CUDA, Supercomputing for the Masses: Part 1

By Rob Farber, April 15, 2008

CUDA lets you work with familiar programming concepts while developing software that can run on a GPU

Are you interested in getting orders-of-magnitude performance increases over standard multi-core processors, while programming with a high-level language such as C? And would you like that capability to scale across many devices as well?

Many people (myself included) have achieved this level of performance and scalability on non-trivial problems by using CUDA (short for "Compute Unified Device Architecture") from NVIDIA to program inexpensive multi-threaded GPUs. I purposefully stress "programming" because CUDA is an architecture designed to let you do your work, rather than forcing your work to fit within a limited set of performance libraries. With CUDA, you get to exploit your abilities to design software to achieve best performance on your multi-threaded hardware -- and have fun as well because figuring out the right mapping is captivating, plus the software development environment is both reasonable and straightforward.

This is the first of a series of articles to introduce you to the power of CUDA -- through working code -- and to the thought process to help you map applications onto multi-threaded hardware (such as GPUs) to get big performance increases. Of course, not all problems can be mapped efficiently onto multi-threaded hardware, so part of my thought process will be to distinguish what will and what won't work, plus provide a common-sense idea of what might work "well-enough".

"CUDA programming" and "GPGPU programming" are not the same (although CUDA runs on GPUs). Previously, writing software for a GPU meant programming in the language of the GPU. An acquaintance of mine once described this as a process similar to pulling data out of your elbow to get it to where you could look at it with your eyes. CUDA permits working with familiar programming concepts while developing software that can run on a GPU. It also avoids the performance overhead of graphics layer APIs by compiling your software directly to the hardware (GPU assembly language, for instance), thereby providing great performance.

The choice of CUDA device is up to you. Figures 1 and 2 show the CUDA N-body simulation program running on both a laptop and a discrete GPU based desktop PC.

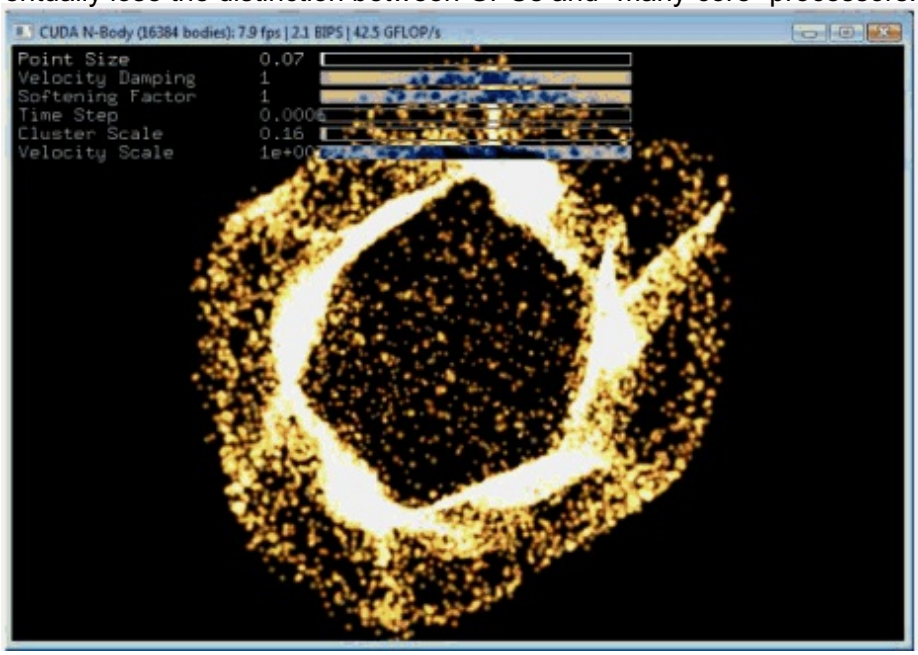
Figure 1: nBody Astrophysics Simulation running on a Quadro FX 570M enabled laptop.

