One Page, Two DMAs: Writing 400 GbE TCP Streams to NVMe Without Touching the Data

Shingo Hattori
shattori@hpcs.cs.tsukuba.ac.jp
University of Tsukuba
Tsukuba, Ibaraki, Japan

Osamu Tatebe (Advisor)
tatebe@cs.tsukuba.ac.jp
University of Tsukuba
Tsukuba, Ibaraki, Japan

Abstract

Data Transfer Nodes write wide-area TCP streams to storage. On a 400GbE testbed, the buffered-write path delivers only 138Gbps on a server whose SSDs sustain 453Gbps: cores saturate copying, dirty-page throttling stalls the application, and stalls leak into TCP as window collapse and spurious retransmissions. We present zcp_ingest, which establishes an end-to-end alignment invariant that lets a NIC RX page serve unmodified as a storage write buffer, using only mainline mechanisms. Header-data split plus one TCP_MAXSEG option makes virtually all payload bytes arrive page-aligned; TCP_ZEROCOPY_RECEIVE maps NIC-written pages into userspace; io_uring writes them to NVMe via O_DIRECT - virtually all payload bytes reach storage without CPU copying. With deterministic sender-side placement (SRAP), sixteen flows sustain 373.4Gbps - 2.7x the buffered baseline and 97.4% of the path ceiling - with bit-exact terabyte-scale transfers and 4.4x lower CPU cost. Controlled violations establish one-flow-per-queue placement as a prerequisite for fair, drop-free, reproducible operation.

1 Problem

Data Transfer Nodes (DTNs) [4] write wide-area TCP streams to storage. On a 400 GbE pair (Xeon Gold 6530, ConnectX-7, eight NVMe SSDs, JBOD) the buffered-write path delivers 138.0 Gbps on a box whose SSDs sustain 453 Gbps (Fig. 2). Buffered ingest never reflects the box. Receiver cores saturate copying [2] (sys 76–79%, idle 6%), dirty pages pin at the 24.6 GB writeback threshold, and the sender is throttled, receive-window-limited up to 93.8%, the RTO (retransmission timeout) inflated from its 201 ms floor to 204–208 ms, with DSACK-confirmed spurious retransmissions in 864 of 928 samples, despite zero wire loss. One flow dies by window collapse (56 KB receive space), sixteen by timer inflation.

Testbed RTT is 0.37 ms, MTU 9000. Senders run CUBIC with socket buffers capped at 2 GiB. The pathology is receiver-side. Writeback stalls the application, which closes the window and inflates the RTO. WAN-RTT behavior, where congestion control and pacing enter the picture, is evaluated on the final poster.

2 Design

Our contribution is that alignment invariant — a NIC RX page serving, unmodified, as an NVMe write buffer — not any single mechanism: (i) on the wire, payload is an integer number of pages (TCP_MAXSEG, the advertised maximum segment size, plus hardware header split); (ii) in the stream, the write offset keeps 4 KiB phase (decoupled cursors with a flush discipline); (iii) in space, flow = queue = core = SSD with no sharing. We verify each by controlled violation: no header split → 0% zero-copy, phase break → 0.04% direct, collision depth 4 → starvation. NVMe-oF over RDMA attains

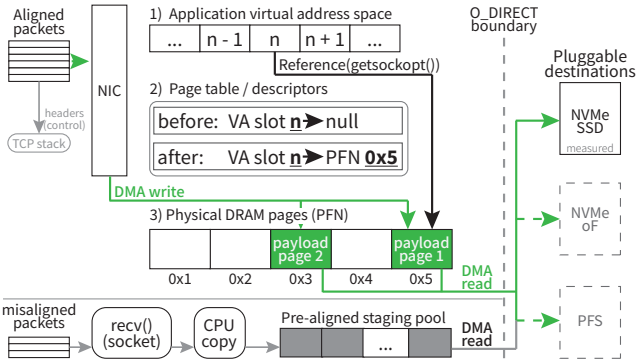


Figure 1: Two receive paths, one write gate. Aligned payloads cross as NIC-written pages, and getsockopt() rewires the page table (null → PFN) instead of copying. Misaligned residue takes one copy into a pre-aligned staging pool. All writes cross a single O_DIRECT boundary as 4 KiB pages. The sink behind it is pluggable (NVMe measured).

the same CPU-bypass property, and RoCE can ride routed overlays (VXLAN/EVPN). But reachability is not transport. RoCE still presumes fabric-coordinated loss control (PFC/DCQCN) and datacenter latencies, and go-back-N recovery degrades on lossy long-RTT paths, so cross-domain DTN transfers remain plain TCP [4]. Our path is receiver-local and works with any unmodified TCP sender.

