DC29 (100%)** | |
|---|---|---|---|---|---|---|
| | **Remote Reads** | **File Copying** | **Remote Reads** | **File Copying** | **Remote Reads** | **File Copying** |
| Base Target | 0 Gbps | 500 Gbps | 0 Gbps | 1 Tbps | 0 Gbps | 2 Tbps |
| Flexible Target | 250 Gbps | 750 Tbps | 375 Gbps | 1.125 Tbps | 750 Gbps | 3 Tbps |

# Run jobs across the grid to simulate remote reads

A fleet of jobs: each job independently tries to hold the target rate asked for by the orchestrator.



# Drink like an Italian: Guide to Data Challenge Remote Read Cocktails

# Preparation

1 Parts corpus: List of datasets to read over the link

# Input matrix: It all starts here!

Gbps · source \ dest

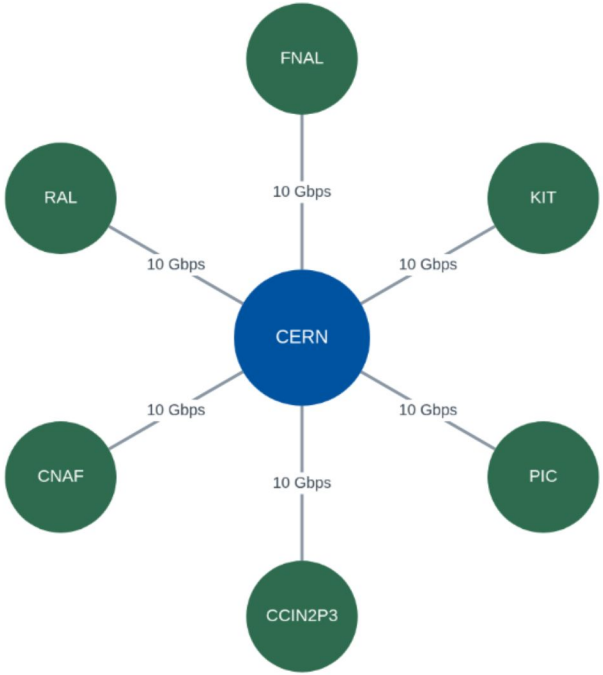
| | T2_CH_CERN | T1_US_FNAL | T1_DE_KIT |
|---|---|---|---|
| T2_CH_CERN | **100** | 10 | 10 |
| T1_US_FNAL | 10 | **100** | 10 |
| T1_DE_KIT | 10 | 10 | **100** |

```
plan:
  max_job_rate_gbps: 0.2        // cap on jobs's rate
  min_jobs_per_cell: 5          // floor on jobs per
  read_size_bytes: 8000000      // Chunk size
  max_bytes_per_file: 32000000  // Session size
  job_duration_s: 180           // hold duration
  overshoot_percent: 5          // target increment
```

# Any topology can be exercised with the input matrix

**Hub-and-spoke — T2_CH_CERN**

Rows = source, columns = dest (DESIRED_Sites), cells = Gbps. Each T1 dest holds 10 Gbps from CERN.

