Fully accelerating quantum Monte Carlo simulations of real materials on GPU clusters

Kenneth P. Esler[1], Jeongnim Kim[1], and David M. Ceperley[1,2]
[1]*National Center for Supercomputing Applications,* [2]*Department of Physics*
University of Illinois at Urbana-Champaign
Urbana, IL, USA
esler@uiuc.edu, jnkim@ncsa.uiuc.edu, ceperley@ncsa.uiuc.edu

***Abstract*—Continuum quantum Monte Carlo (QMC) has proved to be an invaluable tool for predicting the properties of matter from fundamental principles. By solving the many-body Schrödinger equation through a stochastic projection, it achieves greater accuracy than mean-field methods and better scalability than quantum chemical methods, enabling scientific discovery across a broad spectrum of disciplines. The multiple forms of parallelism afforded by QMC algorithms make them ideal candidates for acceleration in the many-core paradigm. We present the results of our effort to port the QMCPACK simulation code to the NVIDIA CUDA GPU platform. We restructure the CPU algorithms to expose additional parallelism, minimize GPU-CPU communication, and efficiently utilize the GPU memory hierarchy. We employ mixed precision on GT200 GPUs and MPI for intercommunication and load balancing. In production-level science runs, we observe typical full-application speedups of approximately 10x to 15x relative to quad-core Xeon CPUs alone, while reproducing the double-precision CPU results within statistical error.**

***Keywords*-QMCPACK; quantum Monte Carlo; GPU**

I. Introduction

One of the great scientific triumphs of the last century was the development of quantum theory, in particular the formulation of the Schrödinger equation and its relativistic counterpart, the Dirac equation. In principle, these equations allow the theoretical prediction of all the physical and chemical properties of matter. The direct application of these equations, however, result in computation that scales exponentially with the number of particles, allowing exact solutions for only small isolated molecules containing a few electrons.

Nonetheless, numerous methods have been developed in proceeding decades which offer increasingly accurate *approximate* solutions to the Schrödinger equation. One of the most promising class of methods, known collectively as *quantum Monte Carlo*(QMC), recasts this equation in integral form, which admits solution through stochastic sampling. This allows the properties of molecules and materials to be computed with very high accuracy with a computational load which scales roughly as N^3, where N is the number of electrons. In contrast, the commonly used *coupled-cluster* method of quantum chemistry, of comparable accuracy, typically scales as N^6 to N^7. Most methods based on density functional theory have the same N^3 complexity as QMC, but with a much smaller prefactor. These methods, however, typically produce significantly less reliable results than QMC.

A. *Quantum Monte Carlo on GPUs*

Quantum Monte Carlo simulations currently consume many millions of CPU hours annually on leadership computing facilities. While this usage is justified by the predictive capability of the method, there is strong motivation to reduce cost. Furthermore, the capacity of current leadership facilities still falls short of the requirements for QMC simulations of important classes of materials, ranging from biomolecules to disordered solids. Recently, the use of graphics processing units for scientific computation has provided a path to reduce the computational cost for algorithms amenable to large-scale parallelization. Fortunately, QMC algorithms afford several levels of parallelism, including, but not limited to the natural parallelism of Monte Carlo methods.

As a result of these considerations, we have ported a large subset of QMCPACK [1], our open-source QMC simulation suite, to the NVIDIA CUDA platform. In particular, we have implemented all the routines necessary to simulate systems of electrons and ions in periodic or open boundary conditions, including molecules, solids and liquids. We have tested it for systems of up to 128 atoms and 512 electrons.

B. *QMC algorithms*

Continuum Monte Carlo methods compute the properties of many-body systems of interacting quantum particles. In this paper, we focus on systems containing electrons and ions and the zero-temperature QMC methods known as *variational Monte Carlo*(VMC) and *diffusion Monte Carlo*(DMC). The goal of these methods is to find the lowest energy solutions to the Schrödinger equation,

$$\hat{\mathcal{H}}\psi\left(\mathbf{r}_1 \dots \mathbf{r}_N\right) = E\psi\left(\mathbf{r}_1 \dots \mathbf{r}_N\right), \tag{1}$$

where $\hat{\mathcal{H}}$ is the *Hamiltonian* total-energy operator given by

$$\hat{\mathcal{H}} = -\sum_i \frac{\hbar^2}{2m}\nabla_i^2 + \sum_{j\neq i}\frac{1}{|\mathbf{r}_i - \mathbf{r}_j|} + \sum_{i,k}\hat{V}_{\text{NL}}(\mathbf{r}_i - \mathbf{I}_k). \tag{2}$$

