CS475

Hsiang Lo

**Project #7B**

**Autocorrelation using CPU OpenMP, CPU SIMD, and OpenCL**

**Goal**

Implement OpenMP, SIMD and OpenCL/CUDA on a GPU to compare performance and comment on relative performance and why it’s that way.

**Procedures**

1. Read the signal file – First line is SIZE
2. Do OpenMP Parallelism for 1 thread //Outer for-loop get pragma
3. Do OpenMP Parallelism for N thread
4. Do SIMD Parallelism //address
5. Do GPU Parallelism
6. Scatterplot the autocorrelation Sum vs the shift
7. Record data and answer questions
8. Draw a bar chart for all of them, pick appropriate unit
9. Answer Commentary

**Notes**

* Autocorrelation/Serial Correlation is the correlation of a signal with a delayed copy of itself as a function of delay.
* Shift before multiplying will result is a much smaller sum.
* Hint: Sine wave hidden in the pattern
* The presence of that harmonic (sine wave) will make the array all positive then all negatives and so on.
* Make only one scatterplot Sum[1]…Sum[512]

**Commentary**

1. **What machines you ran this on**

For this assignment, I ran OpenMP with 1 thread, OpenMP with N thread and SIMD on Flip. For OpenCL, I ran the test on Rabbit in a scheduled time slot for optimal performance. All four tests were ran on flip/rabbit on my Macbook Pro 2018.

1. **Show the Sums{1] ... Sums[512] vs. shift scatterplot**

(Scatterplot the autocorrelation Sums[\*] vs. the shift. Even though there will be Size Sums[\*] values, only scatterplot Sums[1] ... Sums[512].

(I had forgot to include axis label and title, they were later added in editor)

1. **State what the hidden sine-wave period is, i.e., at what multiples of *shift* are you seeing maxima in the graph?**

(7. Tell me what you think the secret sine waves' periods are, i.e., what multiples of *shift* give you maxima in the Sums[\*] scatterplot?)

Given that a sine-wave can be seen from below as having both a positive amplitude and a negative amplitude and reaching where the x coordinate reaching 0 three times in the process.

We can then look back at Q2 of the graph where we can see that the data indicates that the graph reach 0 at it’s x coordinate at around 160, 275 and 375, in the process, it reached the maximum positive amplitude of around 70000 and minimum negative amplitude of -65000.

Given this information, it can be concluded that the Sine Wave’s cycle is around 215.

1. **What patterns are you seeing in the performance bar chart? Which of the four tests runs fastest, next fastest, etc.? By a little, or by a lot?**

It can be seen from the side-by-side comparison in the bar chart that it’s obvious OpenCL is by far the fastest in the comparison of the four tests. It is followed by SIMD at 2nd but then even, the performance is quite below the what OpenCL’s performance. Following SIMD is OpenMP with N thread where I have implemented it with 32 threads. The performance is just slightly worse than SIMD, they are roughly equivalent in their level of performance although the performance differs. Lastly, OpenMP with 1 thread have the worst performance at 4th. It’s performance is practically 0 when compared to the others. In fact, all of them seems to have practically 0 performance when compared to OpenCL.

1. (Draw a bar chart showing the performance for your 1-thread OpenMP, your n-thread OpenMP, your SIMD, and your {OpenCL or CUDA}.)­­
2. **Why do you think the performances work this way?**

For openMP, at a thread count of 1 have the lowest performance of the four tests, and this is due to the fact that OpenMP does not reach its potential and decent speedup until it reaches a thread count of 16/32. So when we are using only 1 thread in this case, we were not able to fully utilize the available resource (threads). As observed, when we increased the thread counts to 16/32, the performance increased dramatically.

SIMD’s performance was by far superior than OpenMP at lower count. The performance definitely increased, and this is due to the fact we are utilizing assembly code for SIMD processing. Due to the lower-level utilization, the processor was able to compute at a much faster rate as result. Furthermore, SIMD is capable of operating the simple addition and multiplication operations and combining the results in such a way that dramatically increased performance as well. I believe this is also because the sum is stored in memory so its easier/faster to fetch.

OpenCL was by far the best of the four tests computed. This is due to the fact that OpenCL is capable of utilize GPU to compute large quantity of data parallelism or data processing. And due to the fact that we are breaking down the already simple command/operations, the GPU was able to quickly process these data/task. Although it should be noted that this worked in our assignment due to large amount of data parallelism and different situation call for different method. ­­