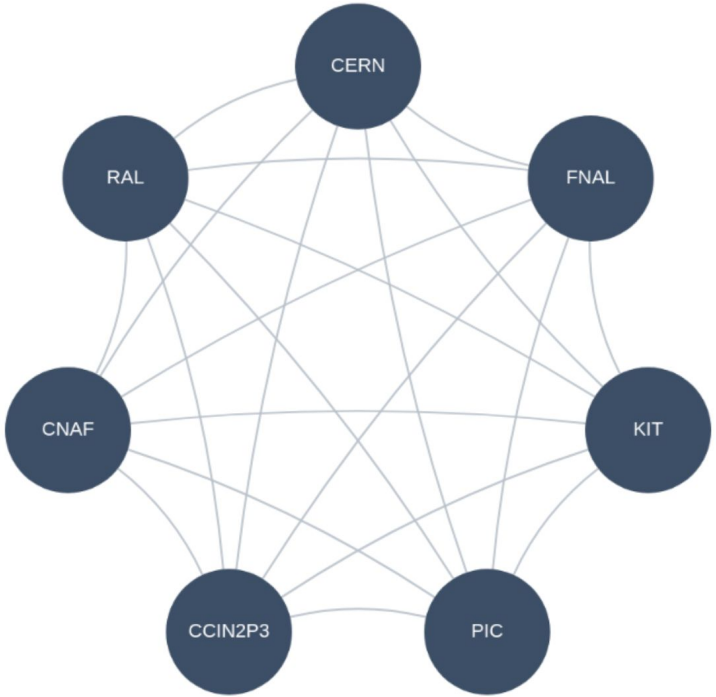



| source | FNAL | KIT | PIC | CCIN2P3 | CNAF | RAL |
|---|---|---|---|---|---|---|
| T2_CH_CERN | 10 | 10 | 10 | 10 | 10 | 10 |

# Any topology can be exercised with the input matrix

**Mesh — T1 sites**

Rows = source, columns = dest (DESIRED_Sites), cells = Gbps. Off-diagonal cells are 5 Gbps; self-links stay 0.




| source | CERN | FNAL | KIT | PIC | CCIN2P3 | CNAF | RAL |
|---|---|---|---|---|---|---|---|
| T2_CH_CERN | 0 | 5 | 5 | 5 | 5 | 5 | 5 |
| T1_US_FNAL | 5 | 0 | 5 | 5 | 5 | 5 | 5 |
| T1_DE_KIT | 5 | 5 | 0 | 5 | 5 | 5 | 5 |
| T1_ES_PIC | 5 | 5 | 5 | 0 | 5 | 5 | 5 |
| T1_FR_CCIN2P3 | 5 | 5 | 5 | 5 | 0 | 5 | 5 |
| T1_IT_CNAF | 5 | 5 | 5 | 5 | 5 | 0 | 5 |
| T1_UK_RAL | 5 | 5 | 5 | 5 | 5 | 5 | 0 |

# Any topology can be exercised with the input matrix

**LAN — dest WN reads the local SE**

Same matrix: dest = source. Jobs land at the site and read that site's storage. Diagonal cells only; off-diagonal stay 0.

| source | CERN | FNAL | KIT | PIC | CCIN2P3 | CNAF | RAL |
|---|---|---|---|---|---|---|---|
| T2_CH_CERN | **10** | 0 | 0 | 0 | 0 | 0 | 0 |
| T1_US_FNAL | 0 | **10** | 0 | 0 | 0 | 0 | 0 |
| T1_DE_KIT | 0 | 0 | **10** | 0 | 0 | 0 | 0 |
| T1_ES_PIC | 0 | 0 | 0 | **10** | 0 | 0 | 0 |
| T1_FR_CCIN2P3 | 0 | 0 | 0 | 0 | **10** | 0 | 0 |
| T1_IT_CNAF | 0 | 0 | 0 | 0 | 0 | **10** | 0 |
| T1_UK_RAL | 0 | 0 | 0 | 0 | 0 | 0 | **10** |

# Spark Jobs Prepare the input to read from each site

From the input rate matrix: Filelist per source

- This can also be achieved with Rucio
- `list_datasets_per_rse(rse)` then *`list_replicas(..., rse_expression=rse)`*

Ensure uniqueness (avoid cache hits / repeated reads)

- From a given source, an LFN is assigned to at most one destination
- Chunking that list into multiple jobs on the same dest keeps them disjoint.
- The same LFN may appear on CERN→D and FNAL→D if the file exists at both sources
- Between multiple jobs at the same site, random draws reduce overlap

# XrdHover

2 parts Payload Process: Job that reads data

# Payload process and its configurations

- C++ client over XrdCl 6
- Reads root files for the configured wall clock time
- Picks a files at random from the provided source file list

