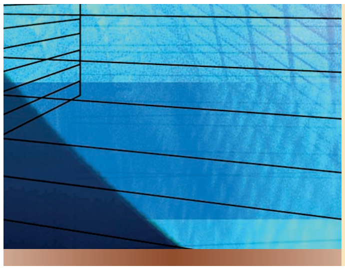
General-Purpose Computations Using Graphics Processors

Dinesh Manocha, University of North Carolina at Chapel Hill

In the past decade, high-performance 3D graphics hardware has become as ubiquitous as floating-point hardware. The graphics processing unit (GPU) has, alongside the CPU, become one of the desktop's two major computational components. Current GPUs consist of 200 to 300 million transistors and provide core clock rates of 400 to 500 MHz. At the high end, GPUs come with 256 or 512 Mbytes of memory and cost around $500.

Developers design these GPUs to operate on data streams consisting of an ordered sequence of attributed primitives like vertices or fragments. Compared to conventional CPUs, GPUs consist of high-bandwidth memories and more floating-point hardware units. For example, a current GPU such as the Nvidia 6800 Ultra has a peak performance of 40 Gflops and a memory bandwidth of 35.2 Gbytes per second, compared to 6.8 Gflops and 6 Gbytes per second for a 3-GHz Pentium 4 CPU. Further, GPU performance for graphics applications throughput has been increasing from two to two-and-a-half times a year. This growth rate is faster than Moore's law as it applies to CPUs, which corresponds to about one-and-a-half times a year.



General-purpose graphics processing extends graphics-processor use to nongraphics applications.

Significantly, the recently announced Sony PlayStation 3 console will have a programmable GPU, the Reality Synthesizer, with a peak floating-point performance of 1.8 Tflops and a cell processor with a peak performance of 218 Gflops (www.us.playstation.com/PressReleases.aspx?id=279). Desktop and laptop systems with multiple GPUs also have become available recently.

Deluxe computer gaming

The multibillion dollar video game market is one of the key factors behind innovations and growth in graphics hardware. First-person-shooters like *FarCry* and *Doom 3* are driving the demand for GPU crunching power. These games use the increased processing power of GPUs to run at maximum detail and antialiasing settings.

Gamers demand the most realistic graphics possible, with every detail set to maximum, all delivered to the display monitor at a consistently smooth frame rate. The newest GPUs' shading and rendering hardware can make real-time gaming look almost as good as offline film rendering, thanks to advanced visual effects such as high-dynamic range lighting, parallax mapping, transparency, multisample antialiasing, and shadows. The graphics pipeline uses floating-point units along with single-instruction and multiple-data capabilities to generate these effects at real-time frame rates.

GPGP

General-purpose computation using graphics processors (GPGP) involves the increasing use of GPUs' computational power in scientific, database, geometric, and other nongraphics applications. These GPGP applications use GPUs primarily designed for rapid rasterization of geometric primitives to shaded pixels on the screen.

To utilize the computational power and memory bandwidth available in these processors, many novel algorithms use the rasterization capabilities for nongraphics applications. Developers refer to these algorithms as *GPU-based* to distinguish them from traditional *CPU-based* algorithms that do not utilize the computational power of GPUs. GPU-based algorithms use inherent pipelining, parallelism, and single-instruction and multiple-data (SIMD) capabilities, along with the GPU's vector-processing functionalities, to perform the computations efficiently.

These algorithms also account for the relatively low bandwidth available between the CPU and GPU by per-

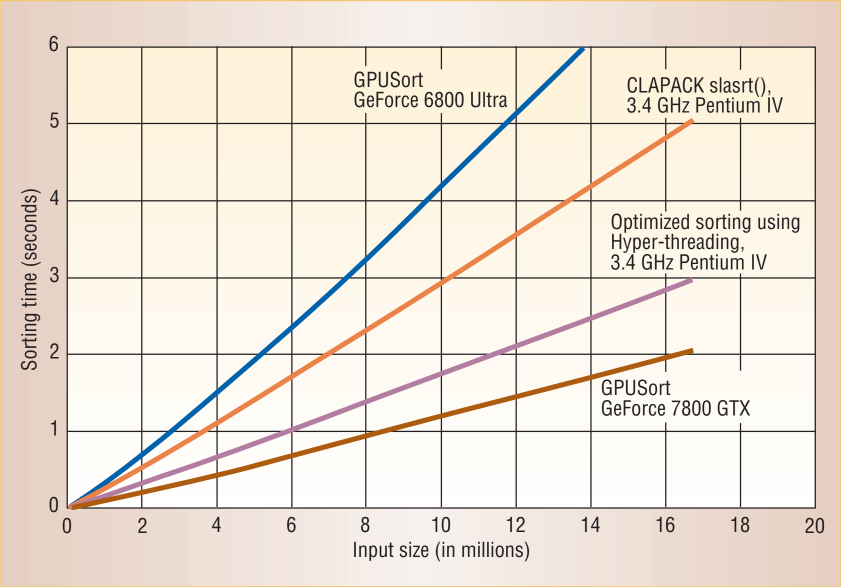


Figure 1. GPUSORT algorithm comparison. The graph compares the performance of optimized GPU- and CPU-based sorting algorithms. The GPU-based algorithm uses the texture mapping and blending hardware on the GPU. Clapack is a hand-optimized Quicksort algorithm. The optimized CPU-based sorting algorithm uses hyperthreading for improved performance.

