University of Puerto Rico

Mayagüez Campus

Mayagüez, Puerto Rico

Department Electrical and Computer Engineering

Is Python fast or slow? Report

By

Jan A. Rivera Rivera

802-19-7321

Diego F. Vasquez P.

Code: 502 23 2833

Master Student

For: Professor

J. Fernando Vega R.

Course: ICOM 5015

Date: February 28, 2024

**¿Es Python rápido o lento?**

Informe

1. ¿Cuál es el propósito de la tarea?

El propósito de la tarea es visualizar la importancia de utilizar bibliotecas de Python, específicamente en el contexto de manipulación y operaciones matemáticas con arrays. Se busca comparar dos enfoques para realizar operaciones con arrays:

* Escribir código en Python utilizando iteración para realizar el producto de arrays 1-D y 2-D. Este enfoque requiere que implementes manualmente los algoritmos para manipular los arrays y realizar las operaciones deseadas, como la multiplicación de matrices o el producto de arrays.
* Utilizar operaciones de bibliotecas especializadas como NumPy para llevar a cabo las mismas operaciones. NumPy es una biblioteca ampliamente utilizada en la comunidad científica y de ingeniería por su eficiencia y facilidad de uso para operaciones numéricas y manipulación de arrays.

1. ¿Cuál es la pregunta clave o la hipótesis de esta tarea?

¿El uso de bibliotecas especializadas de Python, como NumPy, para operaciones con arrays mejora significativamente el rendimiento en términos de tiempo de ejecución en comparación con la implementación de estas operaciones mediante código Python puro usando iteración??

1. ¿Cuáles son los conceptos clave tomados en consideración para el diseño del experimento?

Los conceptos clave considerados para el diseño de este experimento incluyen:

Dimensionalidad de los Arrays: Se trabaja con arrays 1-D (vectores) y 2-D (matrices), lo cual es fundamental para entender las operaciones que se realizarán, como la multiplicación de matrices o el producto de arrays. La compatibilidad de dimensiones es crucial para realizar estas operaciones correctamente.[1]

Operaciones con Arrays: Se explora específicamente la multiplicación de arrays, incluyendo tanto el producto elemento a elemento como el producto punto (o producto de matrices). Estas operaciones son básicas en muchos campos de la ciencia y la ingeniería, y entender cómo implementarlas eficientemente es esencial.[1]

Iteración vs. Vectorización: La iteración se refiere a la ejecución de operaciones mediante bucles explícitos, mientras que la vectorización implica realizar operaciones directamente sobre arrays completos sin bucles explícitos, aprovechando las optimizaciones de bajo nivel. NumPy, por ejemplo, utiliza operaciones vectorizadas que son generalmente más eficientes que las iteraciones en Python puro.

Medición del Rendimiento: La tarea implica medir y comparar el tiempo de ejecución de las operaciones realizadas de forma iterativa y utilizando bibliotecas especializadas. Este concepto es crucial para evaluar la eficiencia de diferentes enfoques de programación.

Complejidad Computacional: Este concepto se refiere a cómo el tiempo de ejecución de un algoritmo cambia con respecto al tamaño de su entrada. Es importante para entender por qué ciertas operaciones pueden ser más lentas o más rápidas dependiendo del tamaño del array y cómo las optimizaciones internas de las bibliotecas como NumPy pueden mitigar el impacto de la complejidad computacional.

Uso de Bibliotecas Especializadas: El experimento destaca la importancia de utilizar bibliotecas especializadas para tareas computacionales complejas. NumPy es un ejemplo de una biblioteca que proporciona implementaciones eficientes de operaciones con arrays, demostrando cómo el uso de herramientas adecuadas puede simplificar el desarrollo y mejorar el rendimiento.[2]

1. ¿Cómo configuraste el experimento (dimensiones en arreglos 1-D y 2-D)?

