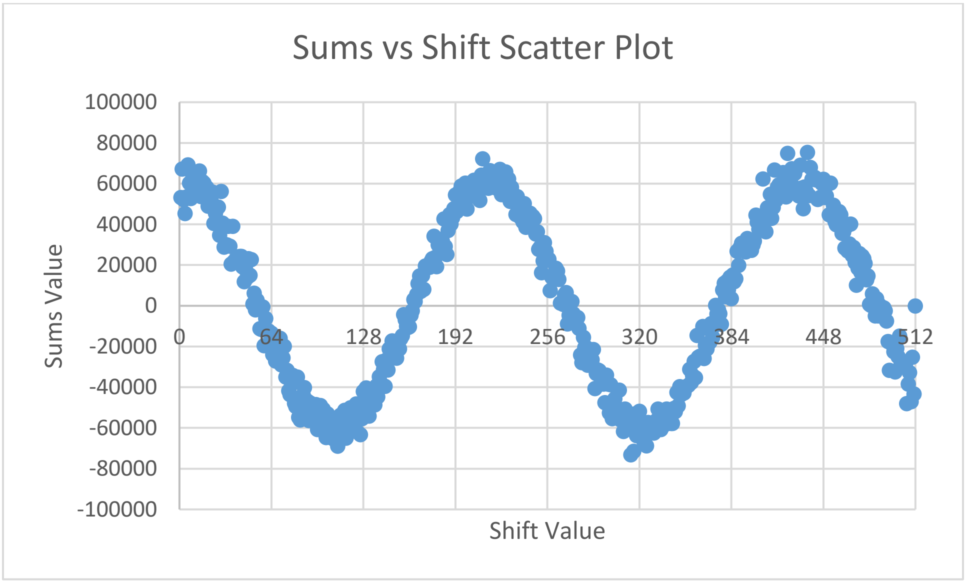
Jesse Thoren

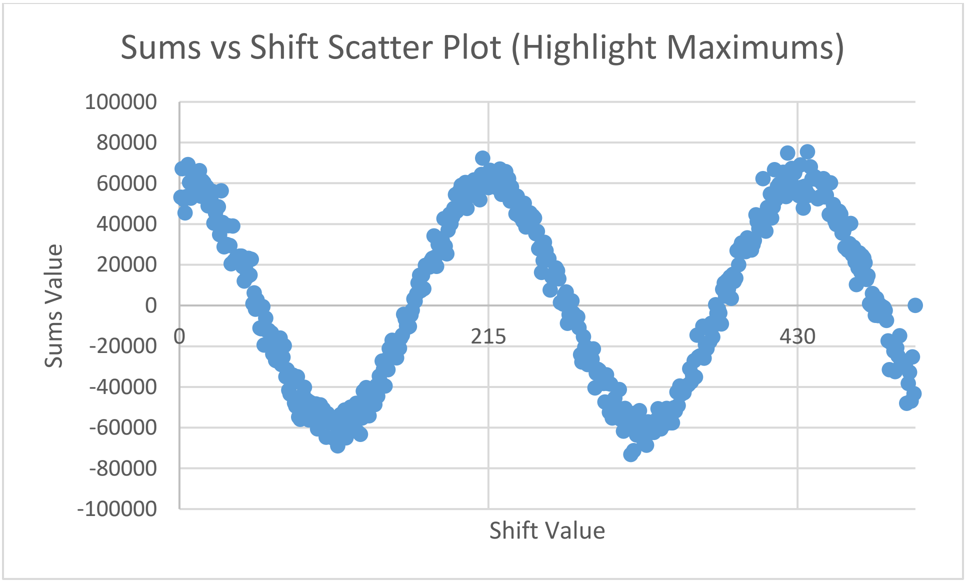
June 11, 2017

CS 475 – Parallel Programming

Project 7 – Autocorrelation using OpenMP, SIMD, and OpenCL

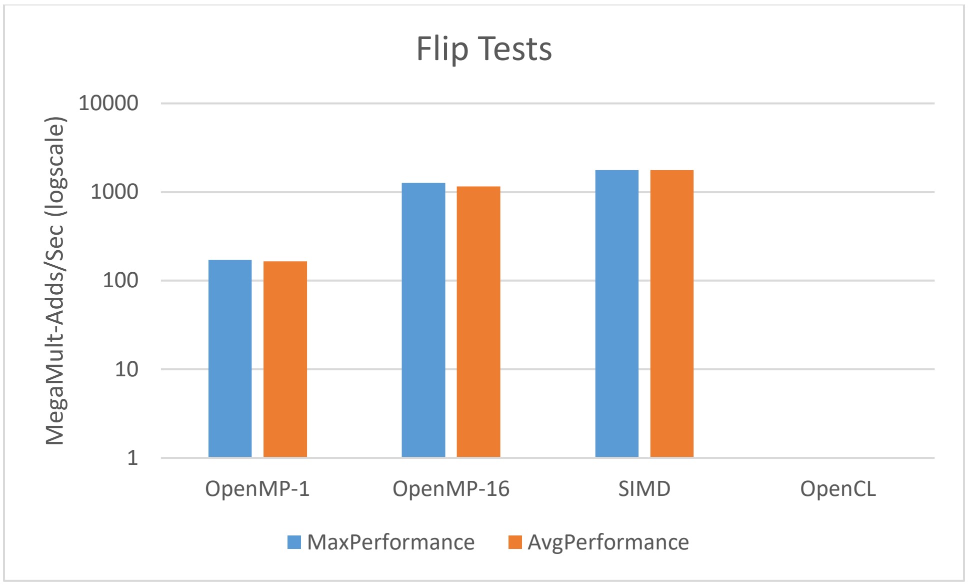
1. I ran the OpenMP test and the SIMD test on flip, and reran the OpenMP test along with the OpenCL test on rabbit so all the sets of data might be somewhat comparable.
2. Below are 2 copies of the Sums vs shift scatterplot, first with a reasonable axis, and secondly to highlight the maximums of the periodic nature of the graph.