In VMC, a *trial wave function*, $\psi_T(\mathbf{R}; \{p_i\})$, is written in analytic form. Metropolis Monte Carlo is used to generate samples, termed *walkers*, with a distribution proportional to $|\psi_T|^2$, over which the total energy is averaged. The parameters are then adjusted to minimize the energy, yielding the best estimate for the lowest-energy eigenstate. DMC simulations begins with the optimized ψ_T, and repeatedly applies a projection operator that maps to a stochastic process of *drifting*, *diffusing*, and *branching*. In the branching phase, walkers are replicated or destroyed based on the local value of the total energy operator, leading to a fluctuating walker population and a potential load imbalance. Details of these methods are available in review. [2]

II. Overview of GPU implementation

While our initial GPU experimentation focused on the most expensive kernels, we found that any computation left on the CPU became a performance bottleneck. Hence, we ported essentially all the computational elements required for the simulation of systems containing electrons an ions, resulting in over 15,000 total lines of kernel code in about 100 kernels, albeit some from partially replicated code in functional specializations.

In the CPU version of QMCPACK, we propagate each electron in a walker before proceeding to the next walker, in order to achieve high cache reuse. In the GPU implementation, we restructure all our algorithms so that all walkers are propagated in parallel, in order to expose a sufficient number of threads for efficient operation. We reorganize and pad our data structures to allow coalesced memory reads and good use of the on-chip shared memory. In addition, we use a mixed precision scheme to achieve high performance while retaining the required accuracy. We also very careful to minimize traffic over the relatively slow PCI-E bus connecting the CPU and GPU by retaining essentially all the data needed for the computation in GPU memory. In particular, between each transfer of $\mathcal{O}(N^{\alpha})$ data, we ensure that we perform at least $\mathcal{O}(N^{\alpha+1})$ work.

To allow scaling to large clusters of GPUs, we utilize MPI for inter-node communication, i.e. for distributing input data, for collecting computed quantities, and for load-balancing. In the DMC algorithm, the branching of walkers leads to a dynamically fluctuating amount of work on each GPU. By migrating walkers between nodes with point-to-point communication, this load imbalance is eliminated. Care is taken to aggregate all the data required for a walker into a single anonymous buffer so that the number of messages across the PCI bus and interconnect fabric is minimized.

Kernel classes

The computational kernels in QMCPACK can be roughly divided into those that evaluate the trial wave function and its derivatives, and those that compute the potential energy. The trial wave function we employ is written as a product of two determinants and a *Jastrow* correlation factor, which resembles an exponentiated sum of pair potentials. The main classes of wave function routines are then: 1) the evaluation of value, gradients, and Laplacians of the electronic orbitals, which are represented in a 3D cubic B-spline basis; 2) the computation of determinant ratios by dotting orbital values and derives with appropriate columns of the determinant inverse matrices; 3) the rank-1 updates of the inverse matrices for accepted moves; 4) the evaluation of the one-body and two-body Jastrow correlation functions.

The representation of the atomic orbitals in the B-spline basis requires significant storage, which limits system size, particularly on GPUs with a maximum of 4 GB of RAM. To allow larger systems to be studied, we developed a mixed-basis representation which combines an expansion in spherical harmonics around atoms with 3D B-splines in the interstitial region, for savings of approximately 10x in the memory requirement.

The potential energy is comprised of: 5) the Coulomb interaction between electrons; 6) the pseudopotential interaction between electrons and ions. In periodic boundary conditions (e.g. for bulk solids and liquids), 5) involves a summation in real space, performed with hardware texture interpolation, and another in Fourier space, while for molecules only the real space sum is required. The pseudopotential interaction involves local part similar to 5) in addition to a nonlocal projection of the wavefunctions onto spherical harmonics, evaluated by computing wave function ratios on a spherical quadrature grid surrounding atoms. The latter ratio evaluations involve kernels similar to 1), 2), and 4).

III. Performance and accuracy

A. Example applications

To evaluate the relative performance and accuracy of our CPU and GPU implementations, we consider two solid-state systems under study with production runs using QMCPACK-GPU. FeO is a parent compound of a major component of Earth's mantel, and is a member of an important class of solids known as transition metal oxides. Experimentally, FeO is believed to transition from an insulator to a metal when subjected to a pressure of about 1 million atmospheres, but the exact nature and mechanism of the transition is not well-understood. Simulations have been performed by Luke Shulenburger at the Carnegie Institution of Washington using QMCPACK on Teragrid resources include NCSA's Lincoln cluster and TACC's Longhorn cluster. Carbon is believed to undergo a transition from the diamond structure to the related BC8 structure at a pressure of approximately 10 million atmospheres, but the transition has not been observed experimentally. Calculations are underway by the present authors to precisely determine the transition pressure using the same resources. Final calculations for a similar study [3] also utilized QMCPACK-GPU.

