



By

Todd A. Dalrymple

DeVry University

ENG227

Prof. