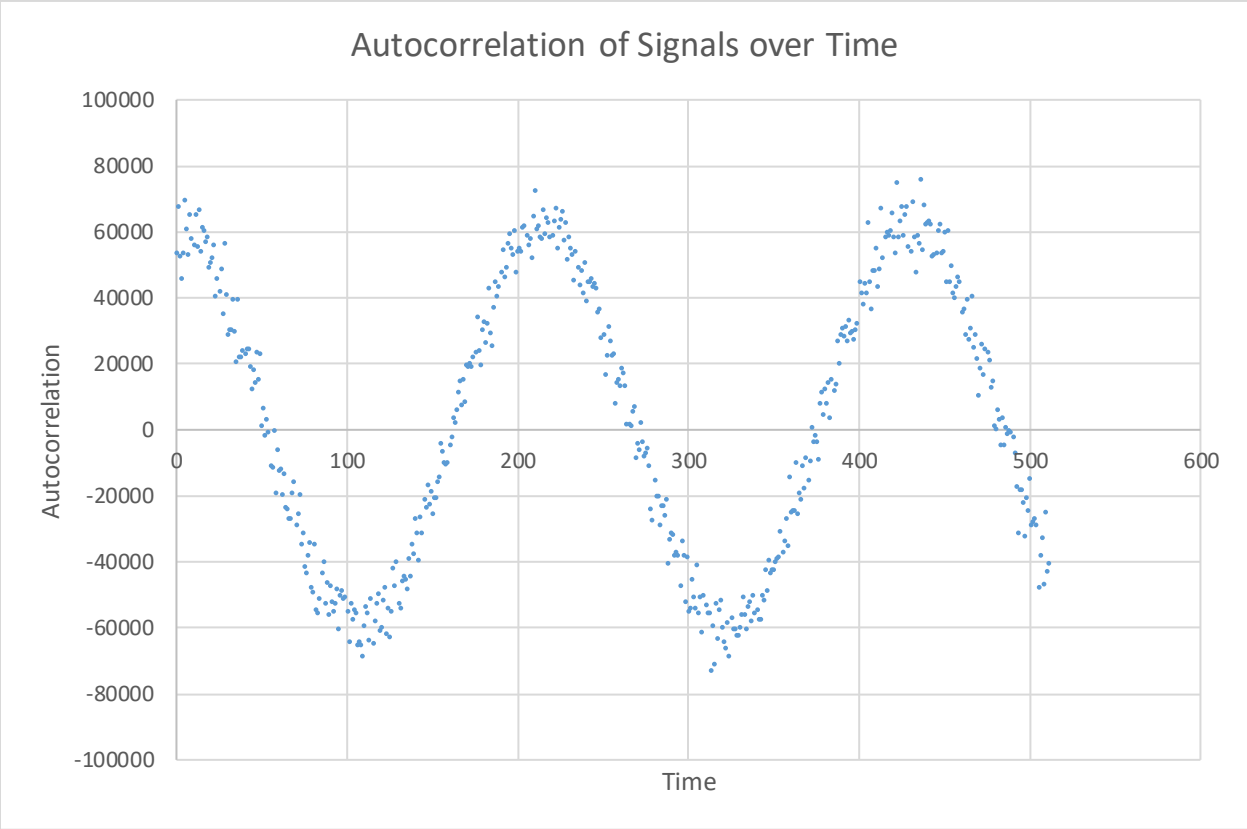
Project 7B: Autocorrelation using CPU OpenMP, CPU SIMD, and GPU OpenCL

Erik Heaney

June 13, 2017

Results

Autocorrelation Sums



Estimated sin wave period:

$$100 * \frac{2\pi}{3} \leq \theta \leq 100 * \frac{3\pi}{4}$$

Analysis

OpenMP Machine Information

Manufacturer:	ASUS
Operating System:	Windows 10 Pro
Processor:	Intel® Core™ i7-3517U CPU @ 1.90 GHz 2.40GHz
RAM:	10.0 GB
System Type:	64-bit operating system, x64-based processor

SIMD SSE CPU Information

Architecture:	x86_64
CPU op-mode(s):	32-bit, 64-bit
Byte Order:	Little Endian
CPU(s):	24
On-line CPU(s) list:	0-23
Thread(s) per core:	2
Core(s) per socket:	6
Socket(s):	2
NUMA node(s):	2
Vendor ID:	GenuineIntel
CPU family:	6
Model:	44
Model name:	Intel(R) Xeon(R) CPU X5650 @ 2.67GHz
CPU MHz:	2660.005
BogoMIPS:	5319.80
Virtualization:	VT-x
L1d cache:	32K
L1i cache:	32K
L2 cache:	256K
L3 cache:	12288K
NUMA node0 CPU(s):	0,2,4,6,8,10,12,14,16,18,20,22
NUMA node1 CPU(s):	1,3,5,7,9,11,13,15,17,19,21,23

