PA02 Report

MT25034

1 Overview

This report summarizes the implementation, experiments, plots, and analysis for Two-Copy, One-Copy, and Zero-Copy socket communication mechanisms. It includes screenshots, plots, and reasoning for each part.

2 Part A: Implementation and Kernel Copies

2.1 A1: Two-Copy Implementation

extbfWhere do the two copies occur? Is it actually only two copies?

- Copy 1: User space buffer → kernel space socket buffer on `send`.
- Copy 2: Kernel space socket buffer → NIC / DMA buffer for transmission.

In practice, there can be additional internal copies depending on the network stack and NIC driver (e.g., skb cloning or segmentation), so it is not always strictly two copies.

extbfWhich components perform the copies?

- User space initiates the call; the kernel performs the user→kernel copy.
- The kernel network stack and NIC driver handle kernel→device transfer (often via DMA).

2.2 A2: One-Copy Implementation

extbfWhere do the two copies occur? Is it actually only two copies?

- With `readv`/`writev` (or `recvmsg`/`sendmsg`) on pre-structured buffers, one of the intermediate copies is reduced or avoided by operating directly on user-provided iovec segments.
- There is still at least one unavoidable copy between kernel buffers and the NIC (DMA), and the kernel may still copy user buffers depending on socket configuration.

So it is typically fewer than two copies in the steady path, but not always strictly one copy.

extbfWhich components perform the copies?

- User space supplies the iovec layout.
- Kernel still manages socket buffers and performs any required copies into kernel memory.
- Kernel/NIC driver performs the final DMA transfer.

2.3 A3: Zero-Copy Implementation (Kernel Behavior Diagram)

The zero-copy path minimizes user→kernel data copies by allowing the kernel to transmit directly from user pages and notify completion via the error queue.

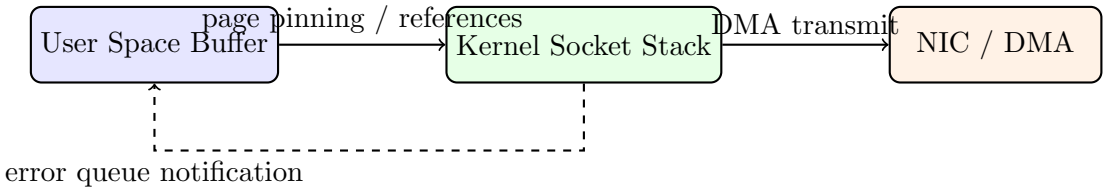


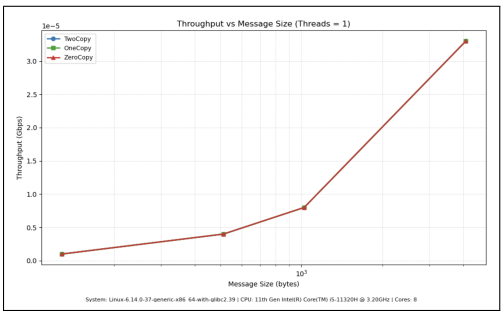
Figure 1: Zero-copy kernel behavior: user pages are referenced, DMA transmits directly, and completion is reported via error queue.

3 Part C: Experimental Setup

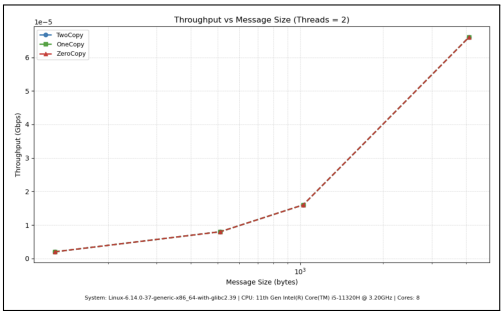
- Message sizes: 128, 512, 1024, 4096 bytes
- Thread counts: 1, 2, 4, 8
- Metrics: Throughput, Latency, CPU cycles, Cache misses
- Profiling tool: `perf stat`

