TECHNICAL DOSSIER // HIGH-PERFORMANCE SYSTEMS & LINUX KERNEL

Linux io_uring Zero-Copy Socket Transmission Internals

Asynchronous Ring Architectures, Page Pinning Lifecycles, and 100GbE Performance

Published: October 30, 2026 | Distribution: Open Engineering Community | Document ID: LT-2026-B16-05

EXECUTIVE SUMMARY & MICROARCHITECTURAL SCOPE

Traditional Linux network transmission requires copying packet payloads from userspace buffers into kernel-allocated socket sk_buff data structures, saturating CPU memory bandwidth and polluting CPU L1/L2/L3 cache hierarchies at 100GbE+ line rates. The Linux io_uring subsystem introduces IORING_OP_SEND_ZC and IORING_OP_SENDMSG_ZC, providing a fully asynchronous, zero-copy packet transmission pipeline that pins userspace pages directly into NIC DMA descriptor rings.

Metric	Standard send()	sendmmsg() (Batched)	MSG_ZEROCOPY (TCP)	io_uring SEND_ZC
Kernel Interface	Synchronous Syscall	Synchronous Syscall	Synchronous + Poll Err	Asynchronous Ring (SQ/CQ)
Memory Copies	1 (User -> Kernel)	1 (User -> Kernel)	0 (Page Pinning)	0 (Zero Copy + Fixed Reg)
Notification Path	Return Value	Return Value	recvmsg(MSG_ERRQUEUE)	CQE32 Completion Queue
100GbE Core Saturation	2.8 Cores	2.1 Cores	1.4 Cores	0.65 Cores
P99 Packet Latency	48 us	38 us	29 us	14 us

1. The Memory Copy Bottleneck at 100GbE

At 100 Gigabit Ethernet line rates, a server must transmit over 14 million packets per second. Conventional socket transmission incurs a mandatory memcpy() from application heap buffers into kernel sk_buff structures. This memory copy saturates memory bus bandwidth (consuming over 25 GB/s of RAM throughput) and causes severe cache line eviction across the CPU core.

2. Architecture of IORING_OP_SEND_ZC

io_uring zero-copy operates via shared memory submission (SQ) and completion (CQ) rings. When an application posts an IORING_OP_SEND_ZC request, the kernel pins the backing physical memory pages using get_user_pages_fast() and attaches them to the skb frag_list. The network interface card (NIC) DMA controller fetches packet bytes directly from application memory without CPU intermediate copying.

3. Notification Lifecycles and CQE32 Completion

Zero-copy transmission introduces an asynchronous page reuse constraint: user memory must not be overwritten until the NIC finishes transmitting the DMA frame. Unlike legacy MSG_ZEROCOPY which required expensive recvmsg() error-queue polling, io_uring returns a dedicated notification Completion Queue Entry (CQE) with the IORING_CQE_F_MORE flag, signaling exact buffer release with near-zero latency.

4. Registered Buffers and Production Tuning

By combining SEND_ZC with pre-registered user buffers (IORING_REGISTER_BUFFERS), the kernel avoids per-packet virtual-to-physical address translation and page table walking overhead. Production benchmarks demonstrate that a single CPU core driving io_uring SEND_ZC can saturate a 100GbE link at 94.2 Gbps throughput with under 15 microsecond P99 latency.

STRATEGIC ENGINEERING TAKEAWAYS & DEPLOYMENT CHECKLIST

- **True Zero-Copy Pipeline:** Eliminates memcpy overhead, streaming data straight from userspace heap to NIC DMA rings.
- **Ring-Based Notifications:** CQE completion entries eliminate complex error-queue syscall polling.
- **Fixed Buffer Pre-Registration:** Eliminates repeated page pinning (get_user_pages_fast) and TLB invalidation overhead.
- **Extreme CPU Efficiency:** Achieves 100GbE line-rate transmission using under 70% of a single server core.