Matthew Hunter Goodroe
goodroem@oregonstate.edu

Project #0
Simple OpenMP Experiment

<u>Introduction</u>:

For the OSU course CS475's Project 0, students were assigned to compile and implement a provided program, adjust the number of threads utilized, and compare and analyze the results. The focus of the comparisons is the computer's specs, performance results, speedup, and parallel fraction.

<u>Approach</u>:

After reviewing the provided program, I entered the VS Code on my personal computer. I gathered my computer's hardware specs, compiled the code through clang, and ran it with the instructions specified 1-thread and 4-thread settings. The specs and results can be seen below:

Trial 1 System Info

Machine	2021 Apple MacBook M1 Pro
Type	Laptop
Chipset	Apple M1 Pro
Number of cores	8 (6 performance + 2 efficiency)
Memory	16 GB

Trial 1 Performance Results:
- 1 Thread: 260 MegaMults/Sec
- 4 Threads: 348.83 MegaMults/Sec

1-thread-to-4-thread Speedup: 1.34

Parallel Fraction: .3389

I have used my 2021 Apple MacBook M1 Pro for the entirety of my time at Oregon State with no complaints on performance, and I was surprised by the results. The difference between 1 Thread and 4 Threads was a lot lower than expected, which in turn provided a lower Speedup and Parallel Fraction. As I was allocating more computing power by increasing the thread count, I expected the 4 Thread performance to be much higher.

Out of curiosity, I decided to port into the OSU servers and run the same tests to see how my personal computer compared. After logging in, providing the same program, compiling using gcc, and then running with the specified 1 thread and 4 thread settings, I received the following results.

Trial 2 System Info

Machine	OSU Flip Server
Type	Server
Chipset	Intel(R) Xeon(R) Gold 6126 CPU @ 2.60GHz
Number of cores	8
Memory	unknown

Trial 2 Performance Results:

- 1 Thread: 439.99 MegaMults/Sec
- 4 Threads: 1533.66 MegaMults/Sec

1-thread-to-4-thread Speedup: 3.49

Parallel Fraction: .95

This performance was very surprising. The results aligned to the provided guidelines in the project instructions and was much more effective in running the program in parallel.

Conclusion:

As I had 2 sets of data to compare on 2 different machines, this provided an interesting comparative analysis of determining why there was a difference in results. I knew the speedup would be less than 4, since not all of the program can be parallelized. For example, array initialization runs serially, and there also may be other workload or resource limitations. The higher speedup results from the OSU Flip Server show that its hardware is better optimized for parallel execution compared to the 2021 Apple M1 Pro. This matched up with the higher Parallel Fraction on the OSU Flip Server, as it reveals that 95% of the execution time can be reduced by using parallel processing, in comparison to the 2021 Apple M1 Pro's 34%. This can be seen in the table below:

Machine Performance Comparison

Machine	1 Thread (MegaMults/Sec)	4 Threads (MegaMults/Sec)	Speedup	Parallel Fraction
OSU Flip Server	439.99	1533.66	3.49	.95
2021 Apple M1 MacBook Pro	260	348.83	1.34	.3389
Difference	*179.99*	*1,184.82*	*2.15*	*.6111*

Upon reflection, the results from the two different machines make clear sense. The OSU Flip Server is designed to handle multiple and varying educational workloads of different students while maintaining a constant power supply. A personal laptop such as a 2021 Apple M1 MacBook Pro is designed for balancing battery life and user-demanded performance. Further exploration into each machine's architecture would likely reveal design and manufacturing choices that enhance or limit parallel performance. In this case, parallel processing would be more effective on the OSU Flip Server.

This project not only provided performance data that showed the ability of parallel processing and using multithreads to improve performance, but also that, even though different machines can use the same number of threads, there is more that goes into optimizing parallel processing.