“Disruptive” Applications of GPGPU Technology

Matthew Curry, University of Alabama at Birmingham, curryml@cis.uab.edu
Anthony Skjellum, University of Alabama at Birmingham, tony@cis.uab.edu

Abstract

GPUs (Graphics Processing Units) are gaining a foothold in various industries to accelerate several types of applications, comprising a potential disruptive technology [1] in computing; that is, one capable of creating a new area of development. Well-known examples include image processing [2], linear algebra [3], and protein folding [4]. These application areas are extrapolations of applications that had previously used SIMD instruction sets, such as SSE (Streaming SIMD Extensions), and leverage the power of GPUs in much the same way that SIMD instruction sets are used. However, the GPUs of today are very different architecturally from SIMD-capable processors. More importantly, GPUs are also unique when compared to the various families of multi-core processors, from Intel’s latest offerings to the Cell Broadband Engine. There exist significant applications that traditionally perform poorly on these types of processing units that stand to benefit highly from GPU-like architectures.

One such application is RAID, explored in the GPU-RAID project [5], an effort that aims to replace expensive hardware-based RAID controllers with software RAID solutions. The basis of this work is in the Reed-Solomon coder/decoder, which is more than an order-of-magnitude faster than the best available CPU implementations. For this particular application, the considerable mathematical capabilities of the GPU are not stressed. Instead, the unique multi-banked multi-memory architecture is utilized to its highest extent, allowing dozens to hundreds of memory lookups to occur simultaneously. Initial investigations into lossless compression expose similar benefits.

With the advent of transparently accessible GPU-based libraries and frameworks, GPU computing could be the ultimate performance booster of many codes with little change to user-level applications. Previously, because of the immaturity and absence of general-purpose programming models, GPU applications were extremely difficult to form and modify without being deeply involved in graphics APIs. Furthermore, until now, true general-purpose toolkits have been vendor-specific [6,7], requiring a developer to choose platforms to support. However, with the arrival of OpenCL [8], there are now two vendor-neutral flexible points between an application and a GPU that will allow for transparent performance gains: Improvements in the hardware (which can execute compatible code more quickly without modification), and improvements in the library (which can evolve with the API or incorporate optimizations and new algorithms). Since the library is programmed in OpenCL, improvements need not be expressed in ways that limit the applicability of the library to a certain vendor.

The goal of this talk is to offer evidence that the GPU is not limited to a small subset of computation in specific fields, but is a way to improve many key application areas today. In order to achieve this goal, we will describe the reasons GPUs are becoming more important as a platform for more traditional HPC/HPEC organizations (such as the U.S. Department of Energy and the U.S. Department of Defense), including the vendor-neutral and familiar means of obtaining performance. We will describe the architecture in a manner to compare and contrast with other popular architectures. Finally, we will present a survey some of the more non-traditional GPU applications and libraries that exercise the platform in unusual and novel ways. We specifically show some of the features of GPUs that allow high performance where CPUs lag.

References

[1] L.J. Bower and C.M. Christensen. "Disruptive Technologies: Catching the Wave," Harvard Business Review, Vol. 73, No. 1, January-February 1995.

[2] J. Cornwall, O. Beckmann and P. Kelly. “Accelerating a C++ Image Processing Library with a GPU,” POHLL 2006: Workshop on Performance Optimization for High-Level Languages and Libraries (colocated with IPDPS06, Rhodes), 2006.

[3] V. Volkov and J.W. Demmel. “Benchmarking GPUs to Tune Dense Linear Algebra,” SC '08: Proceedings of the 2008 ACM/IEEE conference on Supercomputing, 2008.

[4] M.S. Friedrichs, P. Eastman, V. Vaidyanathan, M. Houston, S. Legrand, A.L. Beberg, D.L. Ensign and C.M. Bruns, V.S. Pande. “Accelerating Molecular Dynamic Simulation on Graphics Processing Units,” Journal of Computational Chemistry, Vol. 30, No. 6, 2009.

[5] M.L. Curry, A. Skjellum, H.L. Ward and R. Brightwell. "Arbitrary Dimension Reed-Solomon Coding and Decoding for Extended RAID on GPUs," Petascale Data Storage Workshop, 2008.

[6] D. Kirk. “NVIDIA CUDA Software and GPU Parallel Computing Architecture,” ISMM ‘07: Proceedings of the 6th International Symposium on Memory Management, 2007.

[7] AMD. “AMD Stream Computing: Software Stack,” http://ati.amd.com/technology/streamcomputing/firestream-sdk-whitepaper.pdf, 2007.

[8] Khronos OpenCL Working Group. “The OpenCL Specification: Version 1.0,” 2009.