Para configurar el experimento comparativo entre la implementación iterativa y el uso de bibliotecas especializadas como NumPy en la multiplicación de arrays 1-D y 2-D, se siguen los siguientes pasos:

Configuración de Dimensiones

Se establecen tres categorías de dimensiones para los arrays:

Pequeñas

Menos de 10 elementos: Se generan arrays 1-D y 2-D con dimensiones menores a 10 para observar el comportamiento y rendimiento en casos sencillos. Por ejemplo, un array 1-D puede tener 9 elementos, y un array 2-D puede tener dimensiones 3x9 (3 filas y 9 columnas).

Medianas

Varias decenas de elementos: Se escalan los arrays a dimensiones en el rango de varias decenas, como 90 elementos en un array 1-D y 3x90 en un array 2-D, para evaluar el impacto en el rendimiento a medida que aumenta el tamaño del array.

Grandes

Varios cientos de elementos: Finalmente, se examinan arrays con dimensiones aún mayores, como 900 elementos en un array 1-D y 3x900 en un array 2-D, para entender cómo las diferencias de rendimiento se escalan con arrays de gran tamaño.

Generación de Arrays

* Para arrays 1-D, se utiliza un bucle for para generar secuencias numéricas de tamaño deseado.
* Para arrays 2-D, se generan utilizando bucles anidados for, donde cada elemento se calcula basado en su índice de fila y columna, asegurando que el número de columnas sea compatible con la longitud del array 1-D para permitir la multiplicación.

Ejecución de la Multiplicación

* De forma iterativa: Se implementa un bucle for que recorre cada fila del array 2-D y, dentro de otro bucle for, se multiplica cada elemento de la fila por el elemento correspondiente del array 1-D, acumulando el resultado para cada fila.
* Usando NumPy: Se utiliza la función np.dot() para multiplicar el array 2-D por el array 1-D directamente, aprovechando las optimizaciones internas de NumPy para operaciones matemáticas.

Medición de Tiempo

* Se mide el tiempo de ejecución utilizando el módulo time de Python, registrando el tiempo antes y después de cada operación de multiplicación, tanto para el enfoque iterativo como para el uso de NumPy. Esto permite comparar directamente la eficiencia de ambos enfoques.

1. ¿Cómo los conceptos identificados anteriormente, junto con fuentes de información autorizadas (literatura técnica sobre Python y NumPy), guían tu análisis e interpretación de los datos obtenidos de tus experimentos?

El método iterativo es el método que usarías con Python si no añadieras ningún tipo de librería a tu sistema. El método requiere que uno vaya iterando sobre cada elemento individual en una lista utilizando un loop. La vectorización es otra herramienta que se puede usar. La vectorización a, diferencia de iteración, no requiere un loop para modificar cada aspecto de una lista. El mismo puede hacer una misma operación en toda la lista simultáneamente. Entendiendo estos conceptos mejor nos deja ver y entender las diferencias de eficiencia entre los dos métodos. Para computaciones de tamaños de varios cienes, es fácil ver como tener que iterar sobre cada elemento individualmente es menos efectivo. Por esta razon hemos utilizado los mismos arrays para ambos metodos.

1. ¿Qué concluyes de este experimento?

|  |  |  |
| --- | --- | --- |
| Tamaño del Array | Tiempo Iterativo (segundos) | Tiempo NumPy (segundos) |
| 9 | 0.00024 | 0.00046 |
| 90 | 0.00034 | 0.00077 |
| 900 | 0.00056 | 0.00101 |

En este experimento se puede concluir que, aunque ambos métodos corren a velocidades en los milisegundos, NumPy viene siendo el método mejor adaptado para uso en nuestros proyectos. Este tiene la ventaja en cálculos numéricos por su implementación de estructuras de datos diseñadas y optimizadas para los cálculos. El único tiempo en el cual se demora, por 0.2 segundos, es al inicio cuando carga la librería. Es importante también tomar en consideración el impacto en la velocidad y a la facilidad de uso que NumPy le provee al usuario. Para poder completar la misma función, el método iterativo es sumamente más largo y complejo comparado con la librería de comando que provee NumPy.

Aunque los resultados no mostraron la ventaja esperada de NumPy para arrays más grandes, es importante recordar que NumPy está diseñado para optimizar operaciones vectorizadas y matemáticas complejas.