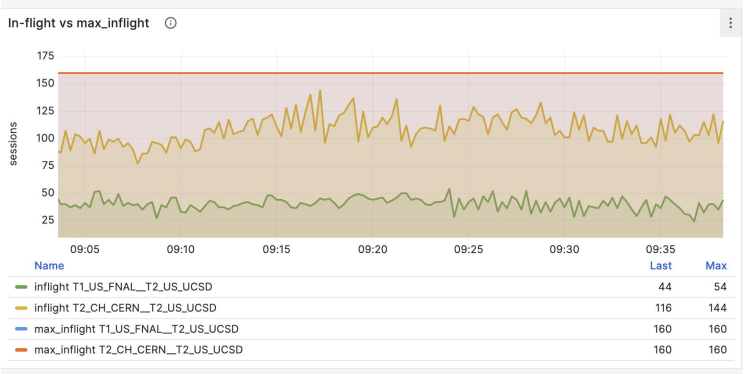
```
xrdhover validate job.json
xrdhover run job.json
xrdhover probe job.json

xrdhover run --endpoint root://host:1094/ \
  --filelist files.txt --rate 1Gbps --duration 30m
```

```
{
  "duration": "30m",
  "seed": 1,
  "auth": { "mode": "x509" },
  "targets": [{
    "endpoint": "root://eoscms…/",
    "filelist": "files.txt",
    "target_rate": "1Gbps",
    "max_inflight": 16,
    "pattern": {
      "type": "sequential",
      "read_size": "8MB",
      "max_bytes": "32MB"
    }
  }]
}
```

# Token bucket algorithm to rate limit itself

**Bucket Properties**

- Bucket capacity = Read Size
  - This means no spikes / rate attempt to go beyond the target
  - No tokens added to bucket once bucket is full
- Refill rate = target rate




**Admission Policy**

Continue reads on currently open files before opening new ones (upto max_inflight config)

# It's a lightweight process

Doesn’t conform to the 1:2 ratio for cpu:memory

- Doesn’t require significant resource

Average resource consumption metrics from around 1K XrdHover Jobs each with a target rate of 100 Mbps

# TCKestrel

3 parts Controller: The orchestrator of Jobs

Video of a Kestrel Hovering

# Fleet controller

The interface for the DC Operator

- TCKestrel ideally runs alongside a schedd
- Pins destinations “DESIRED_Sites” and  get PFNs from rucio

```
tckestrel plan     -c controller.yaml
tckestrel payload  -c controller.yaml
tckestrel resolve  -c controller.yaml
tckestrel render   -c controller.yaml --validate
tckestrel submit   -c controller.yaml
tckestrel jobs     -c controller.yaml
tckestrel metrics  -c controller.yaml --background
tckestrel rm       -c controller.yaml
```

