Accelerating Quantum Monte Carlo Applications using Emerging Architectures

Akila Gothandaraman, Ph. D. Candidate (Email: akila@utk.edu)
Advisor: Dr. Gregory Peterson
Department of Electrical Engineering and Computer Science, University of Tennessee, Knoxville
Category: Graduate, First Place in Supercomputing Conference 2008

1. Summary

Recent technological advances have led to a number of emerging computing platforms such as reconfigurable computing, graphics processing units, multi-core processors that provide the tremendous computational power required by scientific computing applications. Hybrid-computing systems are now a reality and next generation supercomputers are likely to have one or more of these technologies. Given this roadmap of supercomputing, our work aims at comparing these compelling platforms for an important class of simulation methods widely used in physics and physical chemistry. Our current research in the acceleration of Quantum Monte Carlo applications using these emerging architectures as well as in the area of detailed analytical performance modeling will help us understand the best ways of mapping applications onto these platforms. This approach will provide practical experience combined with mathematically-based modeling insight into the potential for deploying hybrid architectures for next generation scientific computing applications.

2. Problem and Motivation

A number of computing platforms have emerged as compelling alternatives for scientific applications, allowing users to exploit parallelism at different levels of granularity. High performance reconfigurable computing (HPRC) platforms [1], such as the Cray XD1 [2] or XT5*h* [3] that combine traditional high performance computing (HPC) systems with reconfigurable logic elements such as field-programmable gate-arrays (FPGAs) over a high speed interconnect provide *polygranular parallelism*, a combination of the coarse-grained parallelism typically exploited using parallel computing and the fine-grained parallelism offered by the FPGAs. Current FPGAs with increased gate density provide the resources to implement floating-point cores on the device or the flexibility to use a customized floating-point or fixed-point precision according to application requirements. Graphics processing units (GPUs) have evolved from fixed-function pipelines to flexible general-purpose computing engines with their increased hardware performance and programmability [4, 5]. They also offer double-precision support and easy-to-use programming platforms, making them attractive platforms for scientific computing.

With the advances in HPC systems, computer simulation has established itself as the third leg of science along with experimental and theoretical sciences. Two widely used scientific simulation methods are *Quantum Monte Carlo (QMC)* and *Molecular Dynamics (MD)*. These methods are useful tools for studying many-body systems consisting of atomic or subatomic particles, astrophysical *N*-body systems, protein-folding, drug discovery and material science research. However, these simulations span multiple time- and length- scales, consist of computationally intensive functions and characterized by high performance demands along with high precision requirements. Presently, researchers are trying to tap into the vast power provided by current HPC systems to address fundamentally new science and engineering problems. Our research explores the use of FPGAs and GPUs as coprocessor accelerators by exploiting the best features of these platforms, to accelerate the Quantum Monte Carlo chemistry application used to study quantum many-body systems.

3. Background and Related Work

Monte Carlo methods are stochastic and rely on high quality random number generators and a Markov process to generate the configurations [6]. For example, Quantum Monte Carlo (QMC) methods provide a practical and efficient way of solving the many-body Schrödinger equation, the fundamental equation of quantum mechanics. They sample the N-body quantum mechanical wave function for the purpose of computing observable properties, such as the potential energy. We employ the QMC algorithm to investigate the quantum mechanical ground states of a rare gas cluster of helium atoms. The extremely weak helium-helium interactions combined with small atomic mass make the helium clusters weakly bound and show interesting properties, among which is their ability to attain a superfluid state. Given the complexity of calculations, the only methods that can accurately model highly quantum clusters such as helium clusters are the QMC methods. We show the various steps of one flavor of QMC called *Variational Monte Carlo (VMC)* in Figure 1.

```
        Step 1: Select a reference configuration, $R(x,y,z)$ for the system of atoms at random.
REPEAT (for $I$ iterations)
        REPEAT (for all configurations)
        Step 2: Obtain a new configuration, $R'$ by adding a small random displacement to all the
        atoms in the above configuration.
        Step 3: Compute the ground-state properties (e.g., energy, wave function) of the atoms in
        the current configuration, $R'$. This step consists of the $O(N^2)$ calculations and is chosen
        for hardware acceleration.
        Step 4: Accept or reject the present configuration using the ratio of the trial wave
        function values,
                        $p = \left|\frac{\psi_T(R')}{\psi_T(R)}\right|^2$
                If $p \geq 1$, $R'$ is accepted and retain the properties of this configuration.
                If $p < rand()$, $R'$ is rejected and $R$ is retained and retain the properties of $R$.
        UNTIL finished (for all configurations)
UNTIL finished (for all iterations)
```

