Comparative Analysis of CPU and GPU Computation Using NumPy and CuPy.

Aman Rakesh Prajapati

A Project Submitted to

GRAND VALLEY STATE UNIVERSITY

In

Partial Fulfillment of the Requirements

For the Degree of

Master of Science in Applied Computer Science

School of Computing

April 2025

The signatures of the individuals below indicate that they have read and approved the project of Aman Rakesh Prajapati in partial fulfillment of the requirements for the degree of Master of Science in Applied Computer Science.

_____

Christian Trefftz, Project Advisor Date

_____

Dr. D. Robert Adams, Graduate Program Director Date

_____

<name of unit head>, Unit head Date

Abstract

This project investigates the performance comparison between NumPy (CPU-based computation) and CuPy (GPU-accelerated computation) for various linear algebra operations. Specifically, it has four key tasks: matrix-to-matrix multiplication, matrix-to-vector multiplication, eigenvalue computation, and singular value decomposition (SVD). The study was conducted using matrices of increasing size (1000x1000, 2000x2000, 3000x3000, and 4000x4000) to analyze how each library scales with computational complexity. Execution times were recorded, and visual comparisons were made for the first two operations. The results indicate that CuPy significantly outperforms NumPy in computational efficiency, particularly for large-scale operations. This project highlights the potential of GPU-accelerated computing for scientific and engineering applications, providing a pathway for optimizing high-performance numerical computations.

Introduction

With the increasing need for high-performance computing in scientific and engineering applications, optimizing numerical computations has become essential. NumPy, a widely used Python library, provides efficient CPU-based computation for matrix operations. However, as dataset sizes grow, CPU limitations become apparent. CuPy, a GPU-accelerated alternative, leverages CUDA to perform similar operations significantly faster.

The primary research problem is to quantify the performance gap between CPU and GPU implementations for large-scale matrix computations. The project aims to answer the following research questions:

1. How does CuPy's performance compare to NumPy for different matrix sizes?
2. How do matrix-to-matrix and matrix-to-vector multiplications scale across the two libraries?
3. What are the computational efficiencies for eigenvalue computation and SVD?

Our project systematically benchmarks these operations and visualizes the results to highlight the benefits of GPU acceleration. This study provides insights into choosing the right computation framework based on performance needs.

Background/Related Work

Numerous studies have explored numerical computation optimizations. NumPy has been the standard for array operations due to its ease of use and optimized backend (BLAS, LAPACK). However, GPU computing has revolutionized performance-intensive tasks. Libraries like TensorFlow and PyTorch use GPU acceleration extensively, yet comparative studies between NumPy and CuPy remain limited. Prior work shows GPUs excel in parallelized matrix operations, but real-world comparisons for different computational tasks are lacking. Our study contributes by providing empirical results on standard numerical operations across CPU and GPU frameworks.

Methods

The study was conducted using two separate scripts: one for NumPy and one for CuPy, ensuring a fair comparison. The benchmarks included:

1. **Matrix-to-Matrix Multiplication**: Computed using np.dot (A, B) and cp.dot (A, B).
2. **Matrix-to-Vector Multiplication**: Performed using np.dot (A, v) and cp.dot (A, v).
3. **Eigenvalue Computation**: Used np.linalg.eigvals(A) for NumPy; CuPy lacks direct support, so we used SVD as an alternative.
4. **Singular Value Decomposition (SVD)**: np.linalg.svd(A) and cp.linalg.svd(A) were used.

Both implementations recorded execution times for matrix sizes ranging from 1000x1000 to 4000x4000. The results were stored in JSON files and analyzed using Matplotlib for visualization.

Results/Discussion

Graphs comparing NumPy and CuPy performance for matrix-to-matrix and matrix-to-vector multiplications revealed a clear trend:

- CuPy significantly outperforms NumPy, particularly as the matrix size increases.
- For matrix-to-matrix multiplication, CuPy's execution time remained relatively low compared to NumPy's exponential increase.
- Matrix-to-vector multiplication showed CuPy maintaining a near-constant execution time, while NumPy scaled linearly.
- Eigenvalue computation and SVD further demonstrated CuPy's advantage in handling large computations efficiently.

While CuPy offers massive performance improvements, it requires a CUDA-compatible GPU. Additionally, memory transfer overheads between CPU and GPU can limit performance gains in smaller computations.