forming a large fraction of the computation efficiently on the GPU, and use it as an effective coprocessor. Further, they take into account the poor performance of programming constructs such as branching instructions in the current programmable GPU pipeline and use alternate strategies for efficiently evaluating the computations, such as blending-based conditional assignments. Finally, an emerging research area focuses on designing cache-efficient GPU-based algorithms for improved performance.

GPGP APPLICATIONS

Researchers have designed GPU-based algorithms for applications such as database and data mining queries, linear algebra computations, sorting, fast Fourier transforms, motion planning and navigation, global illumination, bioinformatics, cryptography, interference computations, and simulation of physical phenomena including fluid flows (www.gpgpu.org).

In most of these applications, the underlying algorithm's performance directly relates to the GPU's rasterization performance, which continues to increase faster than the growth rate for CPUs. In many cases, GPU-based algorithms can outperform optimized CPU-based implementations by almost an order of magnitude.

The following GPU-based algorithms were developed by the GAMMA research group at the University of North Carolina at Chapel Hill.

Sorting

Researchers have extensively studied this fundamental problem for more than four decades. GPUSORT (http://gamma.cs.unc.edu/GPUSORT) is a novel GPU-based sorting algorithm that exploits the computational power of the graphics pipeline. This algorithm uses simple texture-mapping operations to map the bitonic sorting network to GPUs and effectively utilizes the memory bandwidth through cache-efficient memory accesses.

Figure 1 shows that our implementation outperforms most CPU-based implementations. Further, we have observed a factor-of-three improvement in performance between two successive GPU generations: Nvidia's GeForce 6800 and GeForce 7800. We have also designed algorithms to sort 3D geometric primitives that we use for real-time transparency computations in complex 3D environments (http://gamma.cs.unc.edu/SORT/).

Database algorithms

Database management systems form an integral part of many data warehousing applications and often demand high processing power for fast query execution. The GAMMA research group has designed new algorithms to perform fast relational database operations on GPUs including predicates, Boolean combinations, selectivity and aggregation queries, and join queries (http://gamma.cs.unc.edu/DB).

We have used simple fragment programs to implement our algorithms, and we have applied them to databases with a million records. The results of comparisons performed using SSE-optimized CPU algorithms on high-end Xeon processors indicate a per- formance improvement of five to 20 times using our GPU-accelerated database queries on Nvidia GPUs. Further, we have observed a performance growth rate faster than Moore's law in the speed of these queries on three successive GeForce GPU generations: the 5950, 6800, and 7800.

Stream-mining algorithms

Many real-world applications such as sensor networks, network and financial monitors, and online transaction trackers analyze large volumes of data streams, usually collected from different sources. These data streaming applications must process the data elements in real time and use them to estimate the relevant element's frequency. Given the limited memory requirements and the need for computational resources, the underlying CPU can become resource-limited.

We use the fast stream-processing capabilities of GPUs to estimate the frequencies and quantiles in large data streams. We use GPUSORT for histogram computations as well as to construct epsilon-approximate quantile and frequency summaries. This approach has achieved a 200 percent increase in speed over optimized CPU algorithms on high-end PCs (http://gamma.cs.unc.edu/STREAMING/).

Linear algebra computations

Researchers have studied dense linear algebra solvers for several decades and used them in many high-performance simulations such as fluid dynamics. We have used efficient GPU representations to map and analyze the performance of two direct linear system solvers on GPUs—LU decomposition and Gaussian elimination. Our algorithms, based on novel techniques for streaming index pairs, swap rows and columns and parallelize the computation to utilize the multiple vertex and fragment processors.

By using the peak memory bandwidth available on GPUs, our algorithms have attained a 200 percent increase in speed over cache- and SSE-optimized CPU implementations such as ATLAS (http://gamma.cs.unc.edu/LU-GPU).

Geometric algorithms

Geometric computations are fundamental in computer graphics, geometric and solid modeling, robotics, and GIS and related applications. We have developed new GPU-based algorithms for performing Voronoi computations (http://gamma.cs.unc.edu/voronoi), proximity and collision computations (http://gamma.cs.unc.edu/CULLIDE), transparency- and shadow-generation algorithms, distance fields (http://gamma.cs.unc.edu/DiFi), Minkowski sums, swept volumes, and line-of-sight computation algorithms. This list includes novel algorithms for collision detection between deformable and breaking objects that can also handle self-collisions, such as those shown in Figure 2.

