Clinton Hawkes

hawkesc@oregonstate.edu

CS475-400

Project #4

1. This was run on my personal laptop:

Lenovo X1 Extreme

Core i7 9750H (6 cores/12 threads)

32GB ram

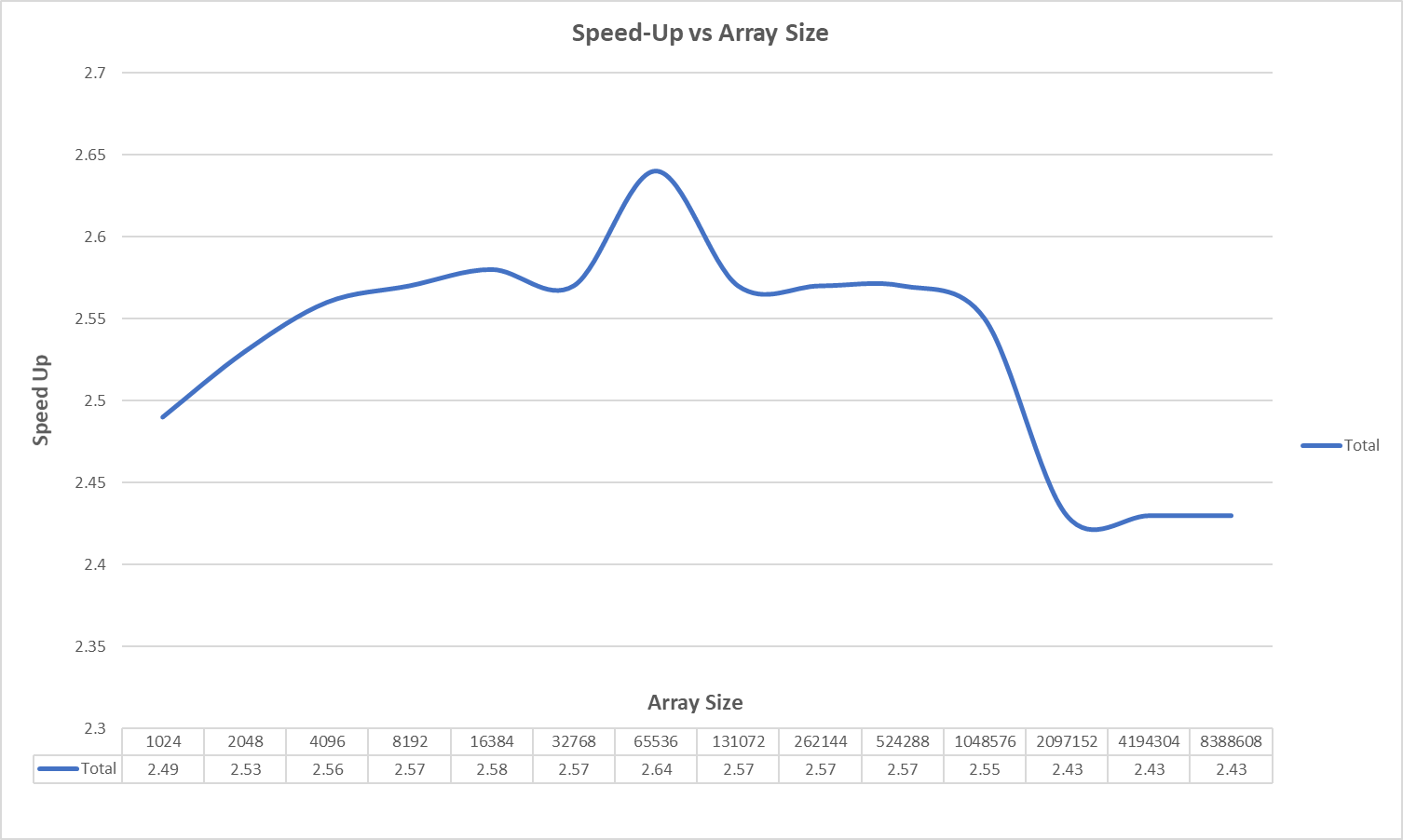
Nvidia GTX 1650

Running Linux kernel 5.5.9

1. Here is the table of performance figures I obtained.



1. Graph of SIMD/NON-SIMD speed-up:



1. The pattern I’m seeing in the speedup is it starts low, rises up and peaks when array size is 65536, and then decreases and levels off at a speedup of 2.43. I think the graph looks a little weird with its abrupt peak and sharp drop-off, but I ran the simulation several times, and there was always this type of variation in speedups.
2. The speed ups are not consistent across the different array sizes. I was expecting to get similar results to the lecture slides. That was not the case for me.
   1. First, my max speedup for SIMD using one core was 2.64. The lectures showed a speedup of almost 4, so this threw me off a bit. I read on Piazza that most of the student’s test produces figures similar to mine, so it may just be that the simulation that was used in the lectures was different from the simulation we ran.
   2. Speedup for a small array size (1024) was 2.49. The speedup slowly increases as the size of the input array increases to 65536, which is where the speedup peaks. As the input array size increases beyond 65536, the speedup declines and appears to level off at around 2.43.
   3. The difference in speedup between the lowest and highest is .21, which is not that impactful unless you are working with a ton of data. Given that SIMD is “free” performance that takes minimal time to implement, I think you should always use SIMD if your calculations allow. Even if the speedup decrease as the dataset grows larger, you will still benefit from using SIMD.
3. Since there is a little overhead of calling the function to multiply and sum the two array values using SIMD, I suspect that is why there is a reduced speedup when arrays are small. There is less time to “catch up” when arrays are small.

We have learned that performance begins to decrease as the dataset size grows large, due to getting values from memory off the chip. We are seeing this in the graph above. Once the array size grows to be over one million, the performance (speedup) drops.