Figure 1: Variational Monte Carlo Algorithm

A research effort reported in literature closely related to our QMC research is the GPU implementation of QMCBeaver for computing the electronic structure of a given molecule, on the nVidia 7800 GTX GPU. This implementation achieves a 30x speedup for individual kernels and an overall speedup of 6x over the optimized software application running on a 3.0 GHz Intel Pentium 4 processor [7]. Another QMC code, DCA++ with application to materials science problems and targeted to GPUs, is proposed in [8]. Our work uses the QMC algorithm to calculate the properties of a cluster of inert gas atoms, which is different from the QMC applications in [7, 8]. To the best of our knowledge, this is also the first effort that compares more than one architecture, both HPRC and GPU, for QMC applications.

4. Approach and Uniqueness

As part of our design approach to target the individual FPGA and GPU co-processors, we consider the following two important aspects:
(a) *Performance* : developing efficient algorithms and architectures to significantly accelerate each iteration of

the QMC simulation.
(b) *Accuracy* : evaluating the precision that guarantees the accuracy of our results.

4.1. FPGA Implementation

Our FPGA architecture for the QMC simulation uses novel ideas, such as a two-region approach applicable to the properties of interest, a generic-interpolation framework that can be used to evaluate the functions, use of fixed-point precision and deep pipelining. Our design is targeted to a single core of the dual-core dual-processor 2.2 GHz AMD Opteron with a Xilinx Virtex-4 LX160 FPGA of the *Pacific* Cray XD1 supercomputer.

- *Two-Region Approach*

We illustrate our novel two-region approach using the plot of interatomic potential energy as a function of the distance r as shown in Figure 2. The potential energy exhibits an infinite range for small r and infinite domain for large r. We divide the potential energy function with non-identical numerical behavior into two regions: region I and region II. We employ an exponential transformation in region I to limit the range of the function between 0 and 1. A pseudo-logarithmic binning scheme is used in region II to deal with the large values of distances [9]. The cut-off value, *sigma*, where the potential energy goes to zero separates the two regions. To avoid the expensive square-root operation on the FPGA following the calculation of squared distances, we re-write the potential energy as a function of r^2. We use a quadratic interpolation approach to evaluate the pair-wise potential energies on the FPGA, thereby effectively precomputing the square root by building it into the interpolation coefficients. Hence, only the precalculated interpolation coefficients for the two regions along with the *sigma* value are needed by the FPGA to evaluate the potential energies.

- *Generic Interpolation Framework*

We have developed a generic user-friendly interpolation framework to calculate the ground-state properties using the FPGA. We use a *fixed-point* representation that gives consistent results with double-precision for all calculations on the FPGA. Our framework provides chemists the capability to load different values of interpolation coefficients to investigate a variety of atomic clusters. Figure 3 shows the top-level block diagram of our implementation. The FPGA implementation consists of the following blocks: Distance Calculation (*distCalc*), Potential Energy Calculation (*top_PE*), Wave Function Calculation (*top_WF*). A major portion of our framework is programmable and replicated for all the kernels. Based on whether we employ transformation schemes for our functions, we use different accumulator configurations (*acc_Sum*) and (*acc_Prod*). The host processor reconstructs the floating-point values of the energies and wave functions from the fixed-point results.