OpenCL CPU Information

mic0 (info):

Device Series: ..... Intel(R) Xeon Phi(TM) coprocessor x100 family

Device ID: 0x225e

Number of Cores: 57

OS Version: ..... 2.6.38.8+mpss3.4.2

Flash Version: 2.1.02.0390

Driver Version: ..... 3.4.2-1 (root@rabbit.engr.oregonstate.edu)

Stepping: 0x3

Substepping: 0x0

mic0 (temp):

Cpu Temp: ..... 47.00 C

Memory Temp: ..... 35.00 C

Fan-In Temp: ..... 30.00 C

Fan-Out Temp: ..... 35.00 C

Core Rail Temp: ..... 35.00 C

Uncore Rail Temp: ..... 36.00 C

Memory Rail Temp: ..... 36.00 C

mic0 (freq):

Core Frequency: ..... 1.10 GHz

Total Power: ..... 39.00 Watts

Low Power Limit: ..... 283.00 Watts

High Power Limit: ..... 337.00 Watts

Physical Power Limit: .... 357.00 Watts

mic0 (mem):

Free Memory: ..... 7442.63 MB

Total Memory: ..... 7698.83 MB

Memory Usage: ..... 256.20 MB

mic0 (cores):

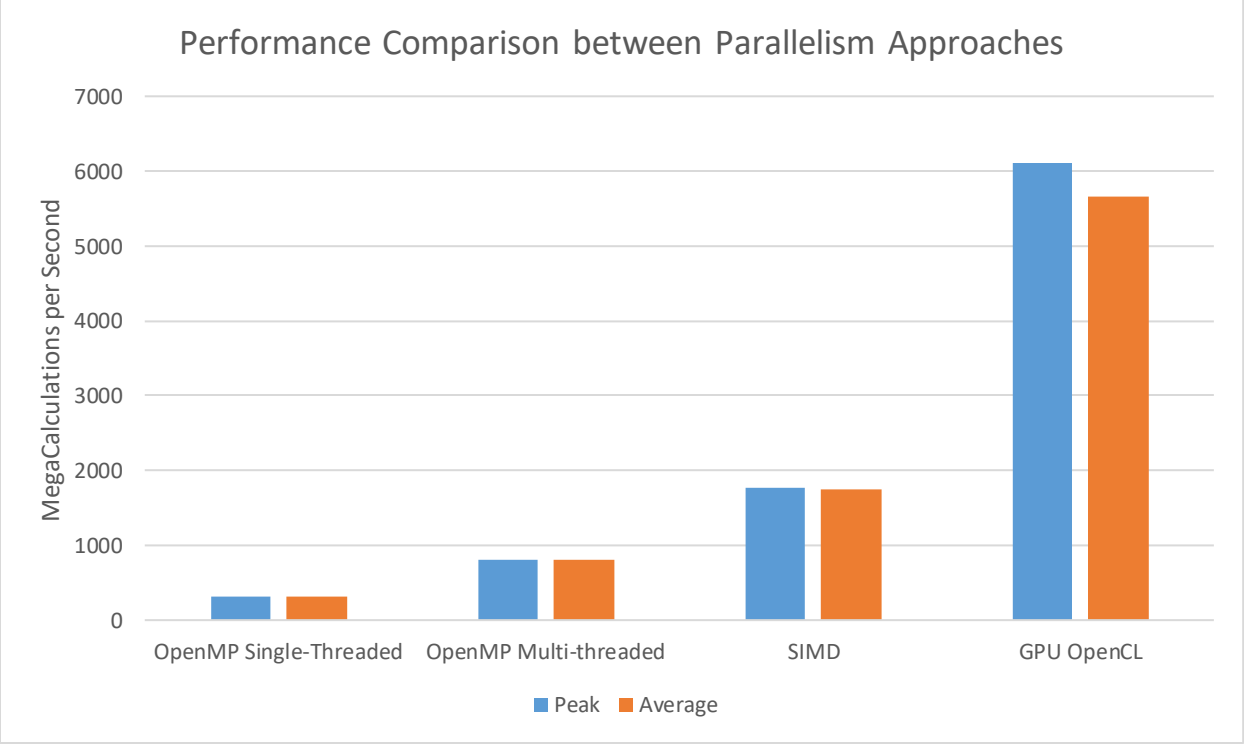
Device Utilization: User: 0.00%, System: 0.03%, Idle: 99.97%