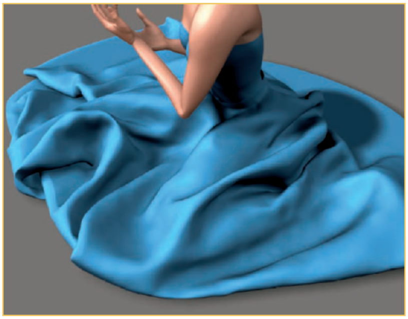


Figure 2. Interactive collision detection for cloth simulation. The cloth is modeled with more than 13,000 triangles and a GPU-based collision-detection algorithm checks for self-collisions at interactive rates. Model courtesy of Rasmus Tamstorf, Walt Disney Feature Animation.

In many cases, we have obtained more than an order-of-magnitude improvement over prior CPU-based algorithms, including collision detection and distance field computations. We have used our algorithms for applications such as interactive walkthroughs of complex virtual environments, terrain processing, dynamic and cloth simulation, and simulating avatar motion in virtual environments.

Motion planning and navigation

Motion planning is a classic robotics problem with applications that include virtual prototyping, surgical simulation, navigation in virtual environments, and computer animation. We have designed novel motion-planning algorithms for dynamic environments and deformable robots in complex environments.

The algorithms for dynamic environments use GPUs' rasterization power to compute distance fields at interactive rates and use the fields to guide the robot toward the goal configuration.

The path planner for deformable robots uses GPU-based algorithms for collision avoidance between the robot and the environment (http://gamma.cs.unc.edu/FlexiPlan). Figure 3 shows an application of this algorithm to compute the path for a deformable catheter used in liver chemoembolization. The GPU-based algorithms outperform prior CPU-based implementations by more than an order of magnitude.

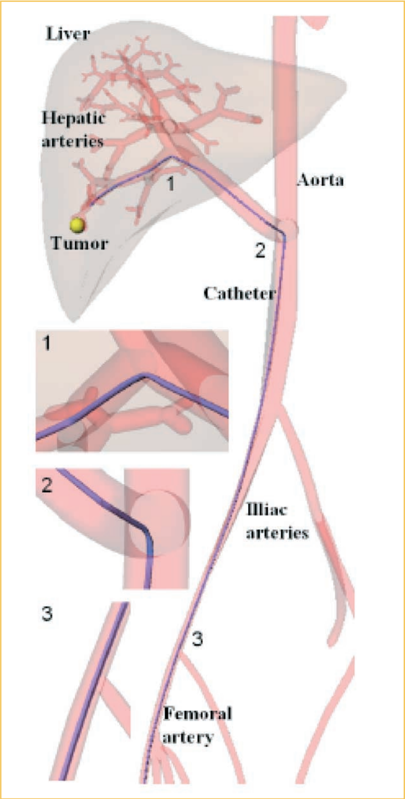


Figure 3. GPU-based path-planning algorithm. This algorithm computed a collision-free path for a deformable catheter in liver chemoembolization. We use GPU-based computations for collision checking and avoidance between the deforming catheter, with about 10,000 polygons, and the geometric model of the arteries, represented by about 83,000 polygons. Model courtesy of William Segars, Johns Hopkins University.

SCIENTIFIC COMPUTATIONS AND COMPUTER-GENERATED FORCES

The GAMMA research group has designed GPU-based algorithms for a variety of scientific applications, including fluid simulation (http://gamma.cs.unc.edu/IMPaSTo/), phase field methods (http://gamma.cs.unc.edu/ICE/), and simulating deformable models. In many of these cases, we reduced the underlying numerical computations to running fragment programs on operands represented in the texture memory.

Computer-generated forces (CGF) emulate battlefield entities and units whose tactical behaviors and decisions are made in part either by human operators or automated-decision algorithms. Over the past few years, some of the major efforts in this area have been directed toward Semi-Automated Forces operational requirements. OneSAF, a composable, next-generation CGF, can represent a full range of operations, systems, and control processes from individual combatant and platform to battalion level. The system has a variable level of fidelity that supports all models and simulation domains.

Real-time terrain reasoning presents computational bottlenecks in terms of scalability and handling many entities in CGF systems. This process includes line-of-sight computations, route planning, and collision avoidance. Our group has developed novel GPU-based algorithms for these three problems and integrated them into the OneSAF Objective System (OOS). These GPU-based algorithms have made a two-orders-of-magnitude improvement in line-of-sight computations within OOS and an almost one-order-of-magnitude improvement in overall simulation on complex terrains.

Some initial results from our work with GPU-based algorithms for nongraphics applications have shown promise, achieving an order-of-magnitude improvement in overall performance in some cases. Encouraged by these results, we and many other researchers and application developers have begun using the GPUs as a coprocessor for other applications.

***Dinesh Manocha** is a professor at the University of North Carolina at Chapel Hill. Contact him at dm@cs.unc.edu.*

Editor: Michael Macedonia, Georgia Tech Research Institute, Atlanta; macedonia@computer.org.