Report: A Programming Framework for Scientific Applications on CPU-GPU Systems

John Owens, University of California, Davis

March 24, 2013

I was awarded the DOE's Early Career Principal Investigator Award in 2004. This was the single most important event in my early career; it validated the research program I had begun and it launched me into a productive research career. It also opened up relationships with DOE scientists across the country. Your confidence in me is very, very, very much appreciated.

At a high level, my research interests center around designing, programming, and evaluating computer systems that use new approaches to solve interesting problems. The rapid change of technology allows a variety of different architectural approaches to computationally difficult problems, and a constantly shifting set of constraints and trends makes the solutions to these problems both challenging and interesting.

One of the most important recent trends in computing has been a move to commodity parallel architectures. This sea change is motivated by the industry's inability to continue to profitably increase performance on a single processor and instead to move to multiple parallel processors.

In the period of review, my most significant work has been leading a research group looking at the use of the graphics processing unit (GPU) as a general-purpose processor. GPUs can potentially deliver superior performance on a broad range of problems than their CPU counterparts, but effectively mapping complex applications to a parallel programming model with an emerging programming environment is a significant and important research problem. As the computing industry moves toward ubiquitous parallel hardware and software, the lessons learned from the GPU, the first commodity parallel processor, are even more important. Our field of "GPU computing" (also called "general-purpose computation on the GPU" [GPGPU]) continues to have a substantial and growing impact on mainstream computing. As one of the early researchers in the field, I was privileged to lead two highly-cited articles[1], supported by this grant, that helped define GPU computing as a field of study.

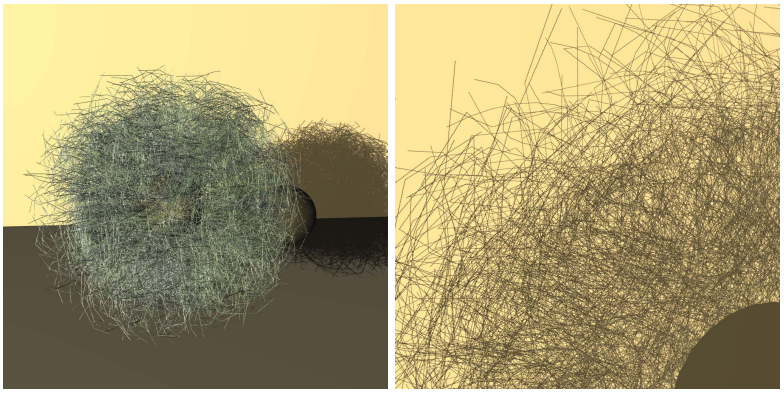
Supported by this award, our work was the first to present an abstraction for GPU data structures as a first-class element in the GPU programming model. With this work, we addressed two important issues in the GPGPU community: first, the difficulty of programming graphics hardware because of the lack of good

[1] John D. Owens, David Luebke, Naga Govindaraju, Mark Harris, Jens Krüger, Aaron E. Lefohn, and Tim Purcell. A survey of general-purpose computation on graphics hardware. *Computer Graphics Forum*, 26(1):80–113, March 2007; and John D. Owens, Mike Houston, David Luebke, Simon Green, John E. Stone, and James C. Phillips. GPU computing. *Proceedings of the IEEE*, 96(5):879–899, May 2008

abstractions and libraries; second, the difficulty of structuring *efficient* computation because the programming model is complex. Based on our preliminary work in parallel data structures[2], we developed "Glift," a library for GPUs that addressed these problems[3]. Our contributions include an abstraction that separates data structures from algorithms, a factorization of data structures into modular components, a definition of an iterator model to tie data structures and algorithms together, and a demonstration that data structures built with this structure both reduce code complexity and yield good performance.

We described Glift in 2006 in ACM Transactions on Graphics[4]. It was also the basis for my student Aaron Lefohn's Ph.D. dissertation, for which he won the 2007 Department of Computer Science Outstanding Dissertation Award. Glift (and this award) was also fundamental in several other publications from our group[5].

Using Glift, we were able to target a long-standing problem in computer graphics that was previously thought too difficult for the GPU: adaptive shadow maps (ASM). In realistic images, shadows provide important visual cues for the viewer, but generating them in real-time with high quality is quite difficult computationally. Most of today's real-time methods severely compromise shadow quality to maintain acceptable performance.



