

Caleb Besser
12/9/2023

Final Project Review

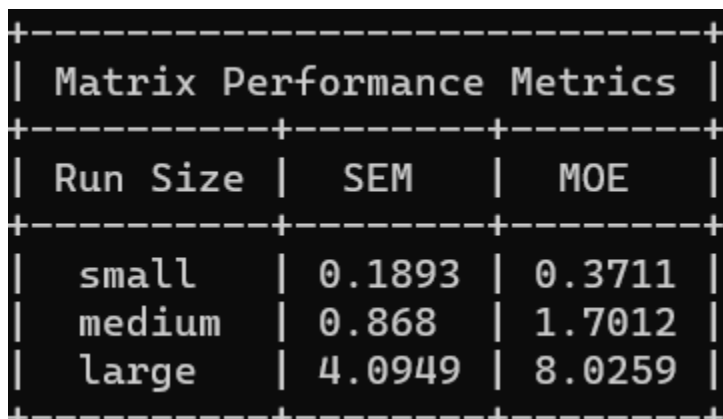
Question:

The question I chose was how does the C language compare to another language in terms of threads. The other language I chose was python, in which I had to use multi-processing instead of threads.

Setup:

As the only real way I could do this, as I messed up my credits for google cloud services, I used the Ubuntu VM that I have been using for assignments this semester. It's running Ubuntu Server 22.04, and as has been pointed out to me, my speeds will vary due to the fact that this code is not the only thing running on my machine. As I have my browser, discord, and other stuff open at the same time, that could be slowing it down.

I ran my tests using a python script , which compiled and ran all the code and spit out the numbers I needed for this review. Here is a picture of the output.

A screenshot of a terminal window displaying a table of matrix performance metrics. The table has three columns: Run Size, SEM, and MOE. The rows represent small, medium, and large matrix sizes. The values for SEM and MOE increase significantly with the size of the matrix.

Matrix Performance Metrics		
Run Size	SEM	MOE
small	0.1893	0.3711
medium	0.868	1.7012
large	4.0949	8.0259

This picture is from the Matrix Multiplication Python version output.

I didn't know what section to put this in so I'll put it here, the three programs that I coded are the Python and C Addition code, and the Monte Carlo pi C code. The other three I also coded, but I used chat GPT to help me with.

Test Results:

I debated the best way to show the results, but I think I will just screen shot all the outputs to show here. The script runs each program a total of 60 times, give or take some based on whether it met the margin of error requirements. This 60 times is split between 3 differing inputs, whether that be size of array, size of matrix etc.. Here is the final output after all the tests have run. The first 3 are the c programs, then the last 3 are the python programs.

+-----+			
Matrix Performance Metrics			
+-----+			
Run Size SEM MOE			
+-----+			
small 0.0668 0.1309			
medium 0.0267 0.0524			
large 1.6947 3.3216			
+-----+			
+-----+			
Monte Performance Metrics			
+-----+			
Run Size SEM MOE			
+-----+			
small 0.3729 0.7309			
medium 1.2943 2.5368			
large 2.7188 5.3288			
+-----+			
+-----+			
Addition Performance Metrics			
+-----+			
Run Size SEM MOE			
+-----+			
small 0.0573 0.1123			
medium 0.036 0.0706			
large 0.1467 0.2875			
+-----+			
+-----+			
Matrix Performance Metrics			
+-----+			
Run Size SEM MOE			
+-----+			
small 0.2154 0.4222			
medium 0.9794 1.9196			
large 4.0195 7.8782			
+-----+			
+-----+			
Monte Performance Metrics			
+-----+			
Run Size SEM MOE			
+-----+			
small 0.7262 1.4234			
medium 1.4822 2.9051			
large 2.1744 4.2618			
+-----+			
+-----+			
Addition Performance Metrics			
+-----+			
Run Size SEM MOE			
+-----+			
small 0.1086 0.2129			
medium 0.1236 0.2423			
large 0.1257 0.2465			
+-----+			
+-----+			

The main variations I have in the testing is the size at which I am stress testing the programs with, for example the monte carlo pi one I am doing, 500, 1000, 1500 as the input. The other way I have variety is the 3 different programs, these being Monte Carlo Pi, Matrix Multiplication, and Array Addition. Originally DNS lookup was my third, but it just would not work for some reason no matter what I did, and we were reaching the end of the semester so I sucked it up and did another one.

The amount of tests I ran for each was dependent on how many were needed to get the Margin of Error(MOE), below 10%. So for some it's more, some less. For example I needed to run the C Matrix Multiplication Large edition, 40 times instead of 20 times in order to get it below 10.

Conclusion:

It's fairly obvious based on the results, that C is faster. The MOE was overall lower on basically everything. Though not actually everything. But this is due to me using random numbers in the generation. So there are chances for python to slightly get an edge, but it's only ever by about 0.2-ish.

What I learned:

To be honest, I knew going into this that C was going to be faster. However, I did learn stuff along the way. The main thing I learned about, and the reason I chose python, was Python's Multiprocessing system. For example it's basically automatically protected, unlike c where you need to use mutexes, so that's cool.

As a side note, as I don't really ever code in python. Man python is really easy to code in, I wish I got to more. But obviously even just based on my results you can see the speed difference. If I was doing more complex things, I could imagine just how much it would start slowing down.

Another side note, I wanted to show them all on a graph as a visual representation of the speeds, but my Ubuntu is headless, so it can't run any graphics as far as I know. Maybe there is a way, Idk. Like using ascii characters to make a bar chart or something. That would be something to add in the future.

References:

The only thing I used was Chat GPT, for when I was stuck or needed help understanding how to do something.