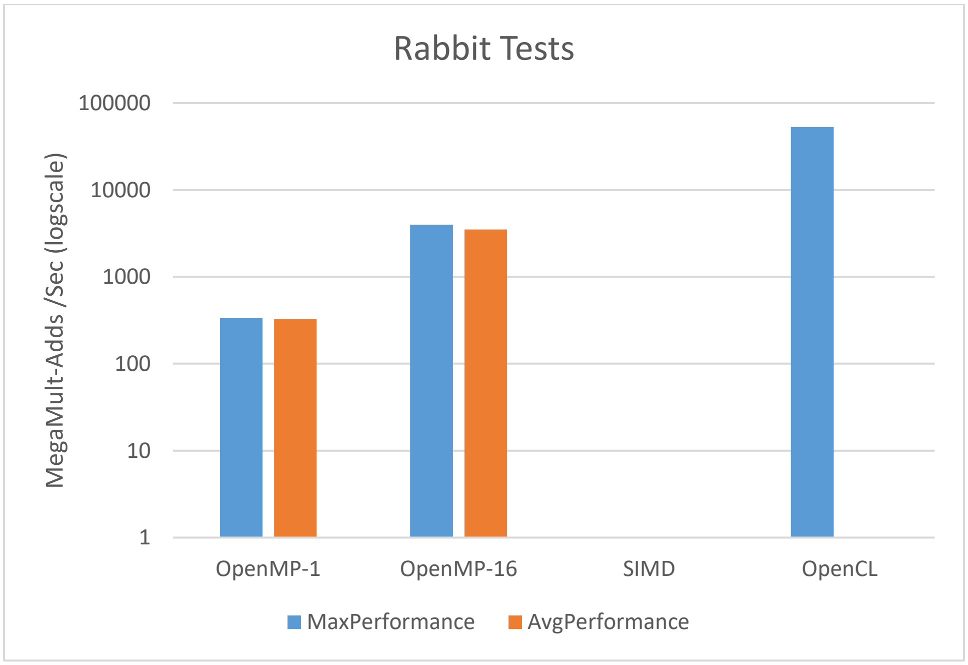


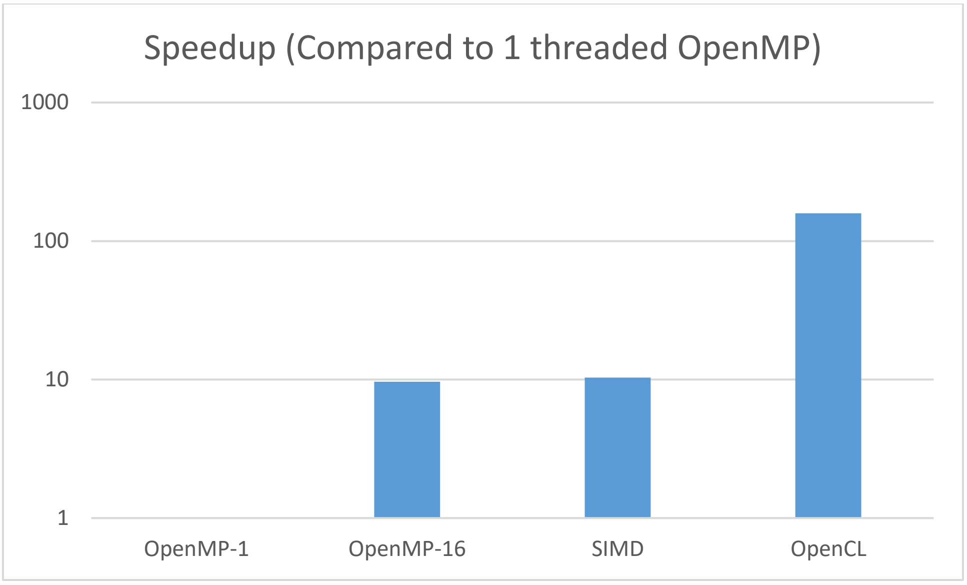


3. As displayed in the 2nd graph in question 2, it appears that the period of the hidden sine-wave is just around 215.

4. Included below are my performance bar charts:







4. (cont) The one threaded OpenMP implementation is clearly the slowest of the 4 implementations.
   OpenMP with 16 threads and SIMD are similar in performance (and an order of magnitude greater performance than 1 threaded OpenMP), but SIMD demonstrated a narrow edge over the 16 threaded OpenMP implementation.
   OpenCL demonstrated an additional order of magnitude of performance over SIMD, and is by far the fastest implementation of the 4.

5. It's clear that OpenMP-1 thread is the clear worst performance wise because there is no attempted parallelism. The single thread does all the work.
   The OpenMP-16 thread implementation is considerably better than the 1 thread implementation, but is not a perfect 16x speedup. This is possibly happening due to variances in cpu load on Flip/Rabbit, or because there is not a dedicated cpu core to running each of the 16 threads, so there is likely additional non-parallizable sections of the program which are preventing perfect parallization.
   SIMD is generating a significant speedup on Flip as a result of using assembly to combine multiple array values into an extended value, multiply 2 of these extended values, and then break them back up to find the answers to multiple multiplications. This is utilizing the fact that the cpu is capable of handling longer data types than a single float, so it's smashing multiple floats together in such a way that a single multiplication operation can complete many multiplications at once.
   OpenCL provides the fastest speedup by far, because it leverages a GPU to grind a number of multiplication-adds at a very rapid speed. Because we're dealing with a single action (multiply/add) a ton of times, the GPU's cuda cores are adept at quickly processing the task.