*** The control loop part is still being designed**

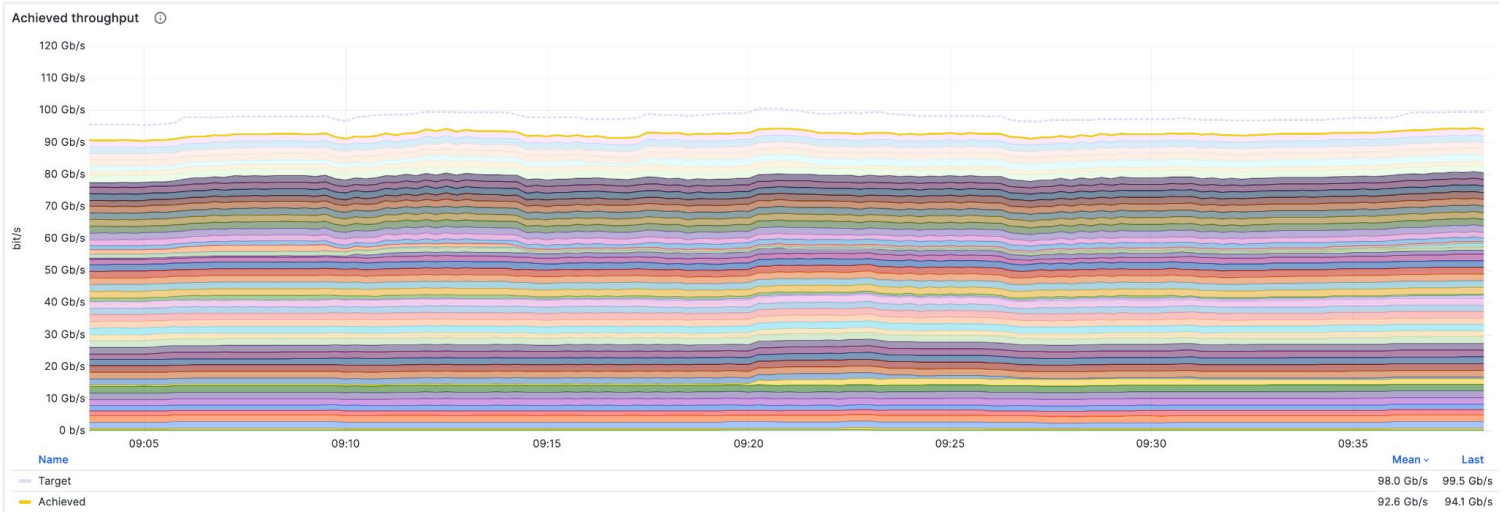
# Controller.yaml

```yaml
campaign_id: premix-dc27
matrix: matrix.csv
filelists_dir: filelists/

plan:
  max_job_rate_gbps: 0.2
  min_jobs_per_cell: 1
  job_duration_s: 1800
  target_rate_sum_gbps: 10
  overshoot_percent: 10
  read_size_bytes: 8000000
  max_bytes_per_file: 32000000

payload:
  xrdhover_version: "0.4.5"
  cmssw: CMSSW_20_1_0_pre2
  required_os: rhel9
```

```yaml
submit:
  chirp_prom_path: ~/alloy/textfile_collector/
  snapshot_interval_s: 30
  condor_pool: vocms4100.cern.ch
  condor_schedd: crab3@vocms059.cern.ch
  keep_claim_idle_s: 0
  requirements:
    arch: X86_64
    opsys: LINUX
    cpus: 1
    memory_mb: 1024
    disk_mb: 1024

auth:
  accounting_group: highprio
  proxy:
    min_ttl_s: 25200

loop:
  max_idle_jobs_per_cell: 20
  max_idle_jobs_per_dest: 40
  max_jobs_per_dest: 50
  max_jobs_global: 2000
  submits_per_minute: 10
  min_job_lifetime_s: 120
  recycle_lfn_after_s: 7200
  rate_deadband_frac: 0.05
  ramp_jobs_per_tick: 1
```

# Tier-2 100 Gbps challenge

European and US Tier 2 sites read from both CERN and FNAL at target rates of 1 Gbps per source
- 5 % overshoot configured




A sample slice of the grafana monitoring from the xrdhover dashboards; the ramp-up throughput has been conveniently cut off ;)