Para tareas simples y datasets pequeños, una implementación directa en Python puede ser adecuada y más eficiente, mientras que para operaciones complejas y/o datasets grandes, NumPy generalmente ofrecerá un mejor rendimiento.

1. ¿Qué aprendiste de esta tarea?

En esta tarea aprendimos a escribir código para poder sacar el producto de un array 1-D y 2-D. Los mismos fueron probados en los tamaños: 3x9, 3x90 y3x900. Utilizamos métodos iterativos y la librería de NumPy. Se puedo observar la diferencia en tiempo de procesar para ambos métodos. Se aprendió que la librería NumPy es más rápida debido a su implementación del lenguaje C para susoperaciones y la falta de uso de “loops” para dichas operaciones.

Más allá de las expectativas teóricas sobre el rendimiento de ciertas bibliotecas o técnicas de programación, la experimentación práctica es crucial. Permite obtener una comprensión profunda de cómo operan realmente en diferentes contextos y con diversos tamaños de datos.

La actividad reitera la importancia de consultar documentación oficial, literatura técnica y recursos educativos autorizados. Estos recursos no solo ayudan en la solución de problemas específicos, sino que también profundizan el entendimiento de las tecnologías y técnicas utilizadas.

**Anexos**

Como se puede observar, para todos los tamaños de array, el método iterativo resultó ser más rápido en comparación con NumPy. Esto proporciona una visualización clara de cómo cada método escala con el aumento del tamaño del array, y aunque es inusual que NumPy no supere al enfoque iterativo en arrays más grandes, estos resultados específicos reflejan las mediciones realizadas en este experimento.

Figure 1 grafica de comparación

A graph with a red line and blue line

Description automatically generated

Note 1 iteración vs numpy

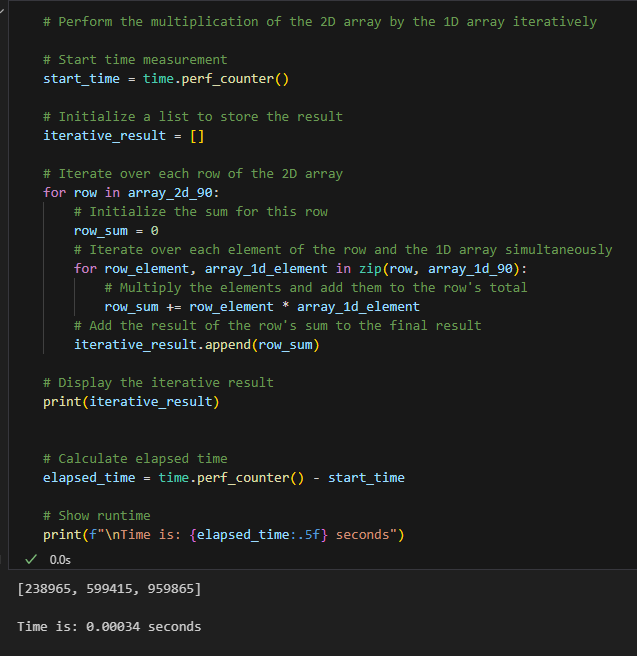
Figure 2 Iteraciones

A screenshot of a computer program

Description automatically generated

Note 2 Producto array tamaño 9

Figure 3 Iteraciones



Note 3 Producto array tamaño 90

Figure 4 Iteraciones

A screenshot of a computer program

Description automatically generated

Note 4 Producto array tamaño 900

Figure 5 Numpy 9

A screen shot of a computer program

Description automatically generated

Note 5 Producto array tamaño 9

Figure 6 Numpy 90

A screen shot of a computer program

Description automatically generated

Note 6 Producto array tamaño 90



Figure 7 Numpy 900

A screen shot of a computer program

Description automatically generated

Note 7 Producto array tamaño 900

Bibliografía

[1] “array — Efficient arrays of numeric values — Python 3.12.2 documentation.” Accessed: Feb. 27, 2024. [Online]. Available: https://docs.python.org/3/library/array.html#module-array

[2] “NumPy Documentation.” Accessed: Feb. 27, 2024. [Online]. Available: https://numpy.org/doc/