A High Performance Programming Model for Large-Scale Molecular Dynamics Calculations on Reconfigurable Supercomputers

Luis E. Cordova
Department of Computer Science and Engineering
University of South Carolina
Columbia, SC 29208
{cordoval}@engr.sc.edu

Melissa C. Smith[1], Sadaf R. Alam[2], and Jeffrey S. Vetter[2]
[1]Engineering Science and Technology Division
[2]Computer Science and Mathematics Division
Oak Ridge National Laboratory
Oak Ridge, TN 37831
{smithmc, alamsr, vetterjs}@ornl.gov

Abstract

We introduce a high performance programming model for large-scale molecular dynamics calculations on reconfigurable hardware. The programming model is based on a methodology to leverage legacy code and accelerate calculations running on reconfigurable supercomputers. We discuss the steps taken in the formulation of scalable, arbitrarily-sized, parallel 3-Dimensional FFT kernels implemented on networks of FPGAs. We achieve sustainable speedup and scaling by exchanging the concept of a network processing-based architecture with gather-scatter operations for a network of streams and accelerator kernels.

Introduction

Accelerating computational chemistry calculations is the ansatz to solve important problems that are otherwise outside the scope of scientific effort due to realistic computational constraints. Molecular dynamics (MD) is a method used to simulate molecular systems solving Newton's equations numerically. Large-scale biomolecular MD simulations for the study of protein folding require calculation of long-range electrostatic forces. Methods based on Ewald sums on the reciprocal space are utilized to further speedup the computation. In order to transform between real and reciprocal domains, we use 3-Dimensional Fast Fourier Transforms (FFTs). Among the methods used to calculate the 3-Dimensional FFTs efficiently, one commonly used is slab decomposition (SD). The SD method allows the use of 1-Dimensional FFTs as a basic module for multi-dimensional FFT calculations. FFTs are ubiquitous in digital signal processing (DSP); however, the scope of operation and data utilized for MD define a totally new and challenging problem for parallelization. The problem is aggravated because of the data movement and accesses while computing over multi-dimensional array-based data structures.

Previous work

The authors are unaware of any previous large-scale MD floating-point implementation on FPGAs giving context to 3-Dimensional FFT kernels. However, here we discuss independent attempts towards parts of the problem. The de facto parallelization methods for distributed and parallel computing architectures use message-passing mechanisms (such as MPI) and multi-threading interfaces (such as OpenMP) in addition to specialized library capabilities such as FFTW [1] for FFT kernels. Other attempts to massively parallelize codes and to dramatically reduce inter-processor communication for the FFT portion of the calculations using CPUs is found in [2]; although a great effort was made to vectorize the code, the improvement would have yielded better performance if limitations of the communication between processing elements (PE) were addressed. Unfortunately, the inefficiency of PEs is inherent in the von Newmann computing model. Furthermore, another salient case is the Blue Gene project's attempt to devise an architecture that effectively deals with the characteristics of MD calculations [3,4,5,6]. The algorithm is "spread-out" on the architecture with multiple-interconnect levels and it implements a volumetric 3-Dimensional FFT instead of the SD method. However, the trade-off between PE and network efficiency remains unsolved because of the difference in bandwidth of the PE and network layers. Conversely, FPGA trends promise to leverage a greater growth than CPUs in explicit parallelism with new generations of devices. Attempts to parallelize floating-point FFTs in reconfigurable hardware demonstrate an already competitive profile between state-of-the-art CPUs and field programmable gate arrays (FPGAs) [7]. One FPGA FFT implementation, reported in [8], is designed to specifically target semi-empirical Car-Parrinello MD types of calculations. When FPGAs are used, bandwidth limitations—indicative of the parallel input format of an FFT—force more effective implementations using FIFO-based producer-consumer models. Examples of such implementations introduce a serial collapsed version of the FFT's butterflies [9]. Finally, the implementation in [10] is optimized for continuous data FFTs in FPGAs and ASICs. The approach extensively uses a corner turn module that we have previously studied and benchmarked in [11].

Reconfigurable computers and Legacy codes

Reconfigurable computing (RC) applications quite often involve the streaming of data past processing logic. The performance advantage gained from reconfigurable computing is largely due to: (1) the ability to process data in a pipelined fashion with "configurable specific instructions," and with spatial parallelism and thus achieve multiple operations per second—note the elimination of instruction fetch and decode steps plus the savings in store/fetch of data from registers across the spatial architecture layout; and (2) data reuse and the ability to exploit the streaming nature of algorithms and its intertwined feature in the way of kernel-stream-kernel compute patterns. We notice that most time consuming portions of legacy code can be rewritten using such compute patterns. Additionally, working with legacy codes allows us to leverage the research hours invested in these algorithms supported by a large variety of theoretical and computational chemistry communities.

