The Performance of Multithreading using Python for Matrix Multiplication

Veasna Bun

University of Washington

Abstract

The topic of concurrency and parallelism is one that is very important in the computer science community. For one to understand concurrency and parallelism one must practice its implementation. We will do this with the use of the threading module in Python and investigate its performance.

1. Introduction

With the rise of multi-core machine, software engineers are aware of how important concurrency and parallelism have become. The use of multithreading has become a hot topic of how software can be written in a concurrent and parallelism way to make program run more efficiently. The purpose of this paper is to see how multithreading can be used to make algorithms perform more efficiently. This can be done in many ways, but for us we will do this by first going over the basis concept of matrix multiplication. Then we will dive into the concept of multithreading, so programmers can see its usefulness. At lastly, we will combine matrix multiplication and multithreading with the use of Python and investigate its performance.

[Reference(Wilson, 2019)]

2. Implementations of Python Matrix Multiplication

One of the most important operations with two matrices is matrix multiplication. In this chapter we will go over how matrix multiplication is computed. Follow with examples that shows hand-on arithmetic and its equivalent Python code.

2.1 Matrix Multiplication Basic

Before getting started, there are a few things to keep in minds when multiplying two matrices.

Consider matrix A and B are in the fixed order of AB, where A is the 1st matrix and matrix B is the 2nd. Matrix A and B are both *m* by *n* matrix, where *m* is representing the number of rows and *n* represent the number of columns. We can represent the two matrices in the following form shown:

$$A = \begin{bmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1} & \cdots & a_{mn} \end{bmatrix} B = \begin{bmatrix} b_{11} & \cdots & b_{1n} \\ \vdots & \ddots & \vdots \\ b_{m1} & \cdots & b_{mn} \end{bmatrix}$$

Keep in Mind:

- Order matter: In most cases $AB \neq BA$
- Matrices can only be multiplied if the number of columns *n* in the 1st matrix equal to the numbers of rows *m* in the 2nd matrix. AB can only be computed if and only if $A_{column:\, n} = B_{row:\, m}$
- The resulting matrix C is the multiplication of the dot product of the 1st matrix with columns of the 2nd matrix.

$$AB = \begin{bmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1} & \cdots & a_{mn} \end{bmatrix} \begin{bmatrix} b_{11} & \cdots & b_{1n} \\ \vdots & \ddots & \vdots \\ b_{m1} & \cdots & b_{mn} \end{bmatrix} = C$$

$$C = \begin{bmatrix} a_{11} \times b_{11} + \ldots + a_{1n} \times b_{m_1} & \cdots & a_{11} \times b_{1n} + \cdots + a_{1n} \times b_{mn} \\ \vdots & \ddots & \vdots \\ a_{m1} \times b_{11} + \ldots + a_{mn} \times b_{m_1} & \cdots & a_{m1} \times b_{1n} + \cdots + a_{mn} \times b_{mn} \end{bmatrix}$$

[Reference (Leon, 2014)]

2.2 Matrix Multiplication Arithmetic

Now that we understand basic of matrix multiplication, we can now explore a few examples by computing the matrix multiplication of two matrices by hand. Then shown the equivalent computation with Python.

Example 1.

Let us begin by examining the case of A and B, where both being 2×2 matrices.

If we set

$$A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \text{ and } B = \begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix}$$

and defining the product of AB by

$$AB = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix} = \begin{bmatrix} 15 + 2 \times 7 & 1 \times 6 + 2 \times 8 \\ 3 \times 5 + 4 \times 7 & 3 \times 6 + 4 \times 8 \end{bmatrix} = \begin{bmatrix} 19 & 22 \\ 43 & 50 \end{bmatrix}$$

We can also represent the same computation in Python.

```
A = [[1,2],[3,4]]

B = [[5,6],[7,8]]

np.dot(A,B)

=> array([[19, 22],

=>        [43, 50]])
```

Example 2.

Let us examine the case of A and B, where A is 2×3 matrix and A is 3×2 matrix.

If we set

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} \text{ and } B = \begin{bmatrix} 7 & 8 \\ 9 & 10 \\ 11 & 12 \end{bmatrix}$$

and defining the product of AB by

$$AB = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} \begin{bmatrix} 7 & 8 \\ 9 & 10 \\ 11 & 12 \end{bmatrix} = \begin{bmatrix} 1 \times 7 + 2 \times 9 + 3 \times 11 & 1 \times 8 + 2 \times 10 + 3 \times 12 \\ 4 \times 7 + 5 \times 9 + 6 \times 11 & 4 \times 8 + 5 \times 10 + 6 \times 12 \end{bmatrix}$$

$$= \begin{bmatrix} 58 & 64 \\ 139 & 154 \end{bmatrix}$$

We can also represent the same computation in Python.

```
A = [[1,2,3],[4,5,6]]

B = [[7,8],[9,10],[11,12]]

np.dot(A,B)

=> array([[58, 64],

=>        [139, 154]])
```

2.3 Matrix Multiplication Takeaway

The takeaway from this chapter is to give the reader an understanding of how matrix multiplication is done via by hand and/ or via python code. The understand of both is essential when wanted to compute matrix multiplication using multithread in Python.

3. Implementations Multithreading in Python

A way to accomplish concurrency and parallelism is with the use of multithreading. In this chapter we will explore the concept of multithreading. Then we will see how multithreading is accomplished with Python via code examples.

