



THE GPU REVOLUTION AT WORK

By Martin Weigel

Wen-mei W. Hwu (ed.), *GPU Computing Gems: Emerald Edition*, Morgan Kaufmann, 2011, ISBN-13: 978-0123849885, 886 pp.



Five or six years ago, using a graphics processing unit (GPU) for anything other than computer games or more serious visualization purposes was something for hardcore hackers only. With the help of graphics description languages such as OpenGL, you could coax your GPU into doing some useful computation by disguising your arithmetic calculation as a set of graphics transformation and projection operations (or *shaders*) on (triangulation) vertices and textures. This has changed dramatically with the advent of general-purpose GPU (GPGPU) programming frameworks such as Nvidia CUDA and, more recently, OpenCL. Since then, adoption of GPGPU computing in the scientific computing world has surged, with application examples in virtually all areas of science.

This unusual success has been, in part, a victory of the marketing divisions of the two major companies in the GPU market—Nvidia and Advanced Micro Devices (AMD), which acquired GPU developer ATI in 2006. Nvidia, in particular, has paved the way for the adoption of GPUs for general-purpose computing by providing and promoting their CUDA API, the development of special-purpose GPUs dedicated to GPGPU, and the launch of an academic partnership program with funding schemes for research and teaching. That researchers have happily swallowed the bait is not merely a consequence of ingenious marketing,

however. The floating-point performance and memory bandwidth of current GPUs significantly exceed those of current CPU systems, so using them makes sense, scientifically as well as economically.

As with any revolution, scientific or not, the GPU boom generates a need for teaching the masses the new skills and ideology of the time. *GPU Computing Gems: Emerald Edition* is the last addition to a series of books edited by Nvidia representatives; the series started in 2004 with *GPU Gems: Programming Techniques, Tips and Tricks for Real-Time Graphics* and, over the course of two more volumes in 2005 and 2007, has successively moved from a description of graphics applications to GPGPU topics.

Like these previous books, the *Emerald Edition* is no textbook on GPU programming and optimization, but rather a collection of relatively short reports on various GPU applications. In 50 chapters of 10–20 pages each, written primarily by disjoint author teams, researchers report on their success in porting the computationally demanding algorithms relevant to their scientific fields from single CPUs or CPU clusters to GPUs, hybrid CPU/GPU systems, and GPU clusters. In 10 sections, these applications span a wide range of topics, from scientific simulation over problems in bio-informatics, data mining, and electronic circuit design to rendering, signal and image processing, and medical imaging. Apart from a few lines of introduction by the corresponding “area editors,”

the sections as well as chapters are essentially independent of each other.

For whom is this book intended? In his introduction, the editor—Nvidia’s Wen-mei W. Hwu—envisioned it as a “convenient means for application developers in diverse application areas to benefit from each other’s experience.” And indeed, Hwu’s book is a lode of almost 900 pages of rather hard to digest rock, but contains gems that will help you improve your GPU code by analogy to how others solved similar problems in their fields.

All chapters start with an introduction into the background of the application field at hand. Most authors manage to give a nicely self-contained description, but some don’t. In any case, apart from the few chapters that might be directly related to your research field, you’ll probably focus more on the algorithmic aspects than the domain-specific considerations.

Many algorithms discussed are inherently (data) parallel, such that GPU implementations are relatively straightforward and the attention is focused on fine-tuning the code for the target architecture. The book is at its best, however, when discussing implementations of algorithms that are *not* easily parallelized. For example, the tree-based algorithms in Chapters 6 and 29 require significant effort to find enough parallelism and to ensure sufficient data locality to allow for substantial speed gains. Even more interesting, perhaps, are the cases in which completely new algorithms are invented for problems that were previously

considered hard (or impossible) to parallelize—a better algorithm usually trumps better hardware.


Quite notably, many of the optimization strategies that are important for GPGPU computing are discussed in detail in this book, albeit scattered over different chapters. These include well-known aspects such as the appropriate use of the GPU memory hierarchy, including caching; avoidance of thread divergence; thread synchronization via barriers, locks, and atomic operations; increased single-precision floating-point performance from fast intrinsic function implementations; and load balancing. Other strategies that are probably less well known, and thus even more interesting, are also discussed, including just-in-time compilation, mixed precision calculations, and advanced manual cache management. One of the most general strategies is, of course, to minimize global memory transfers and maximize the locality of memory accesses.

With the Nvidia logo printed on the front cover of the book, one wonders whether it's a coincidence that only a

few articles discuss OpenCL implementations instead of (or in addition to) those in CUDA, and even fewer give benchmark results for AMD/ATI GPUs (of which only Chapter 4's electrostatics code shows comparable performance for Nvidia and AMD/ATI GPUs). In the book's defense, the GPGPU computing field is currently dominated by CUDA-based projects. However, one hopes that of the 280 submissions for book chapters mentioned in Hwu's introduction, no AMD affine contributions were rejected for non-scientific reasons.

Several chapters discuss real-time or quasi-real-time applications including, for instance, molecular visualization (Chapter 1), automatic speed-limiting and speech recognition (Chapters 32 and 37), and the various studies discussed in chapters on medical imaging. In these cases, using GPUs can make a qualitative difference by rendering such applications possible in the first place. For the bulk of problems, however, using GPUs is more of a quantitative and economic matter, signified by the possible speed-up

(or slow-down) achievable with respect to other architectures. In the past, astronomical speed-ups have been reported by comparing highly optimized GPU codes to sloppy, single-threaded CPU implementations. Fortunately, many of the present contributors resist a further fanning of the GPU hype and return to a fair, apples-to-apples comparison of GPU performance to CPU results. Others, however, do not. The most relevant comparisons in terms of flops/s per dollar or Watt aren't routinely discussed here. Given this state of affairs, readers are left to put largely exaggerated statements—such as “computing is quickly becoming the third pillar of scientific research, due in large part to the performance gains achieved through GPUs”—that appear in the book blurb into perspective.

The book features a useful index that might help readers mine the gems in search of a solution to a specific algorithmic problem. The index is accompanied by online resources containing source code samples—and further information—for some of the chapters. A second volume with another 30 chapters of GPGPU application reports, somewhat more focused on generic algorithms and programming techniques, is currently in the pipeline and scheduled to appear as the “Jade Edition” sometime this month. 

Martin Weigel is an Emmy Noether Research Fellow at the University of Mainz, Germany. His research revolves around computer simulations in statistical and condensed matter physics, with a focus on the development of new simulation and optimization algorithms and high-performance computing. Weigel has a PhD in physics from the University of Leipzig, Germany. Contact him at weigel@uni-mainz.de and www.cond-mat.physik.uni-mainz.de/~weigel.