A difficult scene for shadow map algorithms, this furball consists of 4,000 self-shadowing hairs of 12 line segments each and is shadowed with a $32{,}768^2$ effective resolution, resolution-matched shadow map. For a 1024^2 image, our implementation running on an NVIDIA GeForce 8800 GPU GTX rendered the left image at 60–75 fps with a static light and 20–25 fps for a moving light. The right image is a close-up of the hair shadow on the wall, and renders at 70–75 frames per second (fps) with a static light and 60–65 fps with a moving light. Our implementation was typically 2–3 times faster, and for some scenes up to 10 times faster, than a highly-optimized, GPU-based implementation of the original ASM algorithm.

[2] Aaron Lefohn, Joe Kniss, and John Owens. Implementing efficient parallel data structures on GPUs. In Matt Pharr, editor, *GPU Gems 2*, chapter 33, pages 521–545. Addison Wesley, March 2005

[3] Aaron E. Lefohn, Joe Kniss, Robert Strzodka, Shubhabrata Sengupta, and John D. Owens. Glift: Generic, efficient, random-access GPU data structures. *ACM Transactions on Graphics*, 25(1):60–99, January 2006

[4] Aaron E. Lefohn, Joe Kniss, Robert Strzodka, Shubhabrata Sengupta, and John D. Owens. Glift: Generic, efficient, random-access GPU data structures. *ACM Transactions on Graphics*, 25(1):60–99, January 2006

[5] Joe Kniss, Aaron Lefohn, Shubhabrata Sengupta, Robert Strzodka, and John D. Owens. Octree textures on graphics hardware. In *Technical Sketches Program, ACM SIGGRAPH 2005*, August 2005; Aaron Lefohn, Shubhabrata Sengupta, Joe Kniss, Robert Strzodka, and John D. Owens. Dynamic adaptive shadow maps on graphics hardware. In *Technical Sketches Program, ACM SIGGRAPH 2005*, August 2005; Aaron E. Lefohn, Shubhabrata Sengupta, Joe Kniss, Robert Strzodka, and John D. Owens. Glift: Generic data structures for the GPU. In *Proceedings of the 2006 Workshop on Edge Computing Using New Commodity Architectures*, pages D–15–16, May 2006; Aaron E. Lefohn, Shubhabrata Sengupta, and John D. Owens. Resolution-matched shadow maps. *ACM Transactions on Graphics*, 26(4):20:1–20:17, October 2007; Adam Moerschell and John D. Owens. Distributed texture memory in a multi-GPU environment. In *Graphics Hardware 2006*, pages 31–38, September 2006; Adam Moerschell and John D. Owens. Distributed texture memory in a multi-GPU environment. *Computer Graphics Forum*, 27(1):130–151, March 2008; and Shubhabrata Sengupta, Aaron E. Lefohn, and John D. Owens. A work-efficient step-efficient prefix sum algorithm. In *Proceedings of the 2006 Workshop on Edge Computing Using New Commodity Architectures*, pages D–26–27, May 2006

The key to our implementation was a Glift multiresolution adaptive data structure, the first implementation of its kind, which was an ideal match for the adaptive shadow map algorithm[6]. We improved this work with a new algorithm that yields better performance and eliminates a class of artifacts from the original ASM algorithm. This work marked an important milestone in using general-purpose computing to solve a graphics problem[7].

One of the performance-critical kernels in our shadow algorithm, *stream compaction*, turned out to be a gateway to a set of very interesting problems. The best previous work in this area had $O(n \log n)$ computational complexity. We characterized stream compaction in terms of the *scan* parallel primitive, first introduced in the APL programming language in 1962. We reinvented scan for modern data-parallel processors, implementing the first $O(n)$ (linear-time) GPU scan primitive[8]. With Dr. Mark Harris of NVIDIA, we then generalized our scan implementation into a family of scan primitives and adapted them to new GPU hardware, resulting in two recent publications presenting the first space-efficient $O(n)$ scan implementation[9] and the first GPU segmented scan implementation[10] (which won the Best Paper Award at Graphics Hardware 2007). The scan primitives have allowed us to attack a new class of GPU problems and to allow others to do the same. Within the period of review, we used scan to study sorting primitives, tridiagonal matrix solutions (which we previously explored with Pixar in the context of real-time depth-of-field computation[11]), and sparse matrix operations. Beyond the period of review, we've used scan to address many other problems as well. I did much of the early work on scan while visiting Pat McCormick at Los Alamos National Laboratory and appreciate the feedback of Pat and his team along the way.