Alignment comes from header-data split, not MTU numerology. One socket option (TCP_MAXSEG = 4108, via SYN-ACK) plus hardware GRO (mlx5 SHAMPO) gives 100.00% zero-copy reception at standard MTU 9000. TCP_ZEROCOPY_RECEIVE [3] maps NIC-written pages into userspace. Page-aligned by construction, they satisfy O_DIRECT (direct I/O, bypassing the page cache) verbatim, so io_uring submits them asynchronously, untouched. The zero-copy datapath exposes receive-queue placement, condition (iii), as a functional prerequisite. SRAP, our prior work [6], supplies it. An LD_PRELOAD shim selects source ports at connect(), two flows per SSD (Fig. 1). Phase, like placement, is a sender-side discipline, and 4 KiB-multiple writes prevent slip (nine events in 25 TiB). Feeding the residual back would let the sender re-cut one segment (not padding) and recover within an RTT. Staging remains the bounded fallback.

3 Results

The zero-copy path sustains 373.4 Gbps, 2.705× the buffered baseline and 97.4% of the transfer-engine ceiling (383.4 Gbps, writes removed). Fig. 2 splits that factor. Removing the page cache buys +56.8 Gbps (192.9, every byte still copied), removing that copy a

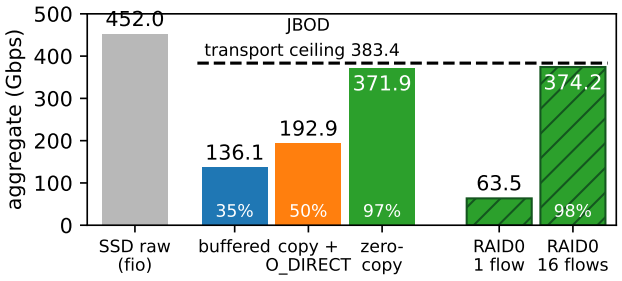


Figure 2: What the write path costs, then what the layout costs (8×CM7-V, freshly formatted, n=3 unless noted). Left, JBOD at 16 aligned flows. Zero-copy reaches 97% of the transport ceiling (dashed: same binary, writes removed), buffered write 30% of what the SSDs absorb. Right, the same zero-copy path on a RAID0 stripe. One flow sustains 63.5 Gbps and eight fill the link (312.4, not plotted), so few-flow senders need no stream fan-out. Sixteen only match JBOD (374.2, n=5, against 371.9), which keeps the path pure.

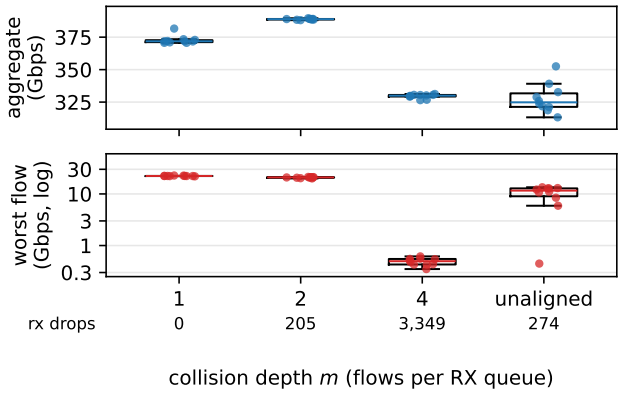


Figure 3: Aggregate hides placement failure (m = flows per RX queue, n=10 per arm). At m=4 the median falls 11.3% while the worst flow collapses 45×, and only unaligned placement scatters.

further +179.0. Neither cure suffices alone. Copying saturates all sixteen cores even with writes removed, while zero-copy needs ten, 22% idle. CPU cost falls from 120.7 to 27.3 millicores per Gbps (4.4×). Layout follows flow count (Fig. 2). Striping wins for few-flow senders, retiring the many-streams reflex, while at sixteen aligned flows our JBOD layout matches it and keeps the path pure. RAID0 CPU-copies 2.7% of all bytes and halts outright at 1 MiB chunks. Nor is this a sprint: 62.5 TiB over 23 minutes (drift −0.8%), a 7.8 TiB dataset in 189 s versus 498 s buffered. Integrity is bit-exact: for 100 GiB, sender, in-flight and read-back SHA-256 match in all three conditions (fallback 0.00003%). The 8 TiB endurance image verified 64 of 64 windows, zero media errors.

