Comparision of Matrix Multiplication in CUDA and Python

Sajina Pathak

March 8, 2025

Memory Access in Python (NumPy)

NumPy uses optimized **BLAS** (Basic Linear Algebra Subprograms) routines, which are often linked with **OpenBLAS** or **Intel MKL**, for efficient matrix operations. The matrices are stored in **row-major order**, meaning that elements in the same row are stored in contiguous memory locations. In a naive implementation of matrix multiplication, each element of matrix A and matrix B is accessed multiple times because each element of A contributes to multiple elements of the result matrix C, and similarly, each element of B is reused. Specifically, each element of A and B is loaded P and M times respectively, leading to significant memory access overhead.

Memory Access in CUDA (GPU Kernel Execution)

CUDA performs matrix multiplication using **parallel threads** that operate concurrently to maximize computational throughput. Unlike traditional CPU-based approaches, CUDA optimizes memory access by leveraging **shared memory and registers** to reduce redundant data loading. Instead of fetching elements from global memory multiple times, which can be slow, CUDA loads submatrices into **shared memory once per block**. This technique significantly reduces memory access overhead, ensuring that each element of A and B is accessed fewer times compared to a naive CPU implementation. Through this shared memory optimization, CUDA accelerates matrix multiplication by minimizing global memory transactions and improving overall computational efficiency.

article amsmath graphicx

Python vs CUDA Runtime Comparison

The table below compares the runtime of Python (NumPy) and CUDA for matrix multiplication across different matrix sizes.

Matrix Dimension	**Python (sec)**	**CUDA (sec)**
10x10	1.6e-05	0.004338
100x100	0.002791	0.001801
500x500	0.231472	0.001501
1000x1000	1.430036	0.001603
2500x2500	35.532359	0.00148
5000x5000	296.949013	0.001532
10000x10000	4356.425496	0.001462

Table 1: Runtime comparison of Python vs CUDA for matrix multiplication.

Discussion

The results clearly indicate that CUDA provides a significant speedup over Python (NumPy) for matrix multiplication, especially for larger matrix sizes. While Python relies on optimized BLAS libraries, it still incurs higher execution times due to its sequential nature and reliance on global memory for frequent data retrieval. In contrast, CUDA leverages parallel processing, shared memory, and register optimizations, drastically reducing execution time.

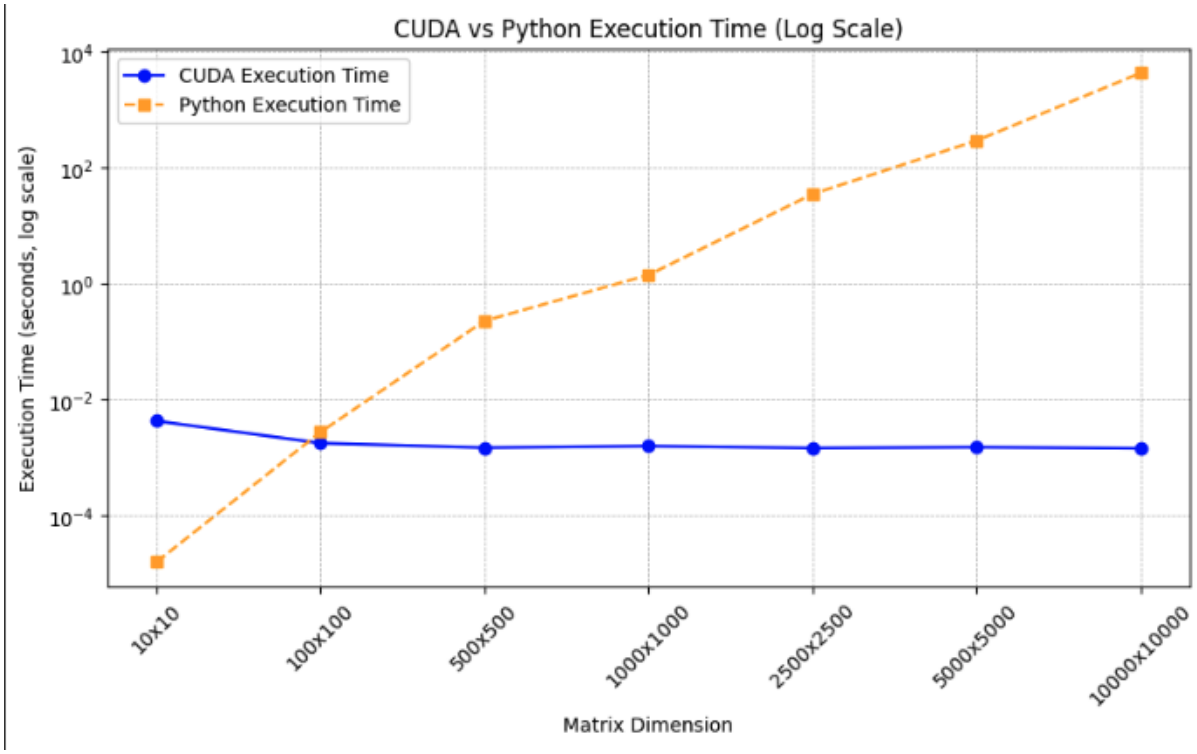

Figure 1: CUDA vs Python Execution Time (Log Scale)

For small matrices (e.g., 10x10 and 100x100), Python and CUDA have comparable execution times, with CUDA showing slightly higher overhead due to kernel launch costs. However, as matrix dimensions increase, Python's execution time grows exponentially, whereas CUDA maintains nearly constant execution time due to its efficient memory management and parallel execution model.

For a 10000x10000 matrix, Python takes over 4000 seconds, while CUDA completes the same operation in under 0.002 seconds, demonstrating a near 3 million times speedup. This drastic difference showcases CUDA's superiority in handling large-scale matrix multiplications efficiently.

Graphical Representation

The performance comparison is better visualized using a log-scale plot, where the exponential increase in Python execution time contrasts sharply with CUDA's near-constant runtime. Below is a plot representing the performance difference:

The figure demonstrates how Python struggles with large matrices, while CUDA remains computationally efficient due to its superior memory and execution model.

Resources Explored

The following resources were explored to understand and analyze CUDA-based matrix multiplication:

1. https://www.quantstart.com/articles/Matrix-Matrix-Multiplication-on-the-GPU-with-Nvidia-CUDA/Matrix-Matrix Multiplication on the GPU with Nvidia CUDA - QuantStart
2. https://github.com/juliansprt/CUDATestProcess?tab=readme-ov-fileCUDA Test Process - GitHub Repository
3. https://vitalitylearning.medium.com/a-short-notice-on-performing-matrix-multiplications-in-pycuda-cbfb00cf1450A Short Notice on Performing Matrix Multiplications in PyCUDA - Medium