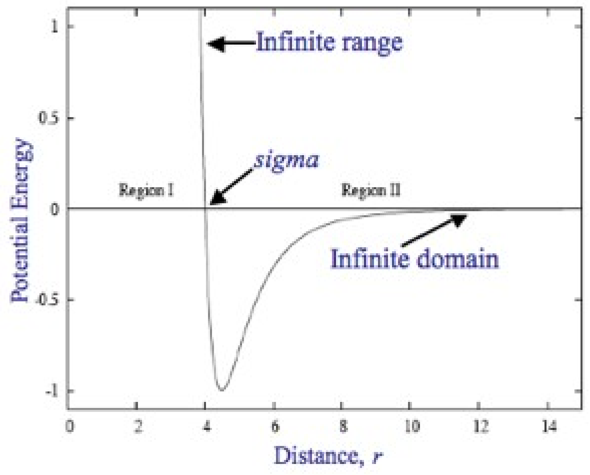
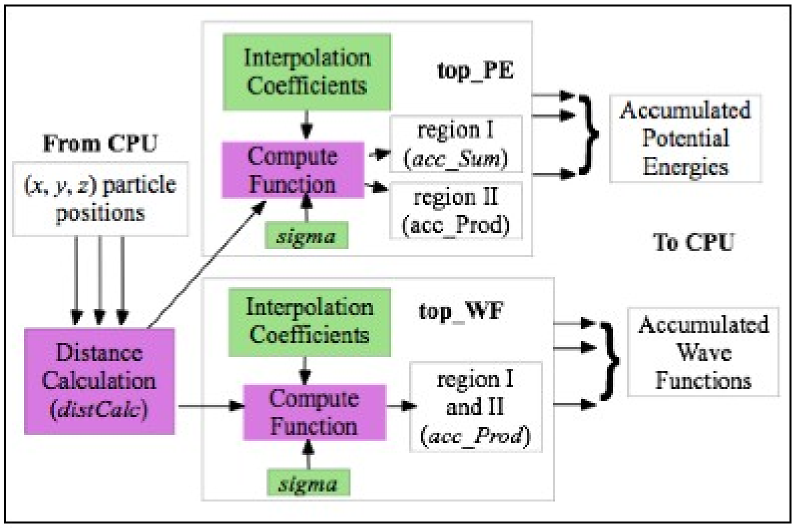


Figure 2: Potential energy as a function of distance



From CPU
(x, y, z) particle positions
Distance Calculation (distCalc)
Interpolation Coefficients
top_PE
Compute Function
sigma
region I (acc_Sum)
region II (acc_Prod)
Accumulated Potential Energies
To CPU
Interpolation Coefficients
top_WF
Compute Function
sigma
region I and II (acc_Prod)
Accumulated Wave Functions

Figure 3: Top-Level FPGA Implementation

4.2. GPU Implementation

We implement the potential energy and wave function kernels onto the NVIDIA GPUs using the Compute Unified Device Architecture (CUDA) programming environment. We target two generations of CUDA compatible GPUs namely, Tesla C870 GPU (128-cores, single-precision) and Tesla C1060 GPU (240-cores, added double-precision support). We use the analytical functions for the evaluation of potential energy and wave function on the GPU directly due to the support on the GPU to evaluate mathematical and transcendental functions. In CUDA, we specify the kernel execution as a *computational grid of thread blocks*. Figure 4 shows the partitioning of the NxN interaction matrix in CUDA. We use a brute-force O(N^2) calculation of all pair-wise interactions. N/p thread blocks operate on a subset of the atoms with each thread within a block calculating the interaction between its atom and other atoms in the system. Our optimization strategies include the use of shared-memory to store a subset of co-ordinate positions of the atoms as well as taking advantage of memory coalescing [10].