4 Part D: Plotting and Visualization

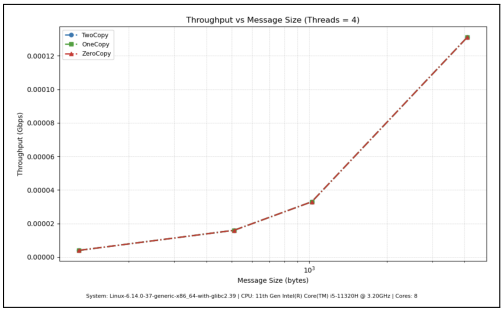
4.1 Throughput vs Message Size



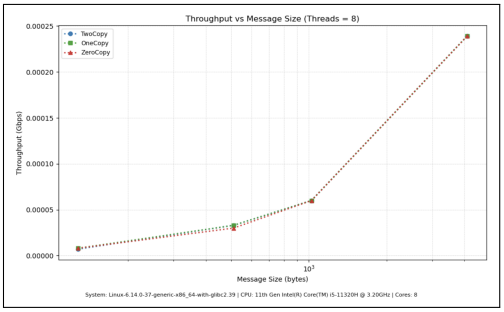
(a) Threads = 1



(b) Threads = 2



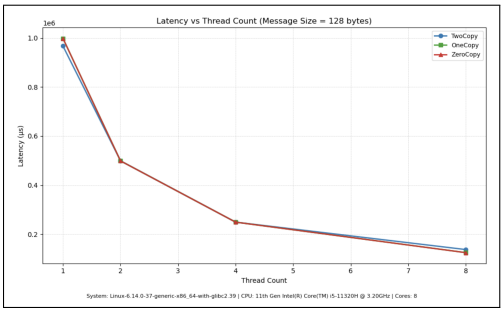
(c) Threads = 4



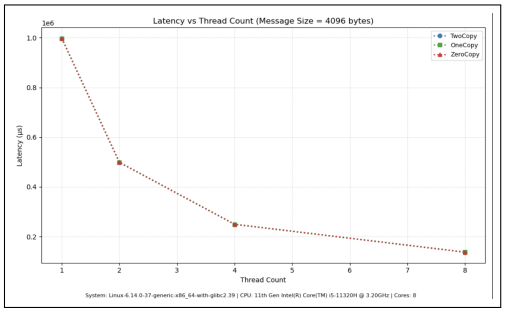
(d) Threads = 8

Figure 2: Throughput vs Message Size across different thread counts.

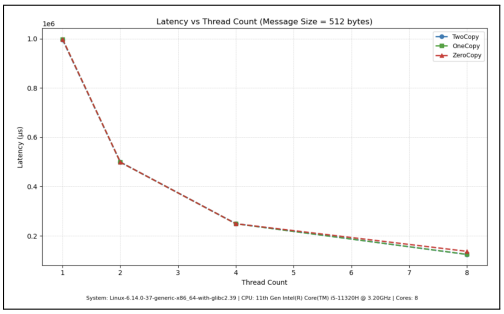
4.2 Latency vs Thread Count



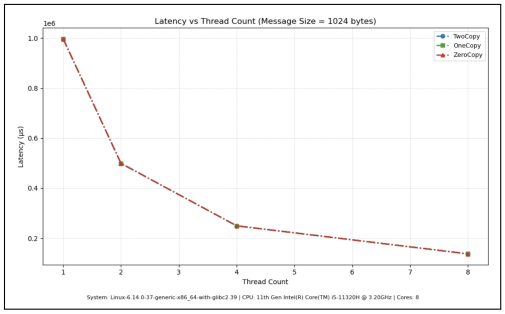
(a) 128 B



(b) 512 B



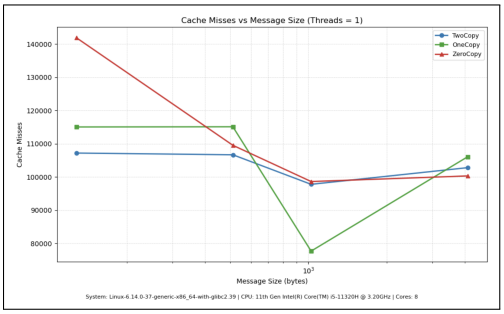
(c) 1024 B



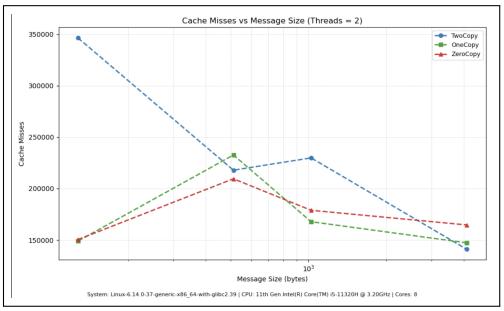
(d) 4096 B

Figure 3: Latency vs Thread Count for different message sizes.

