A Portable, Reusable Framework for Scientific Computing on GPUs

Bryson R. Payne
Georgia College & State University
bryson.payne@gcsu.edu

G. Scott Owen, Irene Weber, Ying Zhu, and Ping Liu
Georgia State University
sowen@gsu.edu, iweber@gsu.edu, yzhu@cs.gsu.edu, pliu1@student.gsu.edu

1 Introduction

Programmable graphics processing units (GPUs) are commonly included as hardware components in new computer workstations, including those workstations used for scientific computing. The current generation of GPU is roughly equivalent in processing power to current CPUs (Central Processing Units), but that power is rarely used to its full capabilities in an average scientific workstation. Numerous advances have been made recently in applying the GPU to parallel matrix processing tasks, such as the Fast Fourier Transform (FFT) [Moreland and Angel 2003].

Our primary goal is to produce a reusable, portable framework [Thompson et al. 2002] for applying GPU-accelerated computation to scientific computing problems. Our secondary goal to apply GPU-accelerated computation to a specific scientific computing problem to serve as a model: automated structure fitting in protein crystallography, a process that normally takes several hours or even days to execute on a sequential CPU.

2 Background

Many researchers in scientific computing desire to accelerate computation-intensive processes using such advances as parallel vector processing on GPUs. However, most existing software packages for scientific computing are already fully featured and would be prohibitively difficult to rebuild from the ground up just in order to take advantage of GPU-accelerated matrix computation. For example, the program used in this research to produce a model of the proposed framework consists of several software packages, both open-source and commercial, with the open-source components comprising hundreds of thousands of lines of FORTRAN (still a highly common scientific computing language) code.

ARP/wARP [Lamzin and Wilson 1999] is the most popular, and most accurate, software package for automatically determining the 3-D structure of proteins from x-ray crystallography data [Badger 2003]. After analyzing runtime data from the automated protein fitting process, it was determined that the most time-consuming step in ARP/wARP is the Refmac refinement step, which uses an open-source molecular refinement program [Murshudov 1997]. Therefore, we focused our attention on applying the GPU-acceleration framework to the Refmac source code, consisting originally of tens of thousands of lines of FORTRAN.

3 Implementation

The Refmac refinement step in the automated protein structure determination process applies a maximum-likelihood operation to matrices that is very similar to smoothing filters in digital image processing. Our first implementation step, therefore, was to demonstrate that the GPU was at least as fast as the CPU at performing convolutions like the Gaussian smoothing filters before applying the framework. It turned out that the GPU (a GeForce FX 5900 with 256MB of RAM) had a speed advantage on larger matrices (2048x2048) of up to 60:1 over the CPU (a 2.8 GHz Pentium 4 with 1 GB of RAM), making Refmac a potentially good candidate for the GPU-acceleration framework.

The second step in our framework was to utilize runtime analysis tools (in this case `gprof`, the GNU profiler used with the GNU C compiler) to determine what portions of Refmac were consuming the most CPU time. It was then necessary to determine if any of those subroutines are good candidates for the accelerated floating-point vector computation possible on the GPU. Two subroutines constituting about 50% of the CPU time in Refmac used large vector computation.

Integrating GPU-CPU distributed computation with existing bioinformatics software introduced a final hurdle: commingling C and Cg or OGSL code for the GPU with the native FORTRAN of Refmac. Fortunately, the freely available `f2c` utility successfully translated the FORTRAN subroutines of interest into compatible C source code, which could then be modified to make use of OpenGL and Cg functions to access the GPU. The open-source GNU compiler then recompiled the mixed C and FORTRAN into the final executable with only minor library specifications needed.

4 Results

Refmac, part of the ARP/wARP scientific computing package, is representative of many existing programs for intensive scientific computation. We have shown that it is possible to successfully modify automatically converted source code from FORTRAN to C in a manner that allows inclusion of OpenGL and Cg or OGSL calls to the GPU and successfully compiles once again into a functioning executable.

As of this writing, only the proof of concept has been completed as discussed above, and the computational load of the two identified candidate algorithms has not been shifted to the GPU. But, by completing the GPU-based acceleration, we hope to be able to achieve about a 2x overall speedup in the automated computation of protein structures under ARP/wARP to further demonstrate the value of this framework.

5 Conclusion

We have successfully demonstrated the application of a generalized framework for integrating GPU-accelerated floating point vector computation into *existing* scientific computing software packages where part or all of the source code is available. With respect to portability, all the Cg code and related C and FORTRAN code developed in this application are directly portable from Windows to UNIX, including Linux, Solaris, and Mac OS X distributions with no modification necessary. This is due to the cross-platform nature of Cg and OGSL and the OpenGL libraries used to access the GPU's functionality.

References

BADGER, J. 2003. An evaluation of automated model-building procedures for protein crystallography. *Acta Crystallographica*, 823-827.

LAMZIN, V. S. and WILSON, K. S. 1997. Automated refinement for protein crystallography. *Methods in Enzymology, 277(B)*, 269-305.

MORELAND, K. and ANGEL, E. 2003. The FFT on a GPU. In *Proceedings of the ACM SIGGRAPH/EUROGRAPHICS conference on graphics hardware*, 112-119.

MURSHUDOV, G.N., VAGIN, A.A., and DODSON, E.J. 1997. Refinement of Macromolecular Structures by the Maximum-Likelihood Method. *Acta Crystallographica D53*, 240-255.

THOMPSON, C.J., HAHN, S. and OSKIN, M. 2002. Using modern graphics architectures for general-purpose computing: a framework and analysis, *Proceedings of the 35th annual ACM/IEEE international symposium on Microarchitecture*, 306-317. IEEE Computer Society Press, 2002.