Alejandro Lucena
2/27/2026

Experiment 3 Report: SpMM

Introduction and Methodology

This report evaluates the performance of two Sparse Matrix Multiplication (SpMM) implementations—“baseline” and warp-coalesced optimized (OPT)—across varying nnz (number of nonzero elements) densities to analyze the impact of coalesced memory access on throughput.

First, the platform details. All tests were conducted on an NVIDIA V100 GPU using CUDA version 12.4.1. Because GPU performance is highly dependent on hardware specifications, the GFLOP/s subsequently reported are specific to this GPU environment.

Next, a brief overview of both implementations. Both kernels implement the C = A x B operation using the Compressed Sparse Row (CSR) format. In the baseline implementation, each thread loops through N columns of matrix B independently. This results in uncoalesced memory accesses. The GPU has to issue many small, inefficient memory requests because neighboring threads are working on completely different rows of matrix B. In the warp-coalesced implementation, however, threads in a warp access adjacent columns “j” of matrix B at the same time. This, in turn, enables memory coalescing, where the hardware merges 32 simultaneous 4-byte fetches into a single 128-byte memory transaction. This maximizes effective bandwidth and reduces the total number of requests sent to the memory controller. Both implementations operate on Compressed Sparse Row (CSR) sparse matrices, the most cache-efficient method for storing sparse matrices.

Finally, the performance metric. To quantify and compare throughput, this report utilizes GFLOP/s as its sole performance metric. For a standard SpMM operation, the operation count is 2*nnz*N. Performance is thus calculated as: GFLOP/s = 2*nnz*N / Time (s) * 10^9.

Performance Analysis: Throughput and Scaling

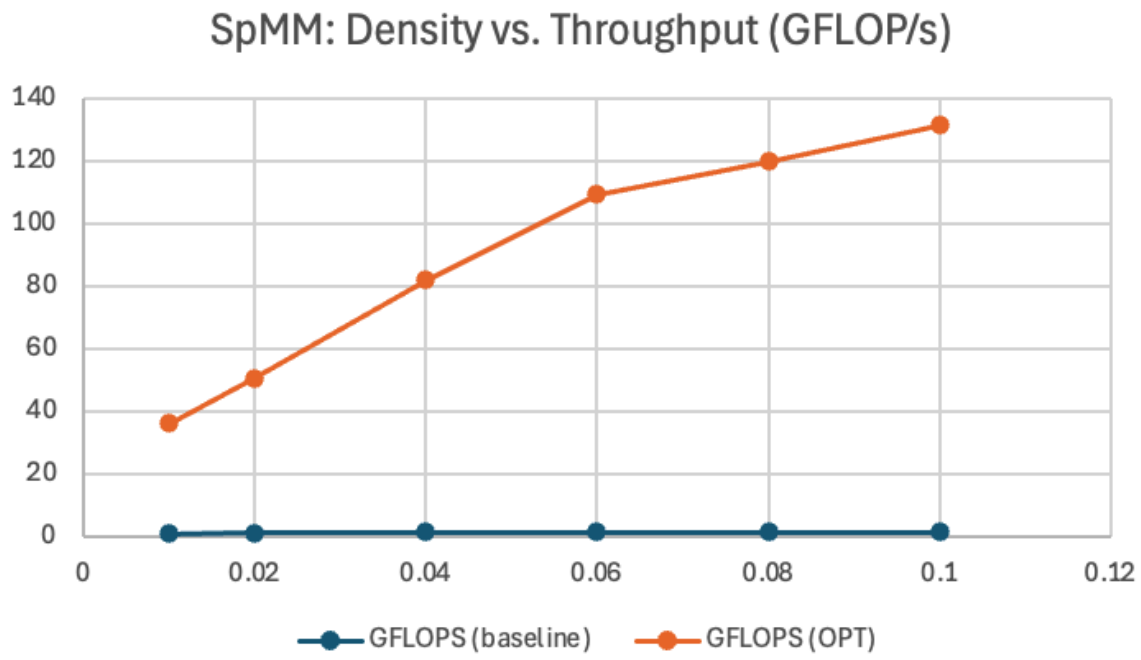

Figure 1

Figure 1 compares the throughput of the baseline and warp-coalesced implementations (M = K = 512, N = 64). The baseline implementation exhibited poor scaling, tapering off at approximately 1.40 GFLOP/s. This stagnation demonstrates that the kernel became latency-bound, with threads spending most of their cycles stalled on uncoalesced global memory loads.

In contrast, the warp-coalesced implementation scaled almost linearly with density, peaking at 131.61 GFLOP/s. This represents a 94x speedup over the baseline. The linear scaling confirms that the warp-optimized kernel transformed the bottleneck from instruction latency to memory bandwidth.

However, the 512 x 512 matrix size is small. Consequently does not provide sufficient work to fully utilize the Tesla V100 GPU. By increasing the matrix size to M = K = 8192, the implementation achieved 544.81 GFLOP/s. This significant increase can be attributed to improved warp occupancy; the larger grid enabled the hardware to effectively hide global memory latency by scheduling ready warps while others waited for data from memory.

Key Takeaways and Conclusion

The results of this experiment demonstrate that memory coalescing is the primary driver of performance in Sparse Matrix Multiplication (SpMM). By assigning a warp to each row and aligning thread accesses to adjacent columns of Matrix B, the optimized kernel achieved a 98x speedup over the baseline, which suffered from inefficient, uncoalesced memory reads. While throughput scaled linearly with density, the peak performance of 544.81 GFLOP/s (achieved at M = 8192) remains well below the V100's theoretical compute limit of 14,131 GFLOP/s. This gap highlights the fact that SpMM is memory-bandwidth-bound; the low arithmetic intensity of sparse addressing "starves" the CUDA cores, as the hardware must fetch row pointers and column indices for every floating-point operation. Ultimately, a high warp occupancy allowed the optimized kernel to approach the practical memory wall of the Volta GPU architecture while maintaining an absolute error of 1e-6 relative to the CPU reference.

Logs

M = K = 8192; N = 64

```
[ajl18@bc11u23n1 exp3_spmm]$ ./spmm_opt
Optimized SpMM: M=8192 K=8192 N=64 nnz=671203
Launching Kernel with Grid=1024, Block=256
Density: 0.01 | NNZ: 671203
Throughput: 544.808 GFLOPS
Max error = 3.8147e-06
[ajl18@bc11u23n1 exp3_spmm]$
```

Error - OPT

```
[ajl18@bc11u23n1 exp3_spmm]$ ./spmm_opt
Optimized SpMM: M=512 K=512 N=64 nnz=2612
Launching Kernel with Grid=64, Block=256
Density: 0.01 | NNZ: 2612
Throughput: 32.65 GFLOPS
Max error = 4.76837e-07
[ajl18@bc11u23n1 exp3_spmm]$
```

Error - Baseline

```
[ajl18@bc11u23n1 exp3_spmm]$ ./spmm_baseline
nnz = 2612\nDensity: 0.01 | NNZ: 2612
Throughput: 1.18714 GFLOPS
Max error = 4.76837e-07
[ajl18@bc11u23n1 exp3_spmm]$
```