3.1 Multithreading

In a multithreaded process on a single processor, the processor can switch execution resources between threads, resulting in concurrent execution.

A traditional or heavy weight process has a single thread of control, where only one task can be accomplished at a time.

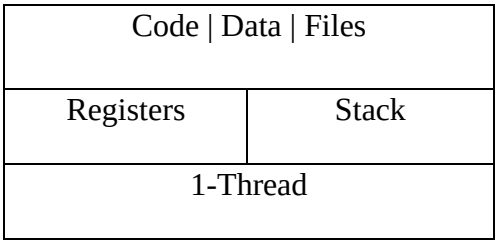

Figure 1:*Single Thread: traditional or heavy weight process*

Multithreading is when a process has multiple threads of control. It can perform more than one task at a time. In multithreading, each threads have its own registers and stacks and share code, data, and files.

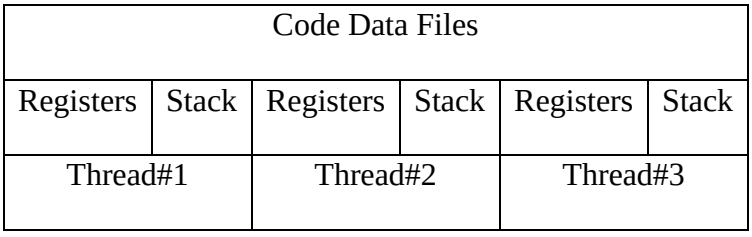

Figure 2:*Multithreading: Each thread has its own registers and stack. Code, Data, and File are shared.*

Here we see why multithreading can be useful. Multithreaded programs can run much faster than on a traditional or heavy weight process. If a task can be completed by breaking down its portion into smaller portion, multithreading can run these tasks concurrently. Resulting in improve performances. [Reference(Neso Academy, 2023, 0:02:40)]

3.2 Multithreading in Python

For a user to use the multithreading module in Python they must first import its code. We can do this by just using the import statement shown in the following code.

```
import threading
```

We first create thread class called Thread1 its code following:

```
class Thread1(threading.Thread):
	def __init__(self, id, name):
	threading.Thread. __init__(self)
	self.id = id
	self.name = name
	def run(self):
		#do some work
```

User may also use the above code to create multiple threads class. Only the class name Thread1 will need to differ. The run(self) does some work that may differ from another. User may want to define their own computation work here.

Creating a thread, the code use follow:

```
thread1= Thread1(1, “Thread 1”)
```

Finally, to start the thread.

```
thread1.start()
thread1.join()
```

Congratulations, users now have a template code for multithreading in Python.

[Reference(Rossum & Drake Jr)]

3.3 Multithreading in Python Takeaway

In this chapter, we focus on a brief overview of what multithreading is. By now readers should be able to implement their own program using multithreading in Python and a understand of how multithreading can improve the performance of an algorithm.

4. Performance Results: Matrix Multiplication & Multithreading

In this chapter we will tie in chapter 2 and chapter 3. We will investigate the performance of doing matrix multiplication using the threading module in Python.

4.1 Performance Results

The results of matrix multiplication of two matrices are shown in the Figure 3:

Single - Thread	2 – Threads	3 – Threads
Execution Time: 0.999 milliseconds	Execution Time: 1.203 milliseconds	Execution Time: 1.503 milliseconds
Note the resulting execution time vary. This is one result of the many resulting outcomes.		

Figure 3: *: Matrix Multiplication & Multithreading Results.*

The resulting time when using multiple threads seem to be slower than running on a single thread. After further investigation, Python threading module only let one execution to run at a time. What it's happening here is that each execution is occurring, and threads are switching to one another. This makes the resulting execution time of multiple threads much slower than single thread. [Reference(Rossum & Drake Jr)]

4.2 Matrix Multiplication & Multithreading Python Document

Reader may also want to review the full documented Python code. The following is a link to the repository hosted on github.

Using Jupyter Notebook via Binder: https://github.com/veasnab/tmath208.

5. Conclusion

We have gone over the basic concept of matrix multiplication and dive into the concept of multithreading. We then intertwined the two concept and saw earlier from chapter 4, that performance of the executions time has gotten slower with the use of multiple threads. This does not mean that the threading module from Python is completely useless. Instead, it gave us the user an insight how one can implement this module to their benefit. One benefit can be that one may want to stream input with a single thread and perform computation on the streaming input with another thread. We end this paper here and leave reader to their newfound knowledge of multithreading and matrix multiplication; in hope that it will benefit them when deciding to work on their own projects involving the threading module in Python.

References

[Neso Academy]. (2023, February 16). *Introduction to Threads* [Video]. Youtube. https://www.youtube.com/watch?v=LOfGJcVnvAk&ab_channel=NesoAcademy

Leon, S. J. (2014). *Linear Algebra with Applications* (7th ed., pp. 30-36). Pearson Education.

Rossum, V. G., & Drake Jr, F. L. (1995). *Threading - Thread-based parallelism*. Python Reference Manual. Retrieved February 16, 2023, from https://docs.python.org/3/library/threading.html

Wilson, C. (2019, June 24). *Multithreading and concurrency fundamentals*. Educative. Retrieved February 16, 2023, from https://www.educative.io/blog/multithreading-and-concurrency-fundamentals