4 Placement

Prior work [5, 6] showed reserved-rate delivery is governed by the hottest RSS queue. Random placement missed reservations in 9 of 10 trials at 95% load, deterministic placement in none of 50. We add controlled violations at the headline point, changing only placement (Fig. 3, ten 60 s trials per arm). Alignment buys predictability before speed. Unaligned placement loses 12.6% of median aggregate (324.8 against 371.8 Gbps) at 3.46% run-to-run variation versus 0.87%. Collision depth m = 4 starves a flow in every trial (0.35–0.62 Gbps, unfairness to 123×) while aggregate retains 88.7%. m=2 is in fact fastest (388.9 Gbps); a crossed-mapping control attributes this to receive-side batching (+13–17% bytes per zero-copy call), not write locality. Receive-queue drops lead (0 → 205 → 3,349 per trial), zero in all ten only at m=1. Under zero-copy, one flow per queue is the prerequisite for *fair, drop-free, reproducible* operation: aggregate masks starvation, and the mean masks the accident.

5 Future Work

We are deploying this design on kernel 6.17's io_uring zcrx generation [7], where alignment is native. It removes the per-page bookkeeping and declares flow steering out of scope, making deterministic sender-side placement a functional prerequisite. The final poster adds egress symmetry and the zcrx comparison.

Acknowledgments

This work is based on results obtained from the project, "Research and Development Project of the Enhanced infrastructures for Post-5G Information and Communication Systems" (JPNP20017), commissioned by the New Energy and Industrial Technology Development Organization (NEDO), and was supported by JSPS KAKENHI Grant Number JP22H00509, JST CREST JPMJCR24R4, the Multidisciplinary Cooperative Research Program in CCS, University of Tsukuba, and Fujitsu Limited.

The author used Claude (Anthropic) [1] as a writing assistant; all content is the author's responsibility.

References

[1] Anthropic. [n. d.]. Claude. Accessed August 2026. https://claude.ai

[2] Qizhe Cai, Shubham Chaudhary, Midhul Vuppalapati, Jaehyun Hwang, and Rachit Agarwal. 2021. Understanding host network stack overheads. In *Proceedings of the 2021 ACM SIGCOMM 2021 Conference* (Virtual Event, USA) *(SIGCOMM '21)*. Association for Computing Machinery, New York, NY, USA, 65–77. doi:10.1145/3452296.3472888

[3] Jonathan Corbet. 2018. Zero-copy TCP receive. LWN.net. https://lwn.net/Articles/752188/

[4] Eli Dart, Lauren Rotman, Brian Tierney, Mary Hester, and Jason Zurawski. 2013. The Science DMZ: a network design pattern for data-intensive science. In *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis* (Denver, Colorado) *(SC '13)*. Association for Computing Machinery, New York, NY, USA, Article 85, 10 pages. doi:10.1145/2503210.2503245

[5] Nathan Hanford, Brian Tierney, and Dipak Ghosal. 2015. Optimizing data transfer nodes using packet pacing. In *Proceedings of the Second Workshop on Innovating the Network for Data-Intensive Science* (Austin, Texas) *(INDIS '15)*. Association for Computing Machinery, New York, NY, USA, Article 4, 8 pages. doi:10.1145/2830318.2830322

[6] Shingo Hattori and Osamu Tatebe. 2026. SRAP: Sender-side Receiver-Aware Port Selection for Deterministic Flow-to-Core Mapping in Hash-Based Flow Steering. In *Proceedings of the 35th International Symposium on High-Performance Parallel and Distributed Computing (HPDC '26)*. Association for Computing Machinery, New York, NY, USA, 549–551. doi:10.1145/3806645.3820062

[7] The kernel development community. [n. d.]. io_uring zero copy Rx. The Linux Kernel documentation. Accessed August 2026. https://docs.kernel.org/networking/iou-zcrx.html