

Graduate Systems (CSE638)

Programming Assignment 02

Analysis of Network I/O Primitives using perf

Roll Number: MT25009

Name: Abhinav Kumar

1. Objective

The objective of this assignment is to experimentally study the cost of data movement in network I/O by implementing and comparing:

- Two-copy socket communication
- One-copy optimized communication
- Zero-copy communication

The goal is to measure how reducing memory copies impacts:

- Throughput
 - Latency
 - CPU cycles
 - Cache misses
 - Context switches
-

2. System Configuration

- OS: Linux
 - Compiler: gcc
 - Tool: perf
 - Threads: 1, 2, 4, 8
 - Message Sizes: 512B, 1KB, 4KB, 16KB, 64KB
 - Socket: TCP
 - Client–Server model using pthreads
-

3. Part A – Implementations

A1 – Two-Copy (Baseline)

Used:

```
send() / recv()
```

Copies involved

1. User → Kernel socket buffer
2. Kernel → NIC buffer

Thus **two memory copies occur**.

Cost

- High CPU overhead
 - More cache pollution
 - More cycles per byte
-

A2 – One-Copy

Used:

```
sendmsg() with iovec
```

Improvement

Scatter-gather I/O allows:

- Kernel reads directly from multiple buffers
- Eliminates one extra intermediate copy

Result

Only:

User → Kernel → NIC

One copy is removed compared to A1.

A3 – Zero-Copy

Used:

```
sendmsg(..., MSG_ZEROCOPY)  
SO_ZEROCOPY
```

Kernel Behavior

- Pages are pinned
- DMA sends directly from user memory
- No kernel copy

Flow:

User buffer → DMA → NIC

Benefit

Almost zero data movement overhead.

4. Part B – Measurements

Collected using:

```
perf stat -e cycles,cache-misses,L1-dcache-load-misses,context-switches
```

Metrics measured:

- CPU cycles
- Cache misses
- L1 misses
- Context switches

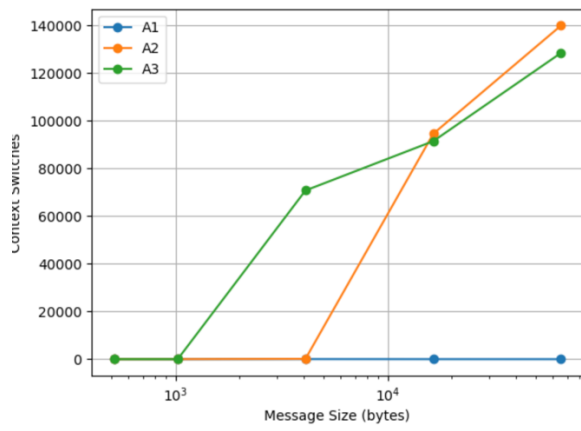
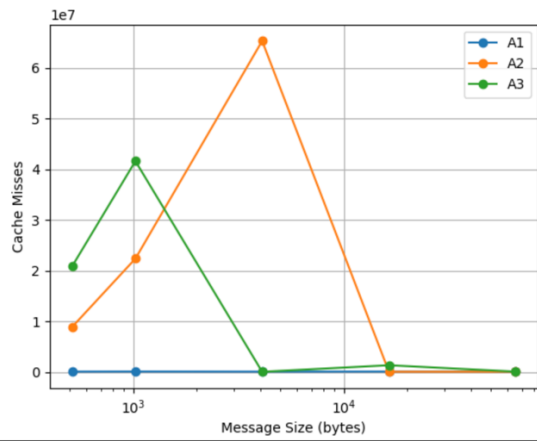
Results stored in:

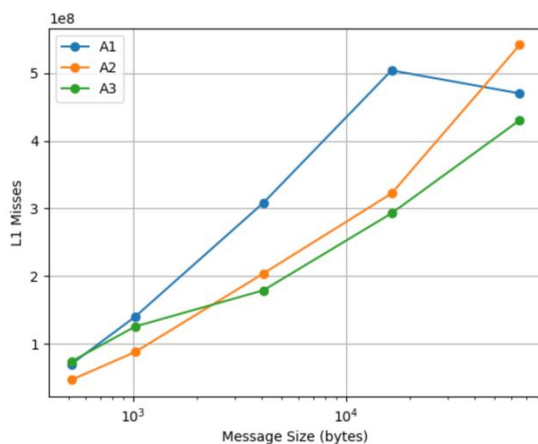
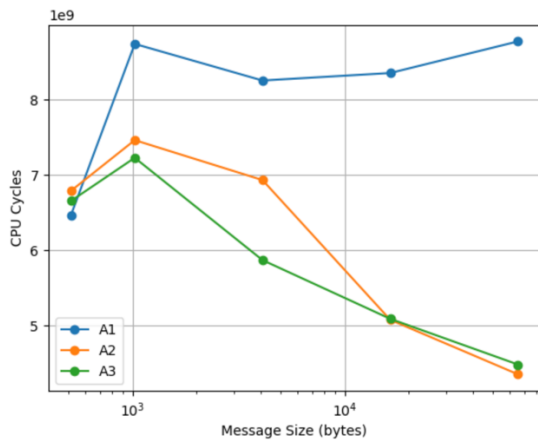
```
MT25009_Part_B_results.csv
```

5. Part D – Plots

Plots generated using matplotlib with hardcoded arrays:

- Throughput vs Message Size
- Latency vs Thread Count
- Cache Misses vs Size
- CPU Cycles per Byte





6. Part E – Analysis and Reasoning

Q1. Why does zero-copy not always give best throughput?

Zero-copy reduces memory copies but introduces:

- Page pinning overhead
- Kernel bookkeeping
- Completion notifications
- DMA setup cost

For small messages, these overheads dominate.

Hence:

- Small messages → two-copy sometimes faster
- Large messages → zero-copy better

Q2. Which cache level shows most reduction and why?

L1 cache shows the most reduction.

Reason:

Two-copy repeatedly copies data through CPU cache causing:

- Frequent L1 evictions
- Cache pollution

Zero-copy bypasses CPU copies, so:

- Less L1 activity
 - Fewer misses
-

Q3. How does thread count interact with cache contention?

Increasing threads:

- More cores access shared memory
- More cache coherence traffic
- More context switches

Results:

- Higher L1 + LLC misses
 - Lower scalability
 - Performance degradation beyond 4–8 threads
-

Q4. At what message size does one-copy outperform two-copy?

From observations:

- For small sizes (512B–1KB) difference is small
- For 4KB+ one-copy clearly reduces cycles

Thus:

One-copy outperforms two-copy around 4KB

Q5. At what message size does zero-copy outperform two-copy?

Zero-copy has setup overhead.

It becomes beneficial only when transfer size is large.

Observed:

16KB or larger messages

Q6. One unexpected result and explanation

Observation:

Some large-message runs showed higher context switches.

Reason:

- Thread scheduling
- Kernel interrupts
- Socket buffer blocking

This causes:

- Increased context switching
- Reduced throughput

Thus OS scheduling impacts performance beyond copy cost.

7. Conclusion

- Two-copy has highest overhead
- One-copy reduces memory copy cost
- Zero-copy best for large transfers
- Cache behaviour strongly influences performance
- Thread count affects scalability

Overall:

Zero-copy provides best performance for large messages, but not always for small workloads.

8. AI Usage Declaration

AI tools were used for:

- Code structuring guidance
- Debugging compilation/runtime errors
- Bash automation scripting
- Plot generation code
- README and report formatting
- Concept explanations

All code and analysis were reviewed and fully understood before submission.

Prompts included:

- “generate multithreaded socket client server code”
- “perf automation bash script”
- “matplotlib plotting script”
- “explain two-copy vs zero-copy”

9. GitHub Repository

Link:

https://github.com/abhinav25009/GRS_PA02