OpenCL Compute Device (GPU) Information

Name = 'NVIDIA CUDA'
Vendor = 'NVIDIA Corporation'
Version = 'OpenCL 1.2 CUDA 8.0.13'
Profile = 'FULL_PROFILE'
Number of Devices = 1
Device #0:
Type = 0x0004 = CL_DEVICE_TYPE_GPU
Device Vendor ID = 0x10de (NVIDIA)
Device Maximum Compute Units = 15
Device Maximum Work Item Dimensions = 3
Device Maximum Work Item Sizes = 1024 x 1024 x 64
Device Maximum Work Group Size = 1024
Device Maximum Clock Frequency = 1071 MHz

Performance Comparison

	OpenMP Single-Threaded	OpenMP Multi-threaded	SIMD	GPU OpenCL
Peak	310.84	814.48	1770.96	6100
Average	309.3	798.73	1749.31	5650



Methodology

Four test programs were created to perform a performance comparison of parallelism strategies. The single-threaded OpenMP program served as a scientific control, while the multi-threaded OpenMP, Intel SSE SIMD, and the OpenCL with a GPU served as independent variables. Each of the four programs performed an autocorrelation calculation on every datum within a data set of size 32,768. Autocorrelation is a statistical method of measuring how much a random data set can be expressed as a wide-sense stationary random process, or in other words, if the data have periodicity. It requires calculating the correlation between the variance in the dependent variable and the independent variable given a time difference. Since each dependent variable within the data set needed to be correlated with every possible discrete time difference, a total of 32,768 squared values, or 1,073,741,824 total multiplication and add operations. Thus, autocorrelation on a large data set was both a realistic operation to be performed by a computer scientist that could also provide an accurate measure of parallelism performance.

The OpenMP solution used a simple pragma statement before the primary for loop with eight threads. The SIMD solution used the Streaming SIMD Extensions (SSE), an Intel instruction set extension for x86 architectures, on the Oregon State University's Flip server. Finally, the OpenCL solution used NVidia Titan Black GPU as the compute device with a local group size of 256. The program was executed on the Oregon State University's Rabbit server. The results were collected in separate CSV files and the graphs were produced using Microsoft Excel.

Discussion

The performance results of this experiment demonstrate the varying levels of parallelism capable across different computing strategies. The OpenMP multi-threaded solution experienced a 2.6 speed up in performance. Since eight threads were used, this means the multi-threaded solution returned a parallel fraction of .32. This relatively low parallelizability may be explained by a few factors. First, the large amount of calculations could lead to temporal incoherence. The pre-fetching distance of the CPU's L2 cache may result in performance degradation as the program executes. Furthermore, the thread count was selected arbitrarily. In further experiments, it would be advisable to perform a set of tests measuring the performance results across a set of thread counts, so that the highest performing result could be used for cross-strategy comparison. Nevertheless, it is this author's assumption that an improvement in parallelizability via OpenMP would likely be marginal, and that this experiment provides sufficient evidence.

The SSE SIMD solution experienced a 5.7 speed up in performance in comparison to the single-threaded solution, and a 2.17 speed up in comparison to the multi-thread solution. This increase in performance can be explained by the test program's 'embarrassingly parallelism'. Since the data set is large, it lends itself to data parallelism. Furthermore, since the Intel SSE instruction set provides a fused multiply-add (FMA) operation, this increases CPU-level arithmetic optimization. Since a CPU can perform one FMA operation in one cycle asymptotically, or two operations within two cycles, while traditional multiply and add operations would take two separate cycles, this causes a doubling in performance. The SIMD solution could possibly achieve greater performance results by adjusting the pre-fetching distance in order to achieve greater temporal coherence, although again it may be assumed that the performance-enhancing effects would likely be marginal.

Finally, the OpenCL solution experienced a speed up of 19.7. This impressive performance increase can be explained by two factors: the embarrassing parallelism of the test program and the

NVidia Titan Black's power difference. As mentioned above, autocorrelation lends itself to data parallelism. A GPU's architecture, with its many hundreds of arithmetic units, is especially designed to perform data parallelism on large data sets. Furthermore, the GTX Titan Black is a powerful, enthusiast-level GeForce 7000 GPU with 2880 CUDA cores. Although comparisons between CPUs and GPUs are notoriously fraught, it would be inappropriate to perform this cross-strategy comparison without exploring the hardware differences between the three tests. Simply put, the Titan Black is a more advanced piece of hardware than the Intel® Core™ i7 CPU operating on this author's laptop. However, it is likely that similar performance results would be achieved with a weaker graphics card. In a more thorough experiment, performance would be measured across a variety of graphics cards, alongside a variety of local group sizes. Uncontrolled variability aside, this author argues that for identical operations performed on a large data set, an OpenCL solution using a GPU as the compute device yields the greatest performance results.