Transatlantic high throughput (400 Gb/s) tests between UMass NET2 and Prague

What a cache-oriented production-path test exposed

Eduardo Bach (UMass Amherst / NET2)

based on the work performed in collaboration with Petr Vokac, Will Leight, Jessa Westclark, Dale Carder, and Rafael Coelho Lopes de Sa

From SC25 to PRG: testing the production stack at scale

Data transfers from UMass to the SCinet cluster and to the Caltech booth

gfal-copy: 700 Gb/s (UMass→SC25) + 400 Gb/s (SC25→UMass)

Redirection mode door used.

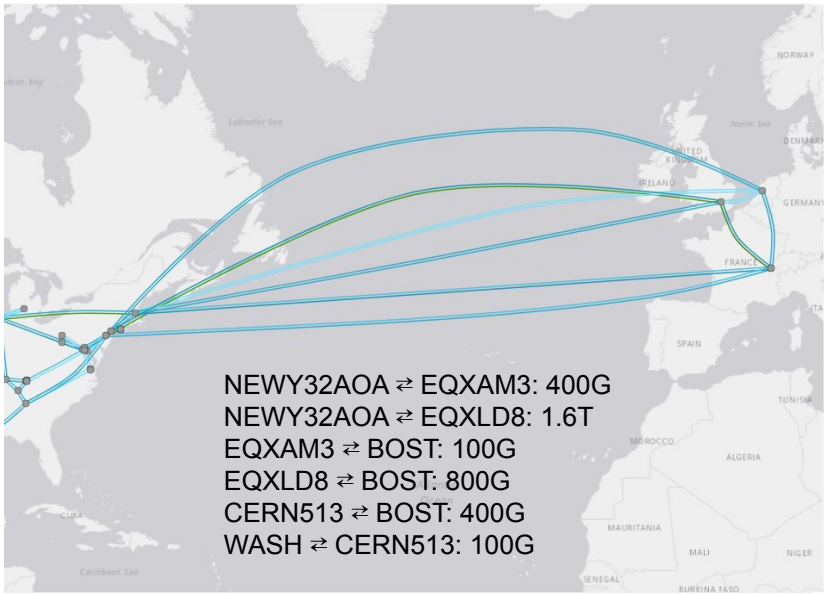
Lesson from SC25: high-rate storage transfers depend on several layers, not only disks.

First test: production path, cache-oriented workload

Test design: production path, reduced disk variability

- Small file list distributed across storage servers.
- Same files copied repeatedly during each pass.
- First read can touch disk; repeated reads should be served from memory/page cache.
- No special storage-server configuration for the test.
- Goal: keep the production stack, reduce disk-layout variability.
- Layers still tested:
  - WAN / LHCONE / ESnet path
  - TCP buffers and congestion behavior
  - WebDAV/gfal behavior
  - dCache doors/pools
  - host and container networking

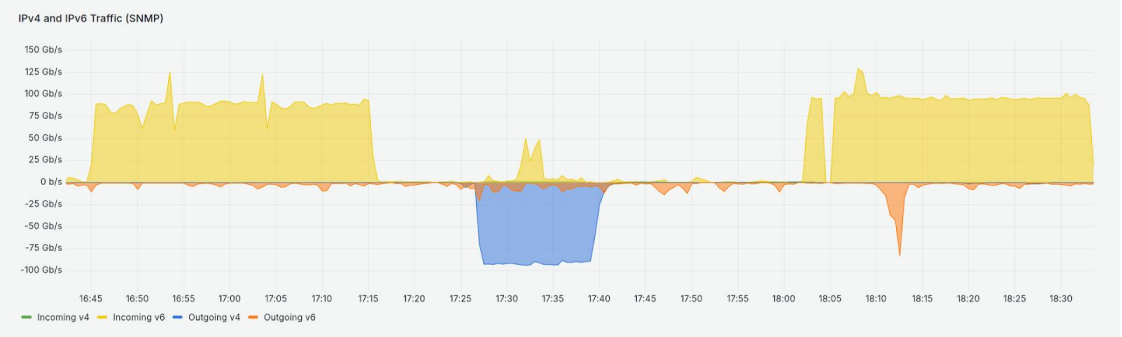
- **PRG dCache storage nodes → NET2 OKD client pods → /dev/null**
- **NET2 dCache storage nodes → PRG storage nodes → /dev/null**



Debugging transatlantic traffic-engineering corner case

Problem observed

- During the very first test, PRG ↔ NET2 appeared capped near **100 Gb/s**.
- This was surprising because:
  - NET2 had demonstrated much higher throughput with other sites.
  - PRG had demonstrated much higher throughput with CERN.
  - The test was cache-oriented, so we did not initially suspect the disk layer.