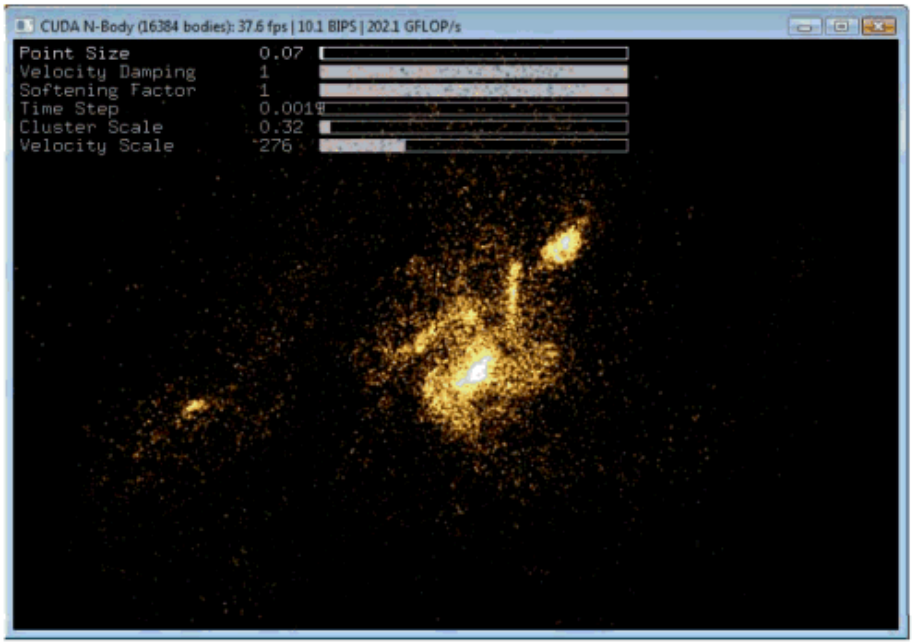
Figure 2: nBody Astrophysics Simulation running on a GeForce 8800 GTS 512MB enabled desktop

Can CUDA really increase application performance by one to two orders of magnitude -- or is all this hype rather than reality?

CUDA is a fairly new technology but there are already many examples in the literature and on the Internet highlighting significant performance boosts using current commodity GPU hardware. Tables 1 and 2 show summaries posted on the NVIDIA and Beckman Institute websites. At the heart of CUDA is the ability for programmers to keep thousands of threads busy. The current generation of NVIDIA GPUs can efficiently support a very large number of threads, and as a result they can deliver one to two orders of magnitude performance increase in application performance. These graphics processors are widely available to anyone at almost any price point. Newer boards will expand CUDA's capabilities by providing greater memory bandwidth, asynchronous data transfer, atomic operations, and double-precision floating point arithmetic among many hardware improvements. Look for the CUDA software environment to

expand as the technology evolves and we eventually lose the distinction between GPUs and "many-core" processors. As developers, we have to anticipate that applications with many thousands of active threads will become commonplace and look for CUDA to run on many platforms, including general-purpose processors.





Example Applications	URL	Application Speedup
Seismic Database	http://www.headwave.com	66x to 100x
Mobile Phone Antenna Simulation	http://www.acceleware.com	45x
Molecular Dynamics	http://www.ks.uiuc.edu/Research/vmd	21x to 100x
Neuron Simulation	http://www.evolvedmachines.com	100x
MRI processing	http://bic-test.beckman.uiuc.edu	245x to 415x
Atmospheric Cloud Simulation	http://www.cs.clemson.edu/~jesteel/clouds.html	50x

Table 1: NVIDIA summary from www.nvidia.com/object/IO_43499.html

GPU Performance Results, March 2008 GeForce8800GTX w/ CUDA 1.1, Driver 169.09		
Calculation / Algorithm	**Algorithm class**	**Speedup vs. Intel QX6700 CPU**
Fluorescence microphotolysis	Iterative matrix / stencil	12x
Pairlist calculation	Particle pair distance test	10x to 11x
Pairlist update	Particle pair distance test	5x to 15x
Molecular dynamics nonbonded force calculation	N-body cutoff force calculations	10x to 20x
Cutoff electron density sum	Particle-grid w/ cutoff	15x to 23x
Cutoff potential summation	Particle-grid w/ cutoff	12x to 21x
Direct Coulomb summation	Particle-grid	44x

Table 2: Beckman Institute table from www.ks.uiuc.edu/Research/vmd/publications/siam2008vmdcuda.pdf