Is Python Slow or Fast?

Hector N. Rivera Miranda

Francisco A. Casiano Rosado

Gabriell Velez Agront

Universidad de Puerto Rico Recinto de Mayagüez

ICOM5015-001D Artificial Intelligence

Professor Jose F Vega Riveros

February 28, 2024

Is Python Slow or Fast?

This report presents a comparative analysis of array multiplication performance in Python using both pure Python and NumPy implementations. The study aims to assess the efficiency gains provided by NumPy's optimized array operations compared to traditional iterative approaches in pure Python. Through experimentation with arrays of varying dimensions, insights into the computational complexity and performance trade-offs between the two methods are explored.

Purpose and Key Question:

Array multiplication is fundamental in numerical computing, crucial for scientific and engineering applications [1]. The primary objective of this assignment is to compare the performance of array multiplication using pure Python and NumPy. The central question driving this study is whether NumPy's optimized array operations outperform equivalent operations implemented in pure Python, particularly as the dimensions of the arrays increase.

Experiment Design:

The experiment's design incorporates key concepts such as array multiplication algorithms, computational complexity analysis, and performance optimization techniques. Understanding the internal workings of both pure Python and NumPy implementations is essential for designing effective experiments and interpreting the results accurately.

Experimental Setup:

We generate random arrays of various dimensionalities, ranging from small arrays with dimensions less than 10, medium arrays with dimensions from 10 to 100 and large arrays with dimensions from 101 to 1000. By measuring the execution time for array multiplication using both pure Python and NumPy implementations across different dimensionalities, we enable a comprehensive analysis of performance scalability.

Analysis and Interpretation:

Our analysis is guided by insights from technical literature on Python and NumPy, including authoritative sources such as the NumPy documentation and scientific publications on numerical computing [2], [3]. We interpret the experimental data by comparing execution times, analyzing trends in performance scalability, and examining the impact of array dimensions on computational efficiency.

Comparison and Conclusion:

In comparing pure Python to NumPy for array multiplication, NumPy consistently demonstrates superior performance across various array dimensions. NumPy's optimized operations lead to faster execution times, especially noticeable with larger datasets [4]. The experiment's visualization underscores this point, with NumPy's efficiency evident in reduced execution times. This can be observed where in figures 1, 2 and 3 NumPy maintains a relative constant time complexity while Python has a linear time complexity. This is because NumPy was developed in C, and it is a low-level language that makes better management of both time

complexity and space complexity when dealing with integers and floats. Python has proven to be more versatile given that in its lists you can store integers, floats, and characters alike but this versality costs the before mentioned complexities [5]. Therefore, for efficient data processing and computation tasks, leveraging NumPy over pure Python is highly recommended.

Lessons Learned:

Through this assignment, we have gained valuable insights into the performance characteristics of array multiplication algorithms in Python. We learned the importance of choosing the right tools and techniques for numerical computing tasks and the significance of performance optimization in scientific programming.

Conclusion:

The comparative analysis of array multiplication performance in Python using pure Python and NumPy implementations provides valuable insights into computational efficiency and scalability. By leveraging NumPy's optimized array operations, researchers and practitioners can enhance the performance of numerical computations and accelerate scientific workflows.

Work Distribution:

Hector: Code refinement, report

Gabriell: Code buildup, report

Francisco: Video script, presentation, video editing

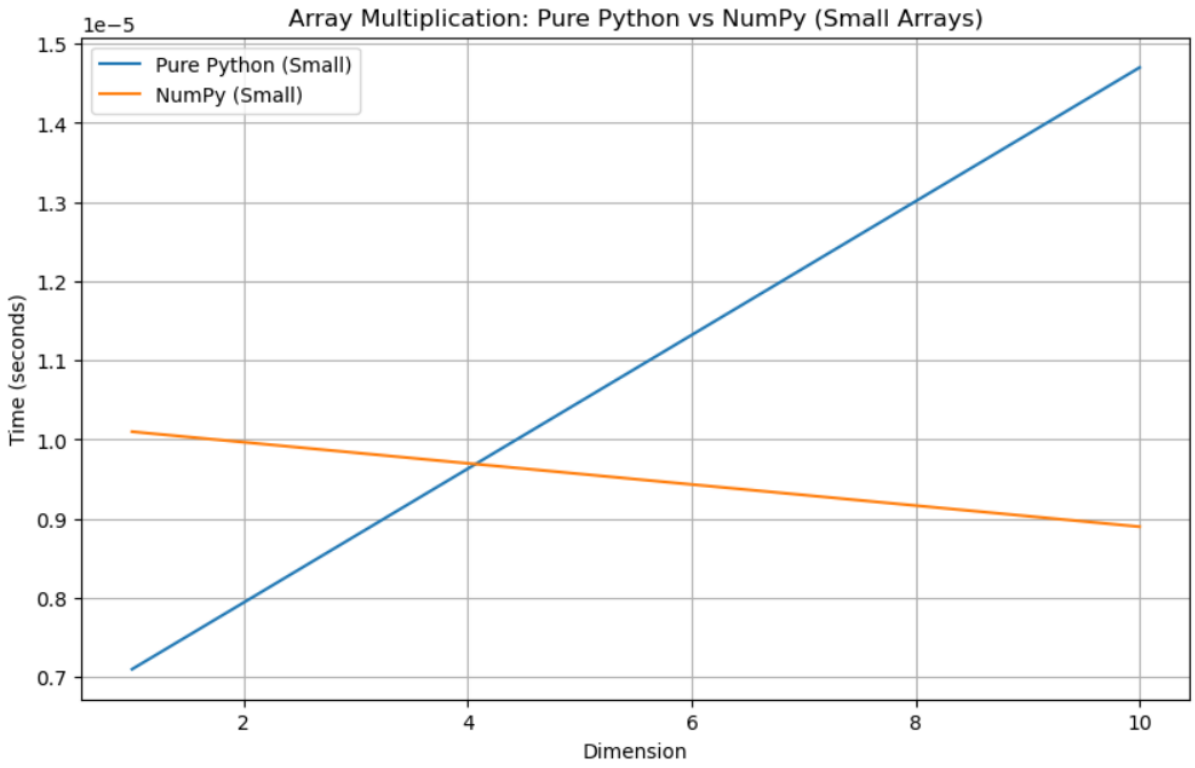

Figure 1: Small Array

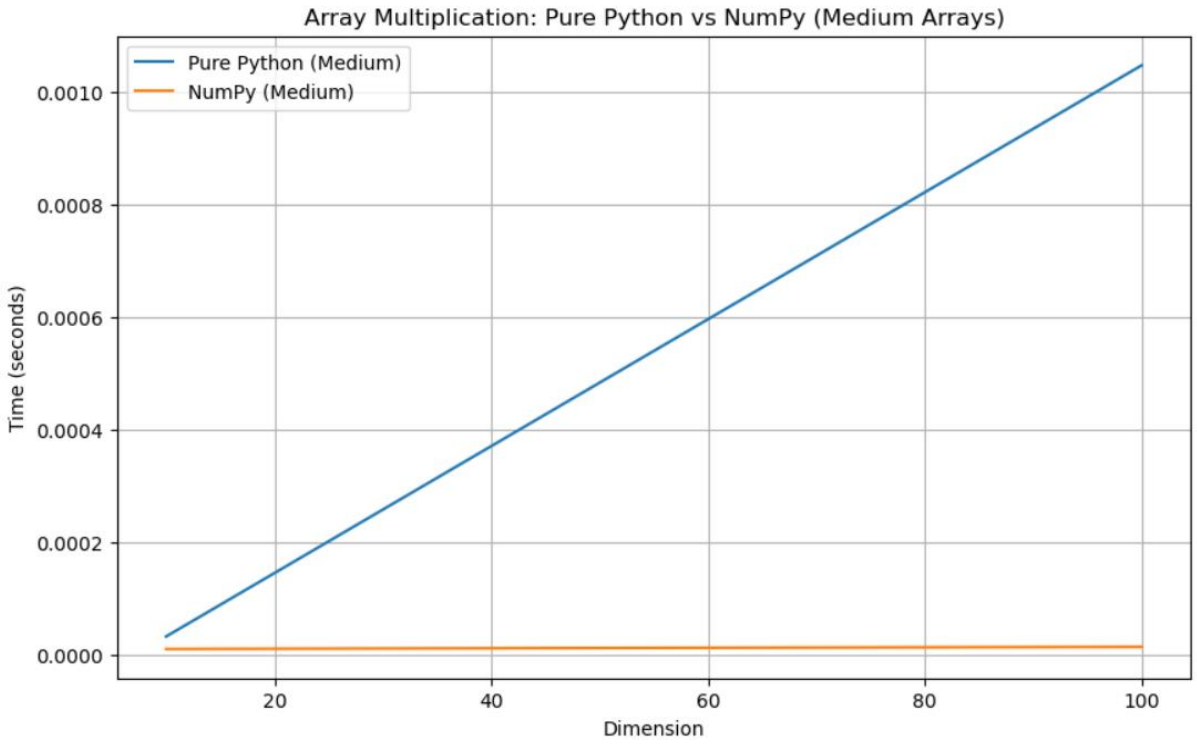

Figure 2: Medium Array

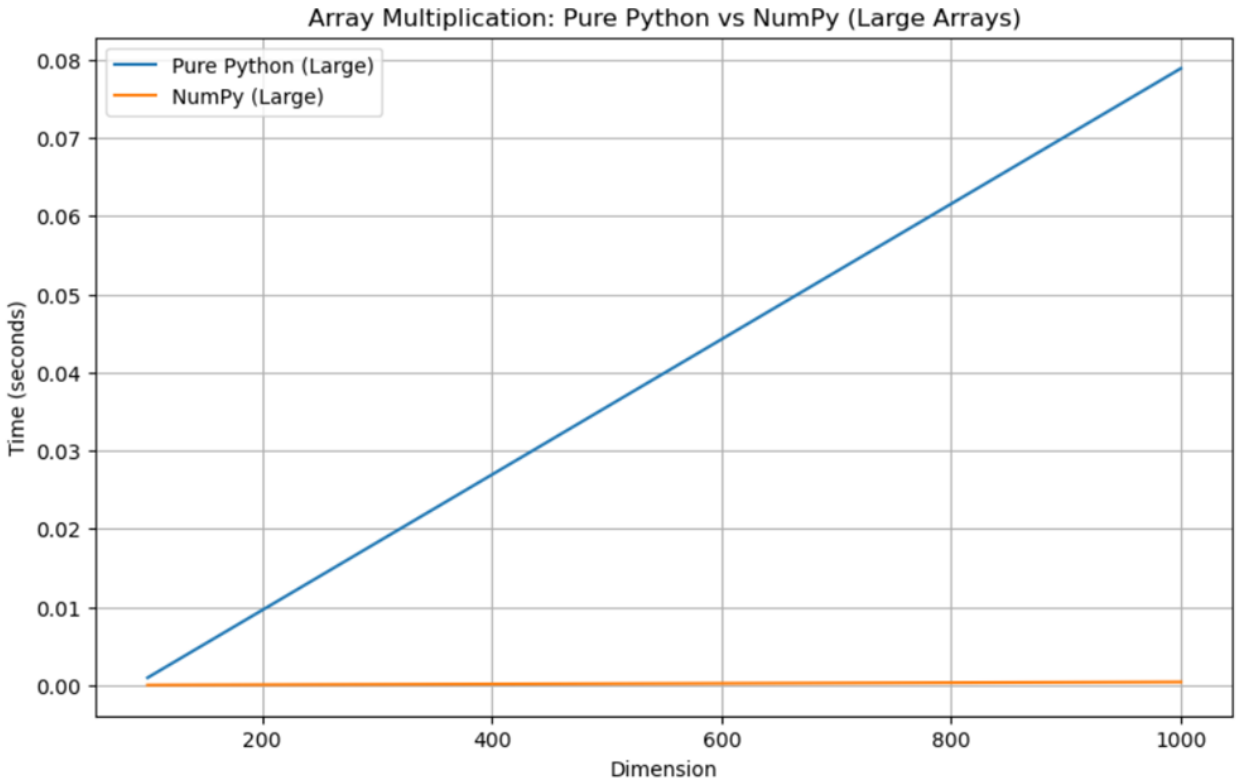

Figure 3: Large Array

References

[1] T. E. Oliphant, Oliphant, T.E. (2006) guide to NumPy. Trelgol Publishing. - references - scientific research publishing, https://web.mit.edu/dvp/Public/numpybook.pdf (accessed Feb. 26, 2024).

[2] C. R. Harris, Array programming with NumPy, https://jyx.jyu.fi/bitstream/handle/123456789/72093/1/s41586-020-2649-2.pdf (accessed Feb. 26, 2024).

[3] W. McKinney, "Python for Data Analysis, 2nd Edition," O'Reilly Online Learning, https://bedford-computing.co.uk/learning/wp-content/uploads/2015/10/Python-for-Data-Analysis.pdf (accessed Feb. 26, 2024).

[4] "The Python Language Reference," Python documentation, https://docs.python.org/3/reference/index.html (accessed Feb. 26, 2024).

[5] J. P. Jones, "A python list versus a Numpy Array," YouTube, https://www.youtube.com/watch?v=FQbbkCFWNjY&ab_channel=JohnPhilipJones (accessed Feb. 28, 2024).