CUDA Implementation of Monte Carlo Based Applications in Computational Chemistry

Akila Gothandaraman, Revati Kumar, Dongliang Yang, Kenneth D. Jordan, Gregory D. Peterson

1 Center for Simulation and Modeling (SAM), University of Pittsburgh
2 Department of Chemistry, University of Pittsburgh
3 Department of Electrical Engineering and Computer Science, University of Tennessee, Knoxville

Abstract

Architectures such as multi-core processors, Reconfigurable Computing (RC) using Field-Programmable Gate Arrays (FPGAs), and Graphics Processing Units (GPUs) have emerged as alternatives in the landscape of high-performance computing (HPC). In our research, we are particularly interested in leveraging GPUs to accelerate classical and quantum Monte Carlo simulations in computational chemistry. GPUs have undergone a tremendous growth due to the ever-growing performance demands from the gaming industry. NVIDIA’s GPU solutions presently provide hundreds of processing cores and tremendous on-chip memory bandwidth making them attractive for general-purpose computing. The Compute Unified Device Architecture (CUDA) programming paradigm allows us to exploit the computational power of the GPU without the need to invoke graphics functions, alleviating the difficulty of programming GPUs for general-purpose computing.

Algorithm

Monte Carlo methods involve sampling a number of configurations of a collection of particles and averaging the properties over a large number of samples. With increased computational resources, one can simulate larger physical systems for longer times.

We use the Quantum Monte Carlo (Variational Monte Carlo) to obtain the structural and energetic properties a cluster of inert gas atoms.

Algorithm

REPEAT for {i iterations}
Step 1: Select a reference configuration, $R(x, y, z)$ at random.
Step 2: Obtain a new configuration, $R'$ by adding a small random displacement to all particles in the above configuration.
Step 3: Compute the ground-state properties (potential energy, wave function) of the particles in the current configuration, $R''$.
Step 4: Accept or reject the present configuration using the ratio of the wave function values.
UNTIL finished

Implementation

Implementation of the Monte Carlo Applications on NVIDIA GPUs using Compute Unified Device Architecture (CUDA)

Quantum Monte Carlo (QMC)

- $O(N)$ atom positions copied from host memory to device memory using CUDA runtime
- Execution Configuration: 1D grid of blocks and block of threads
- Subset of rows of the matrix (NP) to each thread block
- Each thread calculates the interactions between its atom and all other atoms
- Different levels of memory hierarchy on the GPU (16 KB shared memory per block)
- Final reductions on the CPU to produce $O(N^2)$ potential energy or wave function
- We consider naive (entire matrix evaluation) and optimized (upper/lower triangular matrix evaluation) implementations as well as experiment with entirely single-, entirely double- and mixed-precision (single-precision for function evaluations and double-precision for accumulations) algorithms.

Classical Monte Carlo

- The CUDA kernel uses the same grid setup as the QMC with the following differences,
  - In-place reductions of row-wise function values on GPU to produce $O(N)$
  - Different levels of memory hierarchy on the GPU (16 KB shared memory per block)
  - Final reductions on the CPU to produce $O(N^2)$

Conclusions and Future Work

This work explores NVIDIA GPUs platforms to accelerate the kernels of quantum and classical Monte Carlo applications.

Future work includes the following:
- Optimize the GPU implementation of the kernels of the classical Monte Carlo algorithm to study water
- Experiment with the numerical precisions and choose the algorithm that provides the best performance on the GPU (both speedup and accuracy)
- Extend the GPU implementation from the simple TIP4P model to the DPP many-body force field

References