Spring 2019: Advanced Topics in Numerical Analysis: High Performance Computing

1. **Matrix-vector operations on a GPU.**
   Matrix dimensions: 250 x 102400, vector dimensions: 102400
   For all GPUs:
   Shared memory per block: 48.000000 KB
   Max threads per block: 1024

   Output:

   (Device 0) Name: GeForce GTX TITAN Black
   Total memory: 6.379143 GB
   Peak Memory Bandwidth (GB/s): 336.000000

   CPU Bandwidth = 2.692747 GB/s
   GPU Bandwidth = 6.058845 GB/s
   err = 0.000000

   (Device 0) Name: GeForce RTX 2080 Ti
   Total memory: 11.523260 GB
   Peak Memory Bandwidth (GB/s): 616.000000

   CPU Bandwidth = 2.536058 GB/s
   GPU Bandwidth = 8.148880 GB/s
   err = 0.000000

   (Device 0) Name: TITAN V
   Total memory: 12.621382 GB
   Peak Memory Bandwidth (GB/s): 652.800000

   CPU Bandwidth = 2.575553 GB/s
   GPU Bandwidth = 8.689153 GB/s
   err = 0.000000

   Note: I got strange errors when I tried running on servers cuda4 and cuda5.

2. **2D Jacobi method on a GPU.**
   For GPU stats see Problem 1. For all executions, $N = 10^4$.

   Device Name: GeForce GTX TITAN Black

   res 0 = 9999.499937

res 5 = 9995.316006
res 10 = 9993.011037
res 15 = 9991.215151
res 20 = 9989.691633

elapsed time: 3.036214s

Device Name: GeForce RTX 2080 Ti

res 0 = 9999.499937
res 5 = 9995.316005
res 10 = 9993.011034
res 15 = 9991.215143
res 20 = 9989.691625

elapsed time: 1.832207s

Device Name: TITAN V

res 0 = 9999.499937
res 5 = 9995.315988
res 10 = 9993.011032
res 15 = 9991.215141
res 20 = 9989.691609

elapsed time: 1.678875s

Note: I got strange errors when I tried running on servers cuda4 and cuda5.

Final Project Proposal: Anya Katsevich and Terrence Alsup

We will work on parallelizing code from an old research project that implements Kinetic Monte Carlo (KMC) in serial. KMC is an algorithm for modelling the dynamics of a continuous time jump Markov process. The process we will consider is atom movement on the surface of a crystal. In 2D we consider a lattice $\Omega = \{1, \dots, L\} \times \{1, \dots, L\}$. At each site $\vec{\ell} \in \Omega$ on the lattice we have atoms stacked on top of each other. The number of atoms at each site $\vec{\ell}$ will be denoted by the height of the stack $h_{\vec{\ell}}$. The surface of the crystal is then just the atoms at the top of each stack. KMC simulates a Markov jump process describing the evolution of the vector $(h_{\vec{\ell}}(t))_{\vec{\ell} \in \Omega}$ by randomly choosing one lattice site, at which the topmost atom will jump to the top of one of the neighboring stacks. The rates of this process are governed by the height difference between two neighboring stacks. In the serial implementation of KMC, only one jump occurs at a time, and all other atoms, even those far away, must wait before getting a chance to move.

To parallelize KMC we will divide the lattice into sections and then independently run Markov jump processes on each section. After every process has run for a while we will need to synchronize the sections since the jump times are random. We will also need to handle potential conflicts on the boundaries of the sections where atoms may be jumping across. This could potentially be handled with a layer of ghost lattice sites. We plan to transfer the computation to a GPU for the parallelization.

For large values of L, this Markov process can be seen as a discretized version of continuous dynamics (described by a PDE) governing the evolution of a smooth surface $h(x,t)$. This PDE is known and solutions to it can be computed numerically. To test our implementation, we can evolve a continuous height profile forward for a certain amount of time using the PDE, evolve a discretization of that profile for the same amount of time using parallel KMC, and compare the two resulting profiles.