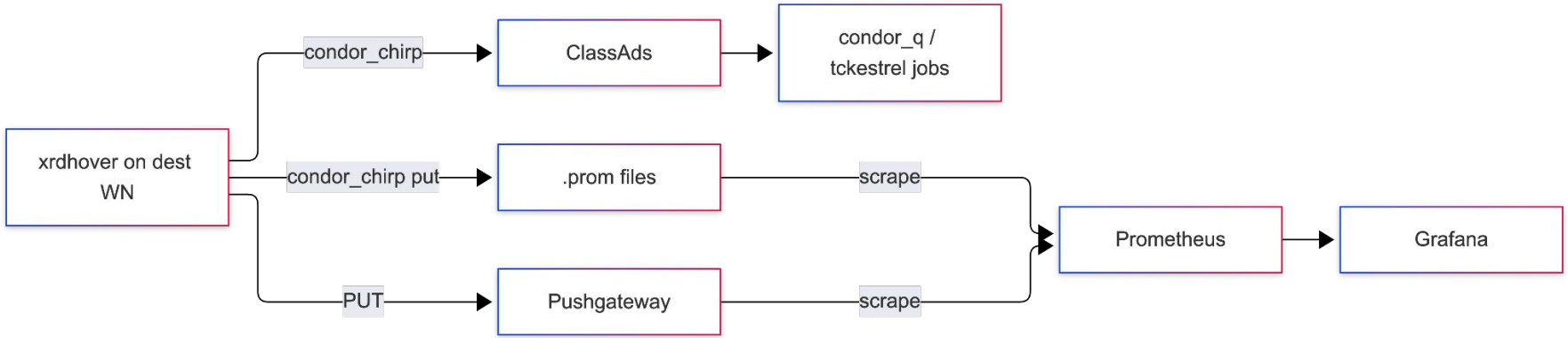
# Monitoring

Some Garnish: Observability

# Architecture

Serves two purposes

- Observability
- Feedback for the control loop

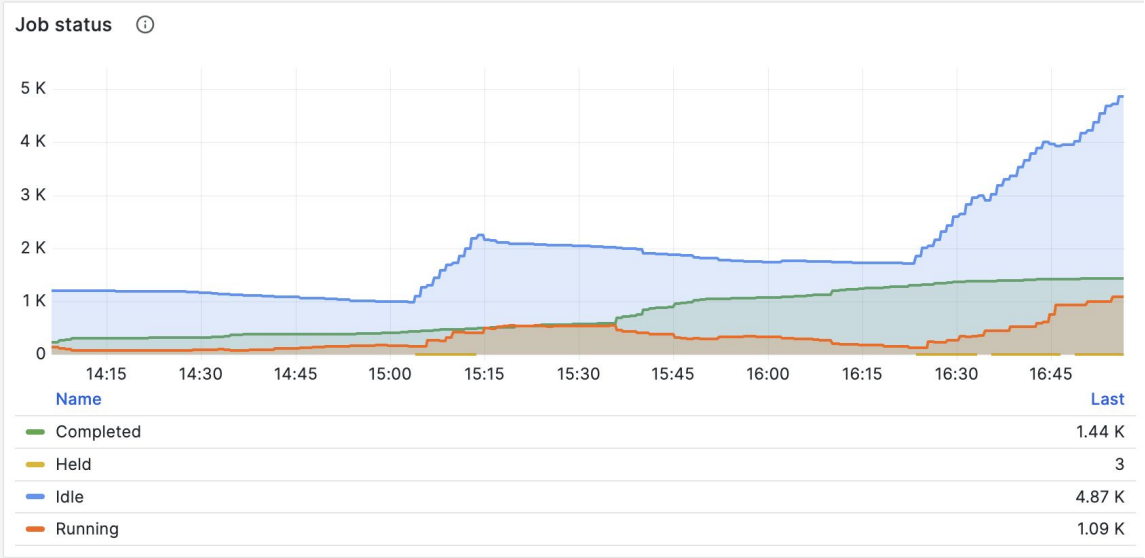


# Getting your job to run is hard to predict or force
- The control loop solves this

Even with a highpriority status within the Global Pool it's hard to predict when does the job go into running state

Submitting thousand jobs is not fast

- This increases the  ramp up time; but does not cause trouble otherwise



# The links are not uniform

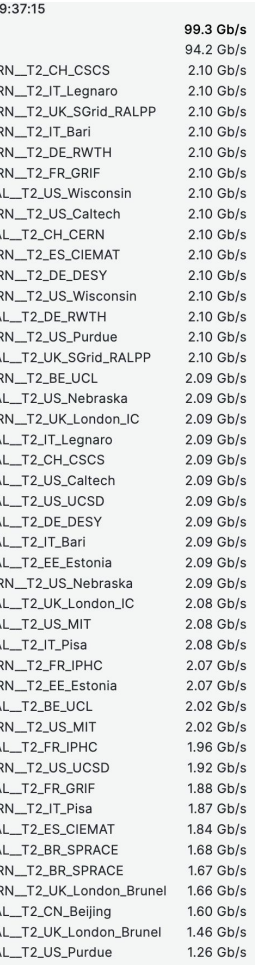
Some links can easily reach the requested rates and go beyond

Others can struggle with them

Plan for max-throughput tests at random intervals to get a better estimate of site throughput*?

- Ask sites for their own numbers?

