EEE3096S Practical 1

Mutikedzi Mudzanani, Lutendo Mulaisi

August 14, 2022

1 Introduction

Run-time speed of a program on a computer depends on many parameters. Some of the parameters are threads, compiler-flag and the width of the bits the program is using. Research shows that the number of threads are not exactly proportional to the speed of the program. Compiler flags influences the instructions to be executed. The bit-width are the slots of memory that the values in the program are stored in.

2 Method

2.1 Running different threads

This experiment consist running a program with 2, 4, 8, 16 and 32 threads. The program is ran five times and the average of the times are taken. The thread count with the smallest average is then taken to run with different flags.

2.2 Running different flags

Just like running different threads, the different flags are compiled and ran for five times then the average is taken. Again, the flag with the smallest time is taken to run at different bit-width.

2.3 Running different bit-width

Finally, the program is ran using different bit-width, five times then get the average. The width with smallest average time is then taken.To determine the best combination, from the above, the best thread count is comined with the best flag count and the best width count.

3 Results

Table 1: Running different flags with 4 threads

threads	Average time [ms]
2	37,68954
4	14,6458
8	27,1508
16	23,7954
32	29,5234

Table 2: Running different flags with 4 threads

Flags	Average time
O0	7,6308
O1	13,608356
O2	32,651
O3	21,6398
Ofast	17,1598
Os	4,18
Og	20,925
funroll-loops	19,8252

Table 3: Running different bit-width with 4 threads and flag -Os

Data type	Average time
float (32 bit)	4,18
double (64 bit)	10,351002
fp16 (16 bit)	21,99488
funroll-loops	19,8252

13. The execution time for python was 1579.535 ms, and the execution time for C was 46.128 ms. C

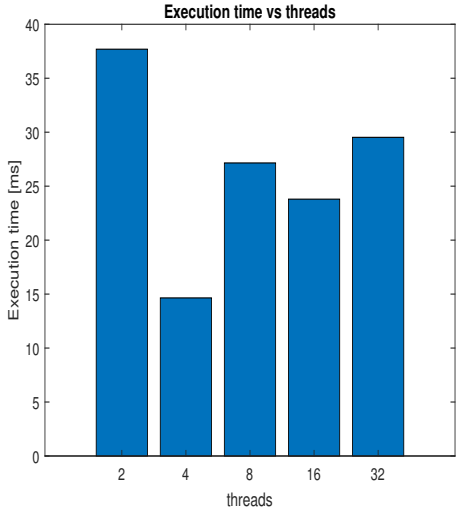

Figure 1: Execution time vs Threads

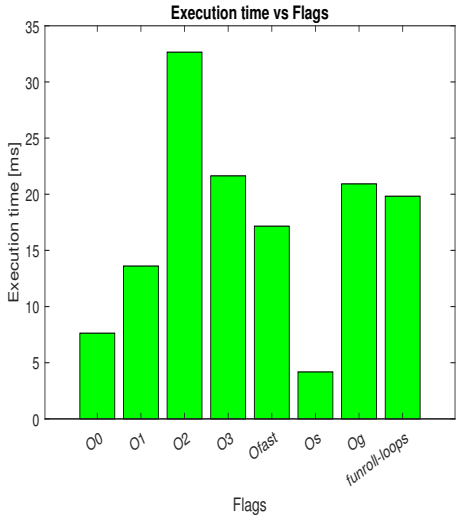

Figure 2: Execution time vs Threads

had a much faster execution time due to python being an interpreted language. The instructions from python are not ran exactly by the CPU, meanwhile C has its code compiled and the instructions are run at a low level much faster.

14.4 No, the code does not run faster as the number of threads increase. The speed increases from 2 threads to 4 threads then it starts to decrease from 8 threads to 32 threads. The raspberry pi has 4 cores, this limits it on much it can improve the speed. When the threads increase beyond 4, this makes the CPU wait for the 4 cores to finish the operation then run the other 4 threads. This dividing and combining of tasks increases the overall time the program takes to run.

17. The flag with the highest speed is -Os, this was unexpected. The expected flag to have the highest speed was -Ofast. This might be due to the pi emulator having small storage. So, when the pi runs the code optimised for storage.

18. Float – 32 bits Double – 64 bits fp16 – 16 bits
19. The best combination was using 4 threads, -Os, and data size of float.

4 Conclusion

13.

tpython = 1579.535 ms and tc = 69.462 ms

The time it took for python code to run is shorter than the time it took the C program to run.

¡comment on this observation¿

14.4

Making the pragram run on 2 and 4 threads improved the speed. The started decreased, from thread 8, 16 and 32. Time is spent dividing up and adding up tasks among threads, that is why the speed decreased.