5. Results

Figure 5 compares the speedup offered by the FPGA and GPU implementations of the QMC algorithm versus the number of atoms. Here, we summarize the speedups for a cluster of 4096 helium atoms. The FPGA implementation uses fixed-point and delivers an accuracy comparable to the double-precision on the host processor and provides a speedup of 40*x* over the software QMC running on one core of a dual-core dual-processor AMD Opteron 2.2 GHz (used as the baseline). The C870 Tesla GPU implementation using single-precision offers a 160*x* speedup over the software algorithm. The C1060 Tesla GPU implementation using double-precision offers a speedup of roughly 16*x* over the software QMC algorithm. While the single-precision GPU implementation provides the maximum speedup, it does not guarantee the accuracy required for our algorithm. Of the three implementations, the fixed-point FPGA implementation provides the best overall speedup and accuracy. We are working on both optimizing our GPU implementation as well as evaluating the algorithm to determine if we can employ mixed-precision algorithms to further speedup the computations on the GPU.

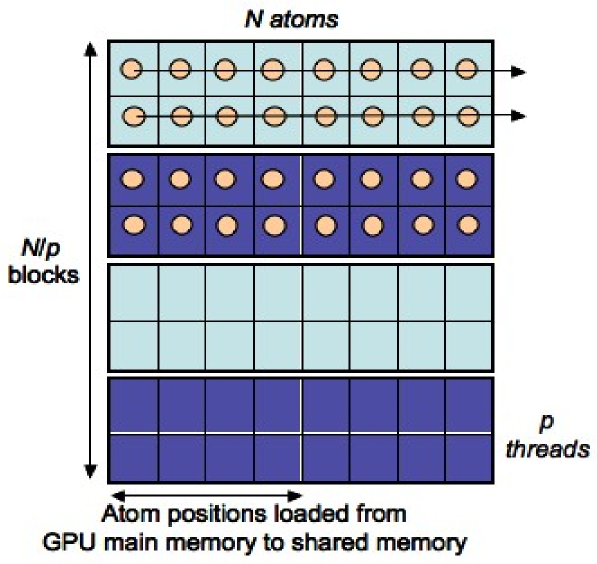


Figure 4: Partitioning of the *N*x*N* interaction matrix on nVidia GPU with CUDA

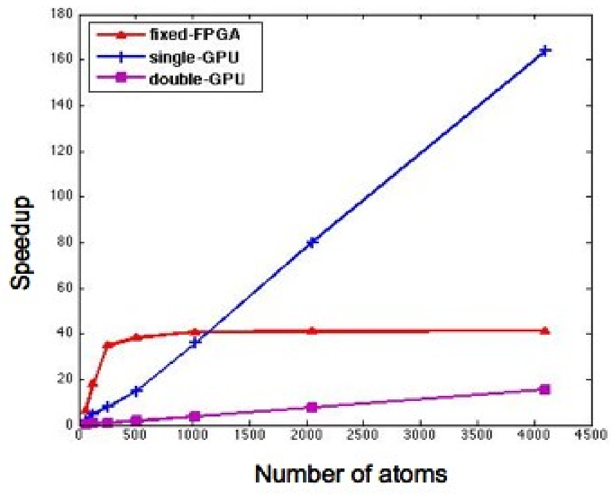


Figure 5: Speedup versus number of atoms

6. Contributions

This research combines innovative algorithms and architectures to accelerate a QMC application, an extremely important and widely used method for studying highly quantum clusters. For the FPGA implementation, our speedup is attributed to the use of pipelining, fixed-point precision of our calculations and the use of deep pipelining strategies. For the GPU implementation, we employ various optimization strategies such as the use of shared memory and coalesced memory accesses.

