

CS 531 HW 3

Hritvik

1 Introduction

The goal is to understand and compare performance between:

- COO-format SpMV (parallel and serial)
- CSR-format SpMV (parallel and serial)

The input matrix (`cant.mtx`) is provided. Goal: converting COO to CSR while parallelizing the histogram and prefix sum stages, then implementing both serial and parallel SpMV kernels.

All timing results reflect 100 iterations of each SpMV kernel

2 Correctness Verification

The output vector was compared against the provided `ans.mtx` using:

```
diff myans.mtx cant/ans.mtx
```

All differences were on the order of 10^{-12} or smaller, well within the Thus, the implementation is numerically correct.

3 Performance Results

The following table summarizes the measured execution times:

Module	Time (seconds)
Load	0.663663
Convert	0.031062
Lock Init	0.000235
COO SpMV (parallel)	1.968822
CSR SpMV (parallel)	0.918589
COO SpMV (serial)	1.477310
CSR SpMV (serial)	1.129287
Store Output	0.023223

4 Performance Discussion

4.1 CSR Parallel vs CSR Serial

CSR parallel is the fastest implementation. It achieves a significant speedup over the serial CSR version (0.918 s vs. 1.129 s).

Reasons:

- Rows are naturally independent, so parallelization is trivial.
- CSR stores each row in a contiguous memory block, improving cache locality.
- No locks or synchronization are required.

4.2 COO Serial vs CSR Serial

CSR serial outperforms COO serial (1.129 s vs. 1.477 s). This is expected because COO has random memory access patterns and writes to arbitrary rows, while CSR accesses memory in long, contiguous runs that the CPU prefetcher can exploit.

4.3 COO Parallel vs COO Serial

Surprisingly, COO parallel is **slower** than COO serial (1.968 s vs. 1.477 s).

Reasons:

- Requires one lock per row to protect updates to the output vector.
- Lock contention is extremely high for sparse matrices with uneven row lengths.
- Atomic lock operations dominate the computation cost.
- Memory accesses are random and not cache-friendly.

Thus, COO is not a good format for parallel SpMV on CPUs.

4.4 Overall Performance Ranking

From fastest to slowest:

1. CSR Parallel (best)
2. CSR Serial
3. COO Serial
4. COO Parallel (worst)

CSR parallel is the only configuration that benefits substantially from multi-threading due to its regular memory layout and row independence.

5 Conclusion

- Data layout has a major impact on SpMV performance.
- CSR provides good spatial locality and is well-suited for row-level parallelism.
- COO performs poorly due to lock contention and scattered memory access.
- The parallel CSR kernel delivers the best performance.