ESnet diagnosis

- ESnet identified an **a–z pair corner case** for the Boston–Amsterdam path.
- Traffic was not hitting the intended traffic-engineering/load-balancing rule.
- It fell back to a shortest path involving a **100 Gb/s link**.

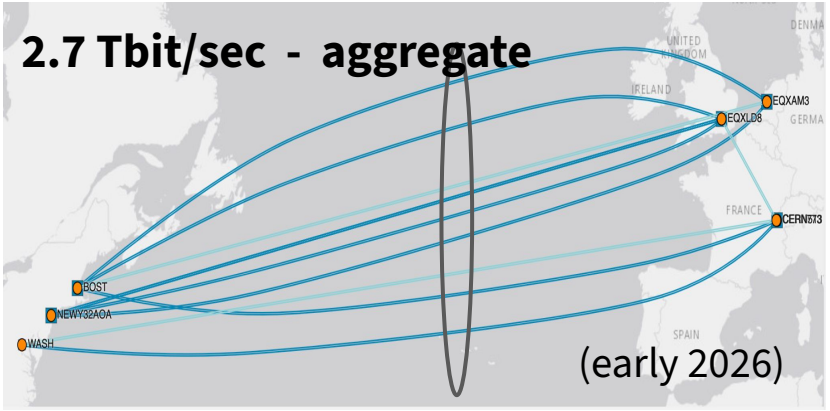
ESnet Transatlantic Traffic Engineering

NET2 traffic was taking the shortest path…

- Which is only 1 path and not the "best path"
- The direct Boston ⇄ Amsterdam (QXAM3 ⇄ BOST) link is only 100G

Solution: load-balance over multiple, unequal paths

- ESnet runs the transatlantic infrastructure on both sides and has full control
- Built virtual tunnels (label switched paths) across available links
  - Further protect LHCOPN traffic by avoiding their paths
- Assign weights to each tunnel, load balance traffic across the tunnels
  - Aggregate capacity Boston ⇄ Amsterdam now closer to ~2 Tbit



This is exactly the kind of issue that targeted pre-HL-LHC tests should expose early.

PRG → NET2 after the fix: 400 Gb/s reachable, but stream count dominates

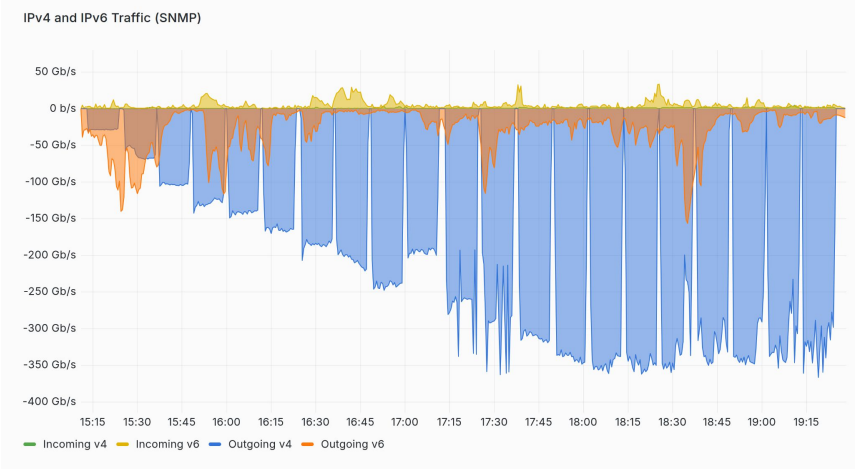
With “normal” transfer counts, throughput was much lower than expected.

Increasing the number of parallel transfer loops increased aggregate throughput.

Run 1 reached the 400 Gb/s class only around ~3k logical transfers.

Adding more transfers produced diminishing returns / noisy saturation.