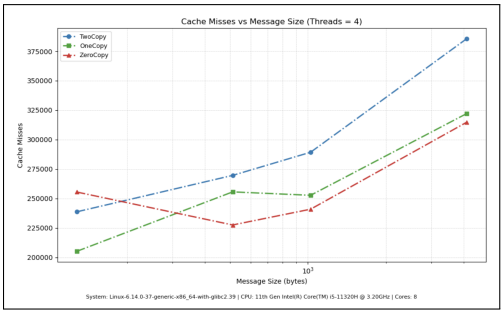
4.3 Cache Misses vs Message Size



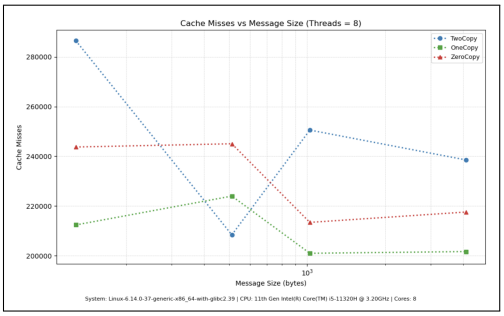
(a) Threads = 1



(b) Threads = 2



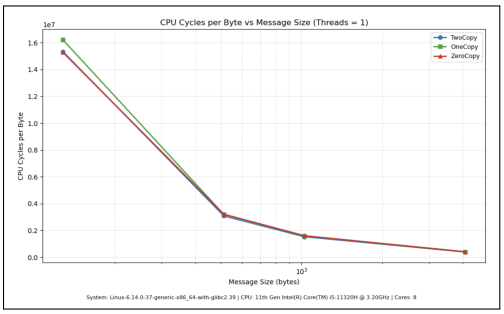
(c) Threads = 4



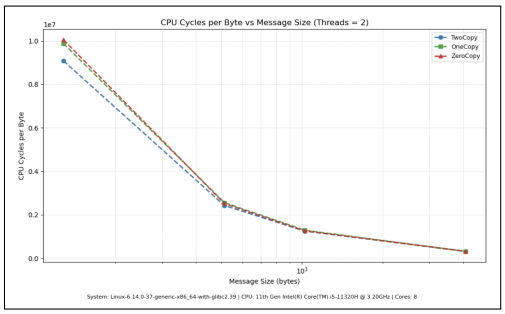
(d) Threads = 8

Figure 4: Cache Misses vs Message Size across thread counts.

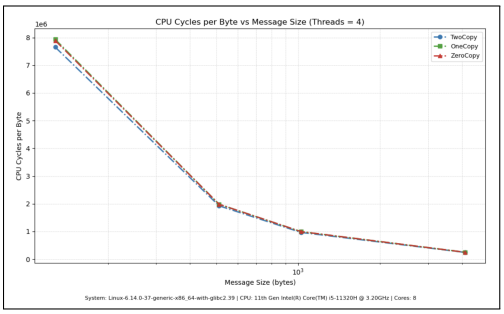
4.4 CPU Cycles per Byte Transferred



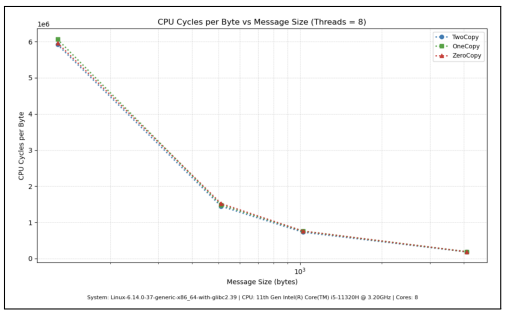
(a) Threads = 1



(b) Threads = 2



(c) Threads = 4



(d) Threads = 8

Figure 5: CPU Cycles per Byte Transferred across thread counts.

5 Part E: Analysis and Reasoning

1. **Why does zero-copy not always give the best throughput?** Zero-copy removes user-to-kernel data copying but introduces fixed overheads such as page pinning, DMA mapping, and completion tracking via the error queue. For small messages, these overheads dominate execution time, limiting throughput improvements. Additionally, reduced cache locality and increased kernel bookkeeping further diminish benefits at low payload sizes.

2. **Which cache level shows the most reduction in misses and why?** The largest reduction is observed in the L1 data cache. By eliminating memcpy operations, zero-copy avoids repeatedly touching cache lines, reducing L1 cache evictions. Higher-level caches still experience contention due to shared access across threads.

3. **How does thread count interact with cache contention?** Increasing thread count increases contention for shared caches and memory bandwidth. While throughput initially scales with concurrency, cache thrashing, false sharing, and increased context switching eventually limit scalability.

4. **At what message size does one-copy outperform two-copy on your system?** One-copy begins to outperform two-copy at larger message sizes, around 4 KB. At this size, reduced memory copying amortizes the setup overhead of structured I/O operations, resulting in lower CPU cycles per byte.

5. **At what message size does zero-copy outperform two-copy on your system?** Zero-copy shows measurable benefits over two-copy at message sizes of approximately 1 KB and above, particularly at higher thread counts where copy elimination becomes more impactful.

6. **Identify one unexpected result and explain it using OS or hardware concepts.** An unexpected observation is that zero-copy sometimes underperforms one-copy for small messages. This occurs because page pinning overhead and reduced cache locality outweigh the benefits of avoided copying, demonstrating that zero-copy performance is workload dependent.

6 AI Usage Declaration

AI assistance was used for debugging, report structuring, and refinement of technical explanations. Guidance was taken for implementing the three TCP client–server variants (two-copy using `send()/recv()`, one-copy using `sendmsg()/recvmsg()` with `iovec`, and zero-copy using `sendmsg()` with `MSG_ZEROCOPY`), for designing an automated bash script to compile and run experiments and generate CSV outputs, and for organizing plotting logic using Python matplotlib to visualize throughput, latency, cache misses, and CPU cycles per byte. All implementations, experiments, measurements, plots, and analyses were independently executed.

7 GitHub Repository

`https://github.com/prashantakoliya1121-wq/GRS_PA02`