Significant contributions of our work include the following:

- This is the **first** work to explore reconfigurable architectures for Quantum Monte Carlo simulations. This implementation provides a speedup of 40*x* over an entirely software implementation with no compromise in accuracy for our calculations.
- In order to ensure that our research reaches the end-users, we provide an ***open-source Hardware-Accelerated Quantum Monte Carlo (HAQMC) framework*** . The framework provides scientists the capability to use the generic interpolation framework to obtain additional properties of interest or change the functional forms of the energies and wave functions to simulate a variety of atomic clusters.
- We are extending our HAQMC framework to include support for GPUs so scientists can take advantage of the attractive price-performance ratio and the tremendous data parallelism of GPUs for accelerating their chemistry simulations.

The open-source framework has been submitted to [11] and can be accessed from [12].

7. Conclusions

Emerging computer architectures promise the ability to expore exciting new questions, obtain previously unobservable properties and identify interesting phenomena in the realm of computational science and engineering. This work explores emerging architectures such as FPGAs and GPUs for computational chemistry applications. We have demonstrated a significant speedup with no compromise in accuracy using reconfigurable computing. Future directions in this research include scaling our application to use multiple computing nodes on the Cray XD1 supercomputer or a GPU cluster. We are also working in the area of developing detailed analytical performance models which will help us identify the best ways of mapping our applications onto individual computing platforms as well as onto the hybrid architectures.

Acknowledgements

This work was partially supported by the University of Tennessee Computational Science Initiative and the National Science Foundation grant CHE-0625598. The author also wishes to thank Dr. Robert J. Hinde, University of Tennessee, Knoxville and Dr. Robert J. Harrison, University of Tennessee/Oak Ridge National Laboratory for advice and suggestions on chemistry related topics. We gratefully acknowledge Mr. Dave Strenski, Cray Inc., who provided us access to the Cray XD1 HPRC platform and the Innovative Computing Laboratory, University of Tennessee, Knoxville for providing us access to the GPU systems used in benchmarking.

References

[1] A. DeHon and S. Hauck, "Reconfigurable Computing, The Theory and Practice of FPGA-Based Computation," Morgan Kauffman, 2007.

[2] Cray Inc., Cray XD1, http://www.cray.com/downloads/Cray_XD1_Datasheet.pdf.

[3] Cray Inc., Cray XT5*h*, http://www.cray.com/products/xt5/.

[4] J. D. Owens, D. Luebke, N. Govindaraju, M. Harris, et al., "A Survey of General-Purpose Computation on Graphics Hardware," *Eurographics*, pp. 21-51, August 2005.

[5] nVidia Inc., http://www.nvidia.com/object/tesla_computing_solutions.html.

[6] J. Doll, D. L. Freeman, "Monte Carlo Methods in Chemistry," *IEEE Computational Science and Engineering* , Vol. 1, Issue 3, pp. 22-32, Spring 1994.

[7] A. Anderson, W. A. Goddard III, P. Schröder, "Quantum Monte Carlo in graphics processing units," *Computational Physics Communications*, Vol. 177, Issue 3, pp. 298-306, 2007.

[8] J. S. Meredith, G. Alvarez, T. A. Maier, et al., "Accuracy and Performance of Graphics Processors: A Quantum Monte Carlo Application Case Study," *Parallel Computing*, Vol. 3, Issue. 35, March 2009.

[9] A. Gothandaraman, G. D. Peterson, G. L. Warren, R. J. Hinde, R. J. Harrison, "FPGA acceleration of a Quantum Monte Carlo application," *Parallel Computing*, Vol. 34, Issue 4-5, May 2008.

[10] L. Nyland, M. Harris, J. Prins, "Fast N-Body Simulation with CUDA," GPU Gems 3, Chapter 31, Addison Wesley, 2007.

[11] A. Gothandaraman, G. D. Peterson, G. L. Warren, R. J. Hinde, R. J. Harrison, "A Hardware-Accelerated Quantum Monte Carlo Framework for *N*-body Systems," *Computational Physics Communications (submitted)*

[12] Open-Source Hardware Accelerated Quantum Monte Carlo (HAQMC) Framework, http://www.ece.utk.edu/~agothan1/cpc.html.