Homework 0

James Andrews

1. Bio

I am a first year graduate student attending George Mason University in the Computational Materials Science Center working toward a PhD in computational science. My background is in pure mathematics, but my current interests are in astrophysics and cosmology. I am new to the field of computational science, but immediately I see the necessity for parallelism to solve modern problems. From this class, I am looking forward to exploring a verity of high performance applications, and becoming proficient in coding such applications.

2. Monte Carlo simulations in material science

I will speak on the Metropolis Monte Carlo method with CUDA enabled Graphic Processing Units done by Clifford Hall, Weixiao Ji, and Estela Blaisten-Barojas. [1]

Computer simulations of fluids and solids get very large very quickly, making them ideal candidates for parallelization. Materials scientists want to find properties such as particle distributions, free-energy, and more, but are restricted by computational time because to truly recreate the physical structure of a material, a number of particles of the order $10^{23}$ would need to be simulated. Because this is an unreasonable prospect, there are schemes to simulate large systems with fewer atoms. Even so, systems of even hundreds of particles may still take unreasonable lengths of time without smart programming; hours, days, or much longer for particularly large systems.

The goal of Hall, Ji, and Blaisten was to create a parallel code that implements CUDA GPUs for a Metropolis Monte Carlo (MMC) simulation, a well known algorithm used in many fields of physics, engineering, and statistics. The challenge of designing a parallel program for a MMC simulation is that they are inherently sequential. What their program does is use an Adaptive Tempering Monte Carlo (ATMC) that utilizes CUDA processors asynchronously, allowing for parallelization. They used pyrrole monomers for their structure with data results within 5% of expected outcomes.

The work was done in CUDA C and compiled by NVCC, CUDA version 4.0 C compiler on a workstation with a quad of Intel i5 2400S, 2.5 GHz cores and one Tesla C2070 GPU equipped with 448 cores and 6 GB of memory.

Their implementation greatly decreased the runtime compared to a serial implementation run only on a CPU. Figure 1 shows the average runtime to cycle through
each particle of their simulation (blue for GPU implementation, red for non-GPU), and the green shows the speed-up factor.

![Figure 1.](image)

As is shown, the simulation scales up very well. There is a speed-up factor of 62.5 for systems with 393,216 particles (342.47 s/iteration compared to 21,392.8 s/iteration). The more particles simulated, the greater the relative speed up. It is easy to imagine that scaling up the number of GPU cores would lead to even more particles that can be simulated. The paper describes using MPI to connect 32 CPU-GPU pairs to achieve a 3 s/iteration runtime for a system of $N = 608,256$ particles.

References

The Metropolis Monte Carlo method with CUDA enabled Graphic Processing Units
Journal of Computational Physics 258 (2014) 871879