Suspicion landed over long distance tuning:
PRG ↔ NET2 RTT: ~90–94 ms



First-order model: bandwidth-delay product

$$\mathrm{BDP} = \text{bandwidth} \times \mathrm{RTT}$$
$$\mathrm{BDP} = 400\,\mathrm{Gb/s} \times 0.09\,\mathrm{s} = 36\,\mathrm{Gb}$$
$$36\,\mathrm{Gb} \div 8 = 4.5\,\mathrm{GB}$$

$$\text{window per flow} \approx \frac{400\,\mathrm{Gb/s} \times 0.09\,\mathrm{s}}{N}$$

For PRG → NET2 at 400 Gb/s and 90 ms:

Parallel transfers	Approx. in-flight data needed per transfer
50	~90 MB
100	~45 MB
200	~22.5 MB
500	~9 MB
1000	~4.5 MB
3000	~1.5 MB

$$\text{throughput per flow} \approx \frac{\text{window}}{\mathrm{RTT}}$$

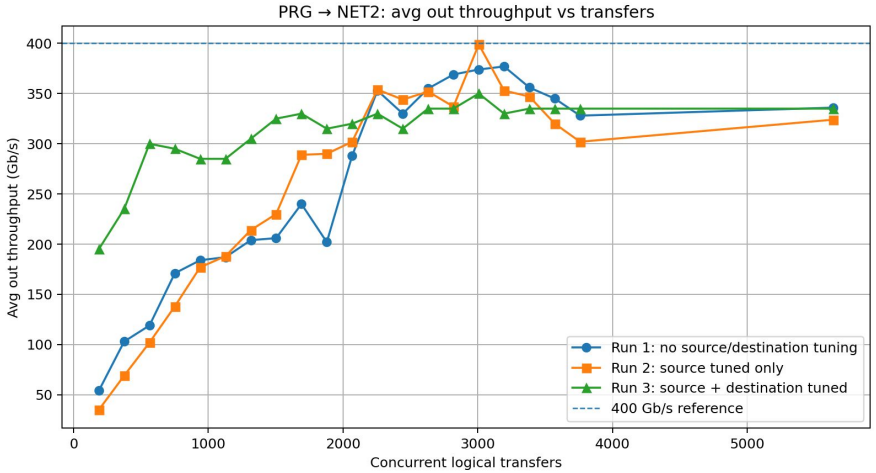
Window	Approx. upper-bound per-flow throughput at 90 ms
4 MiB	~0.37 Gb/s
6 MiB	~0.56 Gb/s
64 MiB	~6.0 Gb/s
128 MiB	~12.0 Gb/s

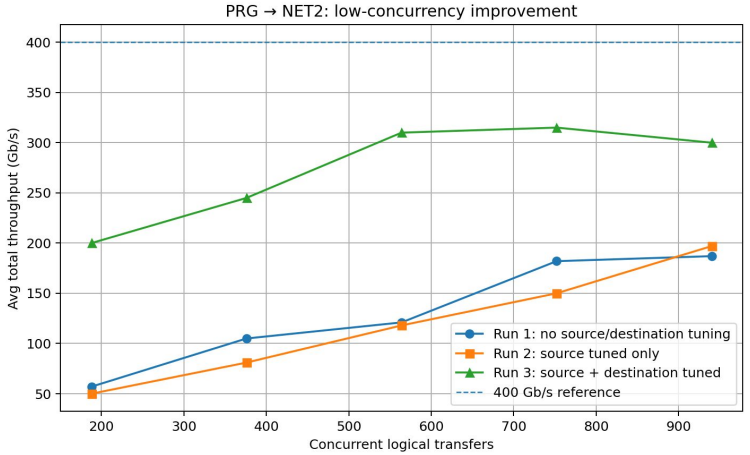
And that matches our runs:

Linux TCP receive auto-tuning can grow the receive buffer for each flow, but no greater than **net.ipv4.tcp_rmem**. **tcp_wmem** similarly bounds send-buffer auto-tuning. **net.core.rmem_max/wmem_max** are broader system socket-buffer ceilings.

