GPU-Accelerated Variational Monte Carlo

B. Tasseff[1,4], N. Setty[2], N. Sengupta[3], Z. Yun[4], S. Abu Asal[2,4], Y. Fang[2,4], S. Pathak[3], J. Moreno[3,4], J. Ramanujam[4], M. Jarrell[3,4]

[1]Department of Physics, University of Northern Iowa [2]Department of Electrical & Computer Engineering, Louisiana State University
[3]Department of Physics & Astronomy, Louisiana State University [4]Center for Computation & Technology, Louisiana State University

The Big Question

How can we model and describe the behavior of strongly-correlated electronic systems (e.g., high-temperature superconductors) in an accurate, computationally-efficient manner?

The Variational Method

In quantum mechanics, the variational method is a strategy used to find and study the ground state of a system, Ψ_G. The procedure is relatively straightforward:

1. Design ("guess") a trial ground state, $\Psi(\alpha)$.
2. Vary parameter α until $E(\alpha)$ is minimized.
3. Obtain and study Ψ_G.

For a strongly-correlated system, the ground state is often impossible to determine analytically. The variational method allows us to find numerically- and physically-accurate ground states with ease.

Variational Monte Carlo

An electronic system may be modeled as a lattice consisting of atomic *sites* housing spin-up and spin-down electrons. The configuration of electrons within the lattice implicitly defines the energy of the system.

- **The Connection**: For every value α_i, there exists *one* configuration with minimal energy, $E_m(\alpha_i)$. The value α_G that *best* minimizes this energy is used to obtain Ψ_G.
- **The Problem**: A 100-site lattice with 50 $\uparrow$ and 50 $\downarrow$ electrons will have nearly 10^{60} possible configurations. It would be difficult to find *one* configuration with minimal energy $E_m(\alpha)$.
- **The Solution**: Randomly sample configurations *near* each minimal energy. Average these results to predict each minimal energy. Repeat while varying α to obtain α_G, E_G, and Ψ_G.

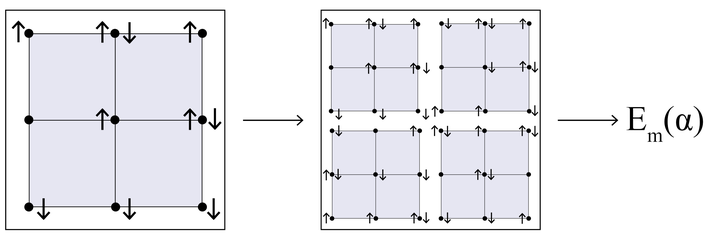

Figure 1: A strongly-correlated electronic system may be modeled as a latticed configuration of interacting electrons. By randomly sampling many configurations, we can approximate $E_m(\alpha)$, the minimal energy for a particular parameter value.

Serial Implementation

The serial (CPU) version of the procedure builds upon concepts described in the preceding section:

1. Begin with a random initial configuration and state $\Psi(\alpha)$.
2. Choose an electron at random and move it, producing $\Psi'(\alpha)$.
   - If $|\Psi'(\alpha)|^2/|\Psi(\alpha)|^2 >$ some randomized probability, compute and store the energy of the configuration.
   - Otherwise, keep the old configuration and repeat 2.
3. Repeat 2 and 3 until reaching an equilibrium energy $E_m(\alpha)$.

This process is then repeated for many α_i. The parameter value which best minimizes E_m is then used to build Ψ_G.

GPU Implementation

The serial procedure uses a single *Markov process*. Running such processes many times allows us to sample and compare more configurations, entailing higher-precision $E_m(\alpha_i)$. This, in turn, allows for the more accurate characterization of the ground state, Ψ_G.

In our research, **graphics processing units** (GPUs) are used to run multiple Markov processes in parallel. The GPU implementation is intended to provide a substantial, cost-effective speedup when compared to the serial counterpart.

Validation

After building the GPU implementation, we first ensured it worked comparably to the CPU version. We also observed the increase in data quality obtained when using multiple Markov processes.

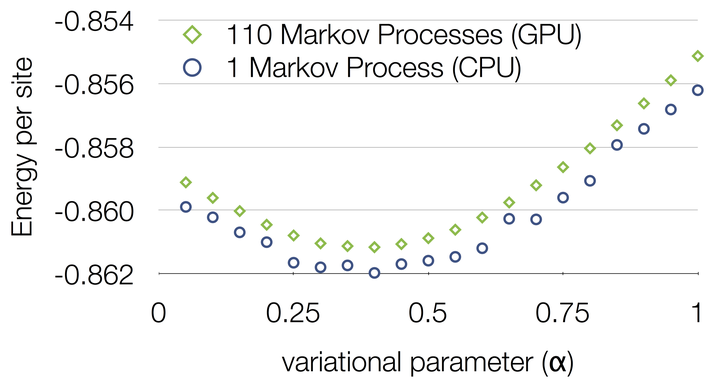

Figure 2: Comparison of variational results obtained for parameter α using a single Markov process (CPU) and 110 Markov processes (GPU).

Results

Finally, to adequately benchmark the GPU implementation, we compared the speeds of CPU and GPU versions of the code.

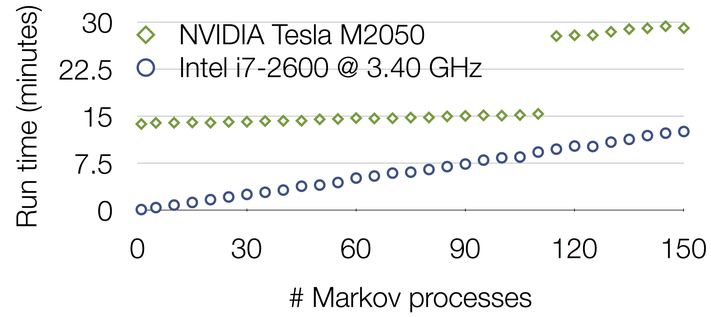

Figure 3: Performance comparison of CPU and GPU variational Monte Carlo implementations executing multiple Markov processes.

Discussion

Upon inspection of figure 2, there is a clear increase in data quality when executing multiple Markov processes. However, as shown in figure 3, the current GPU code does not provide a speedup over the CPU code. Nevertheless, the CPU code scales at a much quicker rate than the GPU counterpart.

Conclusions and Outlook

Although we find benefit in executing multiple Markov processes, the current GPU code does not provide improved performance over its CPU counterpart. However, many improvements remain to be made to the foundational GPU code; such optimization may eventually permit a significant speedup over the serial CPU code.

References

[1] V. B. Shenoy, *Variational Monte Carlo Method for Fermions*, Indian Institute of Technology, Bangalore, India, 2009.

[2] S. Pathak et al., Phys. Rev. Lett. **102** (2009), 027002.

Acknowledgements

This material is based upon work supported by the Louisiana Board of Regents contract number LEQSF(2007-12)-ENH-PKSFI-PRS-01 with additional support from the Center for Computation & Technology at Louisiana State University.