Language Benchmark of Matrix Multiplication

https://github.com/klapa0/big_data_studies

Kacper Klasen FB390707

November 22, 2025

1 Sparse Matrix-Vector Multiplication Benchmark

Sparse matrix-vector multiplication (SPMV) is a fundamental operation in scientific computing, numerical simulations, and data analysis. Its performance depends heavily on the storage format of the sparse matrix and on memory access patterns, due to irregular accesses to the input vector.

In this experiment, we benchmarked two approaches:

1. **Naive CSR** — standard row-wise traversal using the Compressed Sparse Row format.
2. **Cache-blocked CSR (Bucketed/Compact)** — an attempt to improve cache locality by dividing columns into blocks and storing non-zero elements in contiguous memory per block.

1.1 Methodology

The `mc2depi.mtx` matrix from the Matrix Market repository was used as the test case. It has 525,825 rows and columns with 2,100,225 non-zero elements. The input vector x was filled with ones, and the SPMV operation was repeated 50 times to obtain stable timings.

The benchmark was compiled using:

```
g++ main.cpp -o spmv -std=c++17 -O3
```

where `-O3` enables aggressive compiler optimizations for speed.

The cache-blocked implementation used a block size of $CB = 4096$. Non-zero elements were reorganized into a compact bucketed structure before multiplication.

1.2 Results

Method	Average Time per Iteration [s]
Naive CSR	0.00371
Blocked CSR (CB=4096)	0.00580

Table 1: Average execution times for SPMV with naive and blocked CSR.

Both implementations produced identical results (maximum element-wise difference $< 10^{-9}$), confirming correctness.

1.3 Analysis

The cache-blocked version is slower than naive CSR for this matrix. The reason is that the matrix has relatively few non-zero elements per row, and the input vector x fits entirely in CPU caches. Consequently, naive CSR experiences very few cache misses (less than 5% of all cache references). Reorganizing the matrix into blocks introduces additional overhead without providing performance benefits.

Cache blocking is generally beneficial only for extremely large or highly sparse matrices where naive implementations suffer from frequent cache misses.

1.4 Conclusion

This experiment demonstrates that memory optimizations such as cache blocking are not universally beneficial. For matrices similar to `mc2depi.mtx`, naive CSR with compiler optimizations (`-O3`) can outperform more complex blocked implementations due to already low cache miss rates and good locality of the input vector.

References

- Matrix Market: https://math.nist.gov/MatrixMarket/
- ChatGPT, Assistance with LaTeX formatting and benchmark description. https://chat.openai.com/