Transfers	No src/dest tuning
188	57 Gb/s
376	105 Gb/s
564	121 Gb/s

Is tuning one end enough? PRG → NET2 sequence





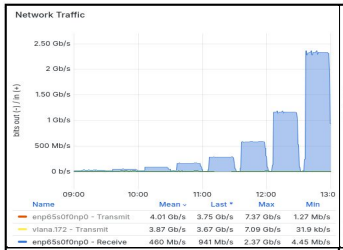
Target using avg total	No src/dest tuning	Source tuned only	src+dest tuned
50% of 400 Gb/s	1,316 transfers, 212 Gb/s	1,316 transfers, 214 Gb/s	188 transfers, 195 Gb/s
80% of 400 Gb/s	2,256 transfers, 357 Gb/s	2,068 transfers, 302 Gb/s	564 transfers, 310 Gb/s
90% of 400 Gb/s	2,820 transfers, 373 Gb/s	2,256 transfers, 354 Gb/s	3000 transfers, 350 Gb/s
95% of 400 Gb/s	3,008 transfers, 380 Gb/s	2,632 transfers, 384 Gb/s	NA
100% of 400 Gb/s	Not reached	3,008 transfers, 397 Gb/s	NA

NET2 destination tuning was not exotic:

```
spec:
  template:
    spec:
      securityContext:
        sysctls:
        - name: net.ipv4.tcp_rmem
          value: "4096 87380 67108864"
        - name: net.ipv4.tcp_wmem
          value: "4096 87380 67108864"
```

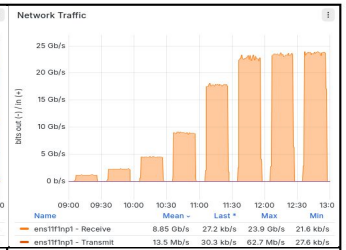
Run 3 reached >300 Gb/s with only a few hundred transfers.

NET2 → PRG: PRG-side tuning and final stable run



No tuning / EL10 default (nfs27), "impressively" slow transfers…

CUBIC + net mem tuning (nfs26), slightly slower and not so stable as BBR



BBR + net mem tuning (nfs25) works best for this RTT 114ms link

Avg throughput details for different number of concurrent downloads from NET2 to PRG (MTU 9000):

# conn	No tuning / EL10.1 def [Gbps]	CUBIC + net mem [Gbps]	BBR + net mem [Gbps]
1	0.03	1.4	1.1
2	0.05	2.3	2.2
4	0.09	4.3	4.4
8	0.16	8.2	8.9
16	0.29	15.3	17.7
32	0.59	20.6	22.8
64	1.16	21.7	23.4
128	2.32	22.7	23.7

WLCG network monitoring here.

NET2 storage tuning (present all the time)

net.ipv4.tcp_rmem = 4096 131072 1073741824
net.ipv4.tcp_wmem = 4096 16384 1073741824
net.core.rmem_max = 2147483647
net.core.wmem_max = 2147483647
net.ipv4.tcp_mtu_probing = 1
net.ipv4.tcp_congestion_control = cubic

Prague net mem tuning (not present only for the stock-config test)

net.core.rmem_max = 134217728
net.core.wmem_max = 134217728
net.core.rmem_default = 212992
net.core.wmem_default = 212992
net.ipv4.tcp_rmem = 4096 87380 67108864
net.ipv4.tcp_wmem = 4096 87380 67108864
net.core.default_qdisc = fq_codel
net.ipv4.tcp_congestion_control = cubic

Current investigation: residual per-flow degradation after TCP tuning

Leading current hypothesis

- Short bursts overflow switch queues even when long-term average is below link capacity.
- Switch discards cause TCP retransmissions.
- Long RTT amplifies the cost of loss.

But, Petr saw under similar WAN/source conditions, the directly connected 400 Gb/s host scaled much more linearly and had no local packet drops, while the 25 Gb/s server path showed sublinear scaling, switch discards, and TCP retransmissions.

Other observations:

- Discards on the 200 Gb/s trunk to the ToR;
- Discards on the ToR port to the 25 Gb/s server;
- TCP retransmissions on the 25 Gb/s server;
- No comparable retransmissions on the 400 Gb/s ps400 host.