1. Extra Credit

|  
|  
|  
|  
|  
|  
|  
V

|  
|  
|  
|  
|  
|  
V

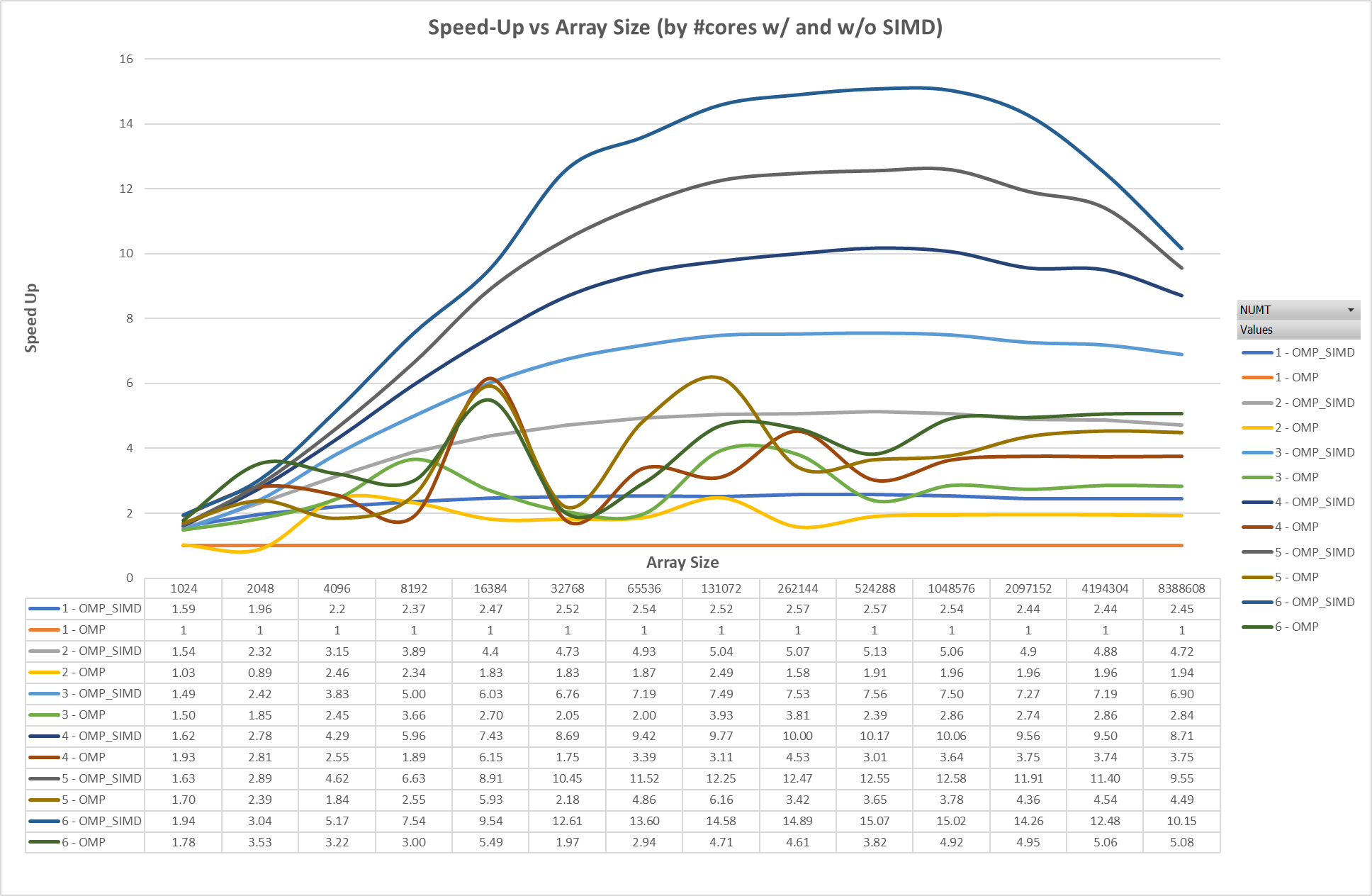
|  
|  
|  
|  
|  
|  
|  
V

|  
|  
|  
|  
|  
|  
|  
V

|  
|  
|  
|  
|  
|  
|

V





Wow! Look at that speedup I got when using OMP and SIMD. I think it is amazing

As expected, when OMP was coupled with SIMD, the speedup what much greater. If you take a look at the graph in problem 3 above, you will see that we got a speedup of around 2.5 (estimated average overall). This speedup was from using SIMD alone, without OMP. So, it makes sense that when OMP and SIMD are used together, one can estimate that the likely speedup will be 2.5 times the speedup seen when using OMP alone. The graph above follows this logic. (roughly) Look at the curves representing 6 cores. When the array size is 262,144 the speedup of OMP alone is 4.61. When SIMD is added to the mix, the speedup for the same array size is 14.89. The speedup gained by utilizing SIMD is actually closer to 3 than the 2.5 estimated, but this was at the peak performance. As the array size increases, the performance gained over OMP alone nears 2.5.

You will notice that speedup for the SIMD w/ OMP curves peak when the array size is about 524,288. After this, when the arrays grow larger, it is apparent that the CPU has a difficult time retrieving the values from memory quick enough to keep up with the calculations. This becomes more and more evident as the speedup increases. If you look at the curve for 6 cores using OMP and SIMD, you can see the dramatic drop in speedup right after the array size of one million. It looks like speedups of around 5 do not cause this slowdown seen with the higher speedups.

The curves for the OMP only runs has a lot of sporadic behavior. I have no idea why the lines are so, seemingly, random. My guess is my computer had a background process that interfered with scheduling, which is why the speedups are all over the place. Interestingly though, the OMP w/ SIMD curves do not seem to be impacted.