B. Mixed-precision accuracy

Whatever the speedups achieved by our GPU implementation, the results would be useless if they did not provide sufficient accuracy. We have found that a judicious combination of single and double-precision calculations yields results which are identical within statistical accuracy to the double-precision results obtained from the CPU version of the code. We have tested the accuracy well beyond the target our usual target error bars, giving us enough confidence to plan to use a similar mixed-precision scheme in our CPU implementation. In particular, double precision is retained in two places: 1) to accumulate the statistical averages over the simulation; 2) to occasionally recompute determinant inverses, which are otherwise updated with the Sherman-Morrison formula when a single electron is moved.

C. GPU speedup and analysis

Figure 1 shows the full-application speedup attained when comparing the GPU version of QMCPACK with the CPU-only version on the system described in section III-A. The speedup ratio is obtained by dividing the performance on Tesla Series 10 GPUs by that obtained when running on the same number of quad-core Xeon E5410 CPUs (each utilizing all four cores). Note that the GPU results are obtained with mixed precision, while the CPU calculations were fully double-precision. Also note that CPU code is highly tuned and is *at least* as fast as any comparable code, with key routines hand-optimized with SSE intrinsics. The speedup generally increases with both the system size (number of electrons) and with the number of walkers assigned to each GPU, simply because more in-flight threads afford the GPU more opportunities to hide latency by context switching.

Figure 2 shows a breakdown the total runtime into classes of kernels for both the CPU and GPU implementations. By comparing these percentages, we can make a number of general observations. First, we note that in kernels that are bandwidth limited on both CPU and GPU (B-spline), the wide memory bus of the GPU allows a smaller percentage of total runtime. In kernels that are bandwidth limited on the GPU, but not on the CPU (inverse update, determinant ratios), the CPU spends a smaller percentage of time than the GPU. Finally, for kernels with working sets that fit in on-chip memory on both CPU and GPU (1-body, 2-body, and distance), the proportion of time spent is about the same.

IV. Future work

Ongoing work will proceed to tune the code for the new NVIDIA Fermi architecture, making use of the L1 and L2 caches and faster atomic operations. In order to allow larger systems to be simulated, we plan to move to a hybrid MPI/threading approach to allow the large read-only dataset to be distributed between GPUs on the same host node.

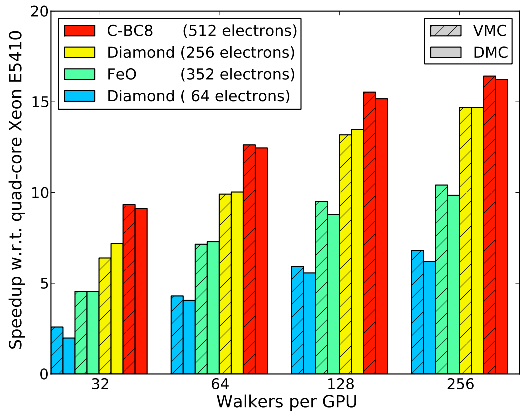

Figure 1. GPU speedup with respect to a quad-core Xeon processors.

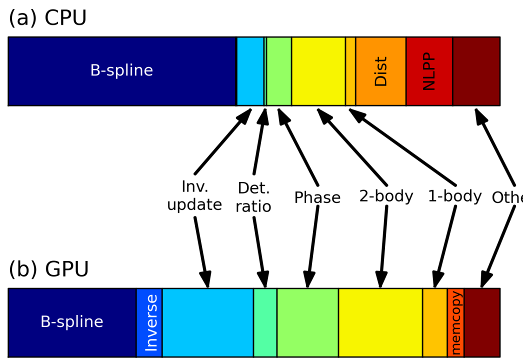

Figure 2. Percentage of run time consumed by the main kernel categories for QMCPACK running on CPUs and GPUs.

Acknowledgments

This work was supported by the U.S. Department of Energy under Contract No. DOE-DE-FG05-08OR23336 and by the National Science Foundation under No. 0904572. This research used computing resources provided by the National Center for Computational Sciences and the Center for Nanophase Materials Sciences and by the National Science Foundation, using Teragrid resources under TG-MCA93S030 and TG-MCA07S016.

References

[1] Jeongnim Kim, K. Esler, J. McMinis, B. Clark, J. Gergely, S. Chiesa, K. Delaney, J. Vincent, and D. Ceperley, “QMCPACK simulation suite, http://qmcpack.cmscc.org.”

[2] W. M. C. Foulkes, L. Mitas, R. J. Needs, and G. Rajagopal, “Quantum monte carlo simulations of solids,” *Rev. Mod. Phys.*, vol. 73, no. 1, pp. 33–83, Jan 2001.

[3] K. P. Esler, R. E. Cohen, B. Militzer, J. Kim, R. J. Needs, and M. D. Towler, “Fundamental high-pressure calibration from all-electron quantum monte carlo calculations,” *Phys. Rev. Lett.*, vol. 104, no. 18, p. 185702, May 2010.