Transfers	Run 1: no tuning	Run 2: src tuned	Run 3: src+dest tuned
188	0.303 Gb/s/transfer	0.266 Gb/s/transfer	1.064 Gb/s/transfer
564	0.215 Gb/s/transfer	0.209 Gb/s/transfer	0.550 Gb/s/transfer
1,316	0.161 Gb/s/transfer	0.176 Gb/s/transfer	0.243 Gb/s/transfer
2,256	0.158 Gb/s/transfer	0.171 Gb/s/transfer	0.148 Gb/s/transfer
3,008	0.126 Gb/s/transfer	0.139 Gb/s/transfer	0.121 Gb/s/transfer
5,640	0.060 Gb/s/transfer	0.060 Gb/s/transfer	0.064 Gb/s/transfer

Destination	Path	1	2	4	8	16	32
ps400	400G direct to core	1.1	2.2	4.4	8.9	17.8	35.5G/s
25G server	core → 200G → ToR → 25G server	0.9	1.7	2.8	4.6	6.6	9.7 Gb/s

That pattern is consistent with microburst loss. Even if the one-minute average is below 25 Gb/s, short bursts can temporarily exceed the egress capacity or the switch buffer capacity. If the output queue overflows, the switch drops packets. On a 90–114 ms path, even small packet loss can hurt a TCP flow badly because congestion control backs off and recovery takes at least one RTT.

Hiro’s observation is real, but the test design suggests it is not necessarily the disk layer.

Next steps: finish the tuned run, then move back toward disk-to-disk

These tests reached the 400 Gb/s class in both directions, but the path exposed several hidden layers: ESnet traffic engineering, high-BDP TCP behavior, endpoint socket-buffer tuning, test distribution, and possibly host/container networking. The cache-oriented test was useful because it reduced disk variability without bypassing the real production stack.

Possible subjects for future investigation

- Collect high-resolution switch discard/queue telemetry; LibreNMS RRD resolution is too coarse.
- Capture per-host TCP retransmits, out-of-order queue, cwnd/rwnd, and RTT during each pass.
- Test paced startup vs burst startup to distinguish steady-state capacity from microburst behavior.

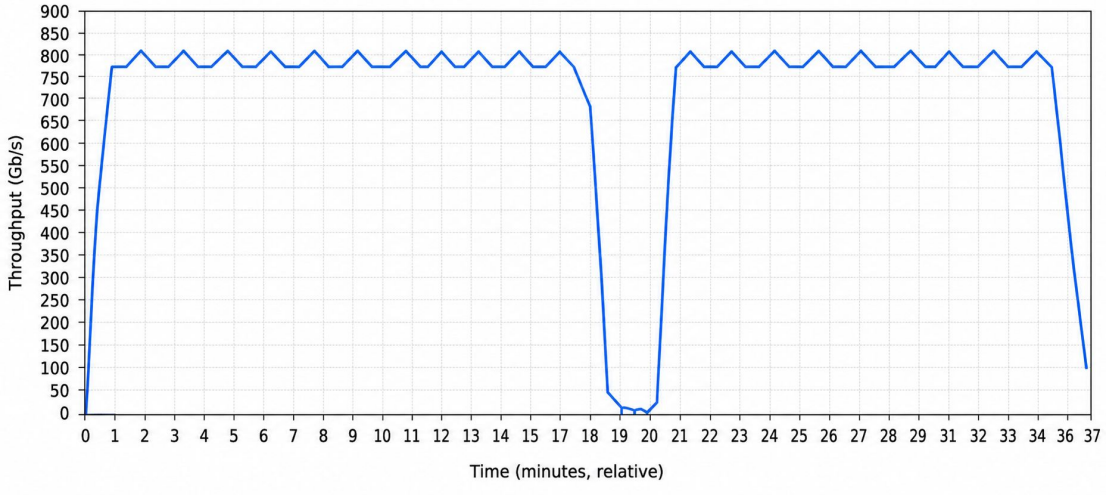
Then move to disk-to-disk tests:

- Now that network and client tuning is better understood.

Thank you

Backup

SC25 Preparation result: redirection mode + encryption @800Gbps



- 800Gbps connecting the Cluster (150 nodes) and dCache cluster
- One door - redirection mode
  - `webdav.limits.acceptors=10`
  - `webdav.limits.threads.max=2000`
  - `webdav.limits.backlog=20000`
  - `webdav.limits.idle-time=900`
  - `webdav.limits.queue-length=2000`
  - `webdav.limits.threads.min=33`
  - `webdav.mover.timeout=28800000`
  - `Tcp tuning`
- 35 disk servers [27 zfs (tuned) + 8 hardware raid (only read ahead)]

Encryption performance comparison
(webdav.redirect.allow-https=false/true)
Right - true
Left - false

- Transfers between computing cluster and dCache cluster (same datacenter)

NET2 > PRG result - WLCG network monitoring