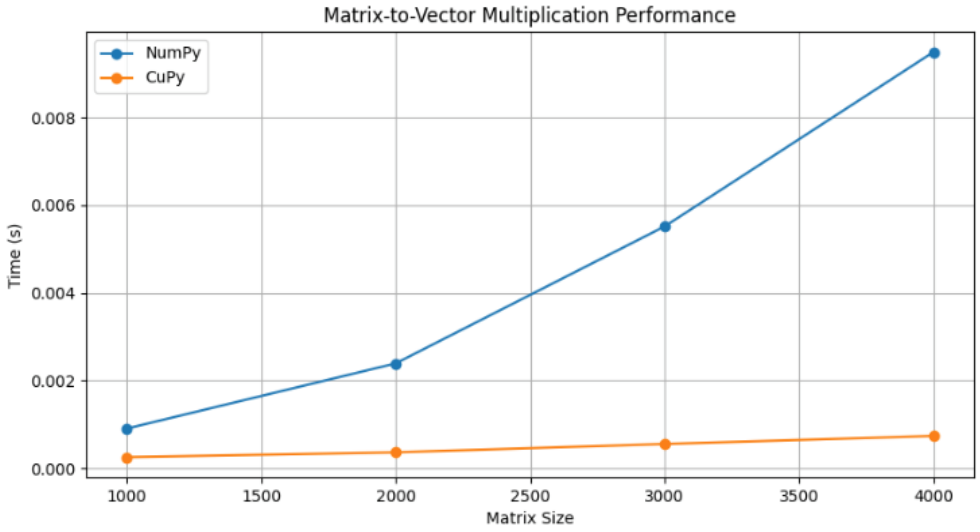

Fig.1: Matrix-to-vector Multiplication

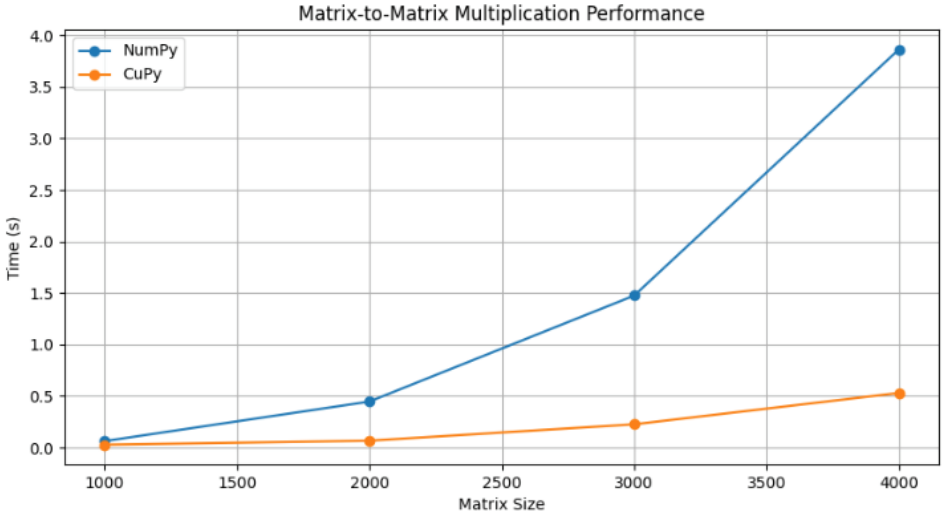

Fig. 2: Matrix-to-Matrix Multiplication

Conclusions

This project successfully demonstrated CuPy's superiority over NumPy for large-scale matrix operations, highlighting the benefits of GPU acceleration in numerical computing. Future work could explore mixed precision computing, memory optimizations, and more complex numerical methods. The findings encourage further adoption of GPU-based libraries for high-performance computing applications.

Bibliography

1. Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., ... & Oliphant, T. E. (2020). Array programming with NumPy. Nature, 585(7825), 357–362. https://doi.org/10.1038/s41586-020-2649-2
2. Okuta, R., Unno, Y., Nishino, D., Hido, S., & Loomis, C. (2017). CuPy: A NumPy-compatible library for NVIDIA GPU calculations. In Proceedings of Workshop on Machine Learning Systems (LearningSys) in The Thirty-first Annual Conference on Neural Information Processing Systems (NIPS). https://cupy.dev
3. Numpy Computation : Computation using Numpy
4. Cupy Computation : Computation using Cupy
5. Comparing the computation capabilities of Numpy & Cupy : Visualization Results