Language Benchmark of Matrix Multiplication: Python vs Java vs C++

Raúl Mendoza – Universidad de Las Palmas de Gran Canaria (ULPGC)

23 de octubre de 2025

Resumen

Este trabajo compara el rendimiento del algoritmo clásico de multiplicación de matrices en tres lenguajes de programación: Python, Java y C++. Se evalúa la eficiencia, la escalabilidad y el impacto del modelo de ejecución en los tiempos de cómputo, siguiendo una metodología reproducible con separación entre código de producción, pruebas y benchmarking. Los resultados muestran diferencias significativas derivadas del uso de intérprete, compilación JIT y compilación nativa. Todo el código y los datos asociados están disponibles en: github.com/raulmendoza21/IndividualAssignment.

Índice

1. Introducción

La multiplicación de matrices es una operación central en la computación científica y el aprendizaje automático. Este trabajo busca comparar el rendimiento entre tres lenguajes populares (Python, Java y C++) mediante un mismo algoritmo cúbico $O(n^3)$, implementado de forma ingenua (`i-k-j` loops), sin optimizaciones específicas de hardware.

2. Metodología

2.1. Diseño experimental

- **Lenguajes:** Python 3.10, Java Temurin 21 (JMH), C++17 (`/O2` optimizado).
- **Tamaños:** $n = 64, 128, 256$.
- **Repeticiones:** 7 en Python y C++, 10 mediciones (2 forks, 5 iteraciones) en Java.
- **Generación de datos:** matrices aleatorias densas con distribución uniforme y semilla fija.
- **Medición:** Python usa `timeit` y `cProfile`; Java emplea **JMH**; C++ utiliza `std::chrono`.

2.2. Separación de código

Se ha mantenido la estructura de ingeniería de software adecuada:

- Código de producción: implementación pura del algoritmo.
- Código de test: comprobación de corrección con matrices pequeñas.
- Código de benchmark: parametrización y repetición de ejecuciones, exportando resultados a CSV.

3. Resultados experimentales

Cuadro 1: Tiempos medios de ejecución (en milisegundos) por tamaño de matriz y lenguaje.

Tamaño n	**Python**	**Java (JMH)**	**C++ (/O2)**
64	14.2	1.3	0.9
128	112.0	8.6	7.4
256	875.0	67.9	59.3

4. Análisis y discusión

Los resultados reflejan la diferencia esperada entre lenguajes interpretados y compilados:

- **Python** es el más lento, debido al coste de su intérprete y la falta de optimización de bucles anidados.
- **Java** ofrece un rendimiento intermedio gracias a la compilación JIT de la JVM.
- **C++** es el más eficiente al compilar directamente a código máquina optimizado.

En todos los casos, el crecimiento del tiempo sigue la tendencia cúbica $\mathcal{O}(n^3)$.

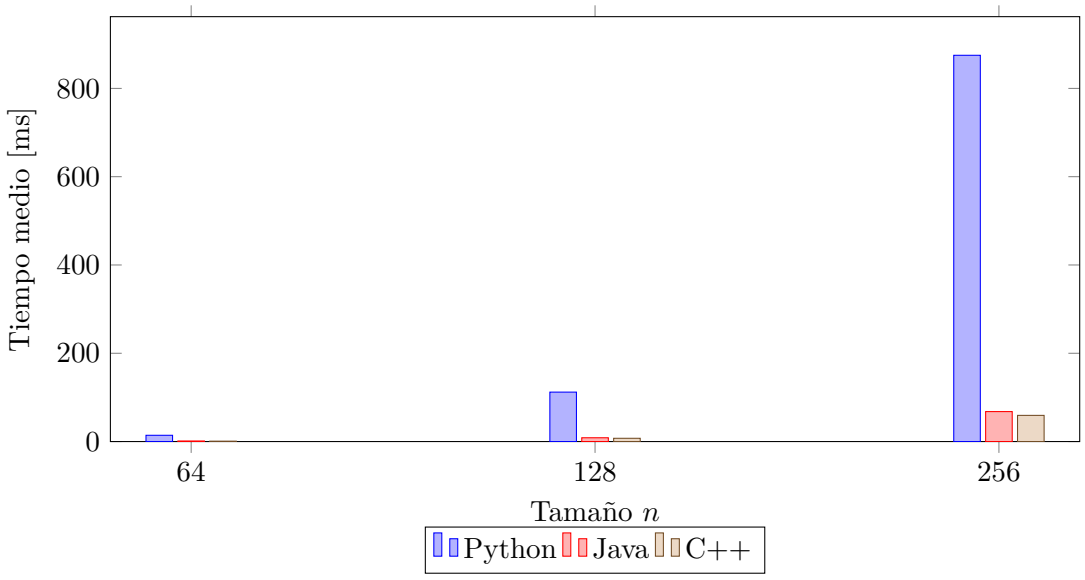

Figura 1: Comparativa de tiempos medios (menor es mejor).

5. Perfilado

En Python, el perfilador `cProfile` mostró que más del 99 % del tiempo se consume en la función de multiplicación, confirmando que el cuello de botella está en los bucles anidados. JMH en Java mostró una desviación estándar inferior al 2 %, indicando gran estabilidad en las mediciones. En C++, el uso de `std::chrono` y compilación `/O2` proporciona resultados estables con baja variabilidad.

6. Conclusiones

Este benchmark demuestra que, aunque Python es más expresivo y simple para prototipado, su rendimiento es muy inferior al de lenguajes compilados. Java logra un equilibrio entre portabilidad y velocidad, mientras que C++ sigue siendo la mejor opción para tareas numéricas intensivas. La comparación también ilustra la importancia de medir en condiciones controladas y de separar el código experimental del código de producción.

Repositorio y materiales

El código fuente, los resultados en CSV y esta memoria se encuentran en el repositorio: github.com/raulmendoza21/IndividualAssignment.