High Performance Programming Model

Unlike the standard trend of having a fast network and multiple number of processing elements, we introduce an architectural model on the other side of the spectrum, having processing elements with more computation efficiency and explicit parallelism and a network that is based on algorithmic data flow—streams—rather than a globally addressable space. We achieve sustainable speedup and scaling by exchanging the concept of a network processing-based architecture with gather-scatter operations for a network of streams and accelerator kernels. Therefore, instead of incurring an increased inter-processor communication because of poor system data locality, the algorithm maps spatially over the hardware platform comprised of FPGA chips linked in a reconfigurable array-interconnect. Stream language compilers exist but they do not target the topologies and interconnect available in reconfigurable supercomputers [12]. Data is laid out with high level of reusability and possessing a highly efficient scheme of fetching and feeding data to the pipeline with no conflicts. Consequently, by interleaving communication with computation, the overhead of memory accesses is significantly reduced between processes. Our implementations adapt and integrate [9,10,11] together with our model into an standalone MD application. We use and propose new transformation [13] techniques to map algorithms to hardware with high level compilers. We analyze and accelerate at the application level a total of four legacy codes, two internal kernel codes and two well known packages, namely LAMMPS and Amber. The optimization was performed on the FFT portion of the codes where a large percentage of the computation time is spent. We also compare the performance of our ports with the version of the codes utilizing the FFTW [1] libraries. Our programming model allows us to explore the trade-off between fine vs. coarse grain granularity PEs and data-paths per PE to maximize application scaling. The programming model mapping makes extensive use of the general purpose I/O (GPIO) parallel links and serial links or chain ports (CPIO) implemented using the multi-gigabit transceivers (MGTs) available on the advanced FPGAs.

Evaluation

We validate our techniques by performing tests on two of the most representative state-of-the-art reconfigurable architectures, namely Cray XD1 [14] and the SRC MAPstation [13] both comprising FPGAs, microprocessors, memory, and interconnect. Finally, we discuss limitations and lessons learned of our programming model and also propose architectural enhancements to future generation of reconfigurable supercomputers.

References

[1] M. Frigo, and S. G. Johnson, “The design and implementation of FFTW3,” *Proceedings of the IEEE*, Vol. 93, No. 2, 2005.

[2] H. Ishida, Y. Joti, M. Higuchi, T. Kano, A. Kitao, and N. Go, “Development of Molecular Dynamics Simulation System for Large-Scale Supra-Biomolecules, PABIOS (Parallel BIOmolecular Simulator),” *Annual Report of the Earth Simulator Center*, Apr/Mar. 2004.

[3] M. Eleftheriou, B. G. Fitch, A. Rayshubskiy, T. J. C. Ward, and R. S. Germain, “Scalable framework for 3D FFTs on the Blue Gene/L supercomputer: Implementation and early performance measurements,” *IBM Journal of Research and Development*, Vol. 49, No. 2/3, Mar/May. 2005.

[4] B. G. Fitch, R. S. Germain, M. Mendell, J. Pitera, M. Pitman, A. Rayshubskiy, Y. Sham, F. Suits, W. Swope, T. J. C. Ward, Y. Zhestkov, and R. Zhou, “Blue Matter, an application framework for molecular simulation on Blue Gene,” *J. of Parallel and Distributed Comp.*, Vol. 63, No. Apr/Mar. 2004.

[5] R. S. Germain, Y. Zhestkov, M. Eleftheriou, A. Rayshubskiy, F. Suits, T. J. C. Ward, and B. G. Fitch, “Early performance data on Blue Matter molecular simulation framework,” *IBM Journal of Research and Development*, Vol. 49, No. 2/3, Mar/May. 2005.

[6] Allen et al. (IBM Blue Gene team), “Blue Gene: A vision for protein science using a petaflop supercomputer,” *IBM Systems Journal*, Vol. 40, No. 2, 2001.

[7] K. Scott Hemmert, and Keith D. Underwood, “An Analysis of the Double-Precision Floating-Point FFT on FPGAs,” *IEEE Symposium on Field Programmable Custom Computing Machines*, Apr. 2005.

[8] K. Araki, T. Sasaki, D. Mizoguchi, U. Nagashima, I. Miyoshi, and T. Tanahashi, “Development of a special purpose circuit for 3D-FFT using FPGA,” *The Japanese Society of Fluid Mechanics: 18th Aeromechanics Symposium*, Dec. 2004.

[9] P. A. Jackson, C. P. Chan, J. E. Scalera, C. M. Reader, and M. M. Vai, “A systolic FFT architecture for real time FPGA systems,” *High Performance Embedded Computing Conference*, Sept. 2004.

[10] T. Dillon, “An efficient architecture for ultra long FFTs in FPGAs and ASICs,” *High Performance Embedded Computing Conference*, Sept. 2004.

[11] S. Akella, D. A. Buell, L. E. Cordova, J. Hammes, “The DARPA Data Transposition Benchmark on a Reconfigurable Computer,” *8th International Conference on Military and Aerospace Programmable Logic Devices*, Sept. 2005.

[12] M. Gordon, W. Thies, M. Karczmarek, J. Lin, A. S. Meli, C. Leger, A. A. Lamb, J. Wong, H. Hoffman, D. Z. Maze, and S. Amarasinghe, “A Stream Compiler for Communication-Exposed Architectures,” *ASPLOS,* Oct. 2002.

[13] W. Böhm, and J. Hammes, “A transformational approach to high performance embedded computing,” *High Performance Embedded Computing Conference*, Sept. 2004.

[14] J. L. Tripp, H. S. Mortveit, A. A. Hansson, M. Gokhale, “Metropolitan road traffic simulation on FPGAs,” *IEEE Symposium on Field Programmable Custom Computing Machines*, Apr. 2005.