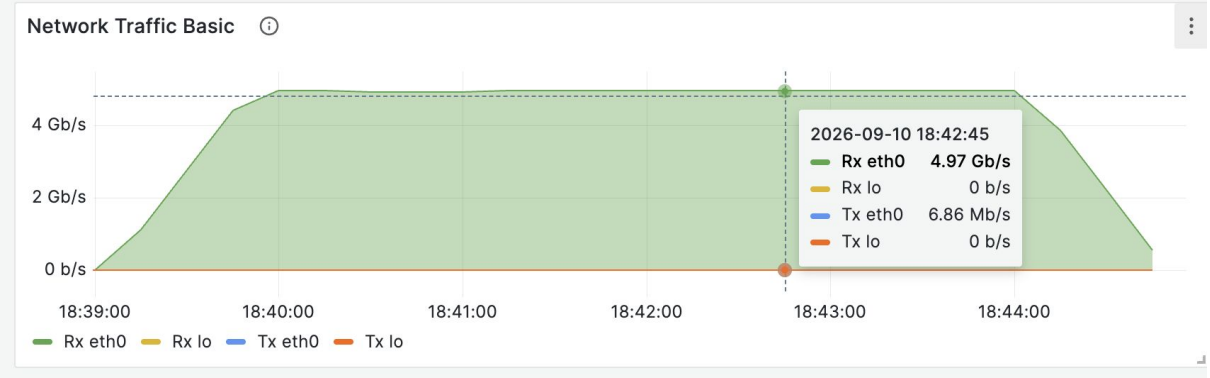
*Or quietly run tests at midnight and hope no one would know?



# Should sites really not look into verifying their network monitoring?

- Is being 10% off really ok?

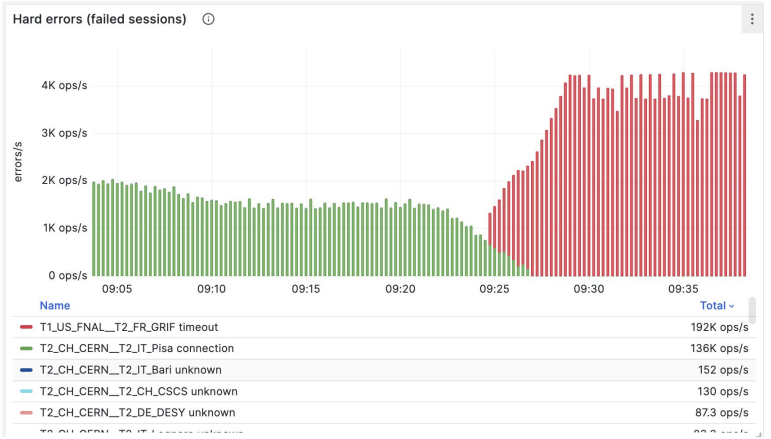
```
=== run summary ===
run_id:        T2_CH_CERN__cmsxrd-readgen
job_id:        nic-validate
elapsed:       300.101 s
sessions:      5804 ok / 0 fail
bytes:         185.72 GB (185720000000)
ops:           23215
achieved:      4.95 Gbps
target:        5.00 Gbps (from '5Gbps')
results:       /tmp/xrdhover-nic-validate/T2_CH_CERN__cmsxrd-readgen
by_cms_site:
```



# Error reporting

Following error classes per session:

| CLASS | MAPS FROM |
|---|---|
| auth | 3010, not authorized, permission denied, GSS |
| not_found | 3001, no such file |
| timeout | ETIMEDOUT, timed out |
| trust | certificate verify, unknown CA |
| connection | TLS, handshake, ECONNREFUSED, ECONNRESET |
| redirect_loop | redirect limit (AAA) |
| client_error | kXR_InvalidRequest, kXR_NotFile |
| server_error | other kXR 3xxx |
| unknown | leftover status or message |




Considering taking errors into account for the controller

- Not sure if that would be useful

# More Ideas to Explore

Analyse current traffic patterns from jobs

- Use them as sampling distributions for read patterns

# Thank you!