More generally, I feel that scan provided us an excellent example of a need to rethink our approach to programming parallel machines. Scan is a primitive that is of little use on a scalar machine, and hence would not be part of serial programming primitives. But on the GPU, we demonstrated that scan is both useful and efficient, and the ample computational horsepower of the GPU provides an order of magnitude speedup for a scan implementation over the CPU.

Over the period of review we also shipped three releases of our popular open-source "CUDA Data Parallel Primitives" (CUDPP) GPU primitive library[12]. CUDPP also shipped as part of NVIDIA's CUDA SDK and rapidly gained acceptance as one of the critical libraries for general-purpose computing on the GPU[13]. This library, developed as a collaboration between our group and NVIDIA, implemented a series of broadly applicable data-parallel primitives

[6] Aaron E. Lefohn, Joe Kniss, Robert Strzodka, Shubhabrata Sengupta, and John D. Owens. Glift: Generic, efficient, random-access GPU data structures. *ACM Transactions on Graphics*, 25(1):60–99, January 2006; and Aaron Lefohn, Shubhabrata Sengupta, Joe Kniss, Robert Strzodka, and John D. Owens. Dynamic adaptive shadow maps on graphics hardware. In *Technical Sketches Program, ACM SIGGRAPH 2005*, August 2005

[7] Aaron E. Lefohn, Shubhabrata Sengupta, and John D. Owens. Resolution-matched shadow maps. *ACM Transactions on Graphics*, 26(4):20:1–20:17, October 2007

[8] Shubhabrata Sengupta, Aaron E. Lefohn, and John D. Owens. A work-efficient step-efficient prefix sum algorithm. In *Proceedings of the 2006 Workshop on Edge Computing Using New Commodity Architectures*, pages D–26–27, May 2006

[9] Mark Harris, Shubhabrata Sengupta, and John D. Owens. Parallel prefix sum (scan) with CUDA. In Hubert Nguyen, editor, *GPU Gems 3*, chapter 39, pages 851–876. Addison Wesley, August 2007

[10] Shubhabrata Sengupta, Mark Harris, Yao Zhang, and John D. Owens. Scan primitives for GPU computing. In *Graphics Hardware 2007*, pages 97–106, August 2007

[11] Michael Kass, Aaron Lefohn, and John Owens. Interactive depth of field using simulated diffusion on a GPU. Technical Report #06-01, Pixar Animation Studios, January 2006. http://graphics.pixar.com/library/DepthOfField

[12] CUDPP documentation, examples, and source code can be found at https://code.google.com/p/cudpp/.

[13] In his keynote at the NVISION 2008 conference, Dr. David Kirk, chief scientist at NVIDIA, cited three libraries used as core software libraries for GPU computing: two NVIDIA-written libraries and CUDPP.

including scan, segmented scan, sort, and random number generation for use in GPU computing applications, with many other primitives added after the period of review. The library also comes with test code and ample documentation to allow it to be easily integrated into applications across a wide range of domains. We have continued to extend and maintain CUDPP even after the conclusion of this award. Maintaining a complex open-source software package is a time-consuming and difficult task, but it is a crucial part of ensuring the impact of our research.

A FINAL PROBLEM WE TACKLED UNDER THIS GRANT was how to build an abstraction for writing programs across multiple GPUs. At the time, GPU parallelization either took the form of trivial parallelization (where the problem is evenly divided among GPUs, but requires no communication between GPUs) or a simple pipeline. Our work presented a distributed-shared memory abstraction and implementation across GPUs[14], such that all GPUs shared a single shared memory address space but the memory is distributed across the GPUs. Our underlying system hid the details of sharing and migrating data from GPU to GPU. The most important contribution of the work was our identification of the hardware and software support necessary to properly support a DSM abstraction.

[14] Adam Moerschell and John D. Owens. Distributed texture memory in a multi-GPU environment. In *Graphics Hardware 2006*, pages 31–38, September 2006; and Adam Moerschell and John D. Owens. Distributed texture memory in a multi-GPU environment. *Computer Graphics Forum*, 27(1):130–151, March 2008

AGAIN, I WANT TO EMPHASIZE the impact this award had on my career. You showed a lot of faith in me by funding our work. Thank you for your confidence. I believe that the trajectory of DOE high-performance computation has justified your investment and I look forward to continuing to work with DOE scientists for the remainder of my career.