High Performance Computing
Anthony Trubiano
Assignment 4 (due April 15, 2019)

Note: All CPU computations were performed with an Intel Core i7-8750H Processor with base frequency 2.20 GHz. The maximum main memory bandwidth is 41.8 GB/s. At 16 double precision operations per cycle, the theoretical max flop rate would be about 35.2 GFlops/s. It has 6 cores and can reach 12 threads through hyper-threading.

1. **Matrix Vector Multiplication on a GPU** We perform a matrix vector multiplication both on a CPU and GPU and compare the bandwidth achieved on each. We compare CPU and GPU results to check for errors. We use a vector size of $N = 16384$. The CPU implementation uses OpenMP on my machine with 8 threads.

Machine	Bandwidth (GB/s)
CPU	5.5
Cuda 1	25.8
Cuda 2	237
Cuda 3	0.14?
Cuda 4	41.2
Cuda 5	24.5

   Table 1: Bandwidth for a matrix-vector multiply using different GPUs, with a system size of $N = 16384$.

2. **2D Jacobi Method on a GPU** Here we apply Jacobi iteration to solve the 2D Laplace equation on the square with homogeneous Dirichlet boundary conditions both on a CPU and GPU. The residuals are computed from the CPU and GPU to see if we get the same result. We then compare the time each takes to perform 1000 iterations of the method on a 1000×1000 matrix.

Machine	Time (s)
CPU	2.57
Cuda 1	0.2
Cuda 2	0.1
Cuda 3	0.85
Cuda 4	0.16
Cuda 5	0.24

   Table 2: Time taken (in seconds) for the Jacobi method for 1000 iterations on a 1000×1000 system.

   Looking at the results, we see in the best case (CUDA2), a speedup of almost 20 to 50 times the performance on a CPU using OpenMP with 8 threads, which is quite significant. On average over the GPUs, we see a speedup of about 5 to 8 times the CPU performance. I am not sure why, but on CUDA3, my result was incorrect and the computation took longer than the CPU (matrix-vector multiply). This is strange considering my code works fine on the other systems.

3. **Project Proposal** I will be working with Ondrej Maxian and Tristan Goodwill broadly on "Parallel Algorithms for CFD". We will begin by writing a solver for Poisson's Equation on a periodic domain using a parallel FFT. We are currently unsure whether we will write the FFT ourselves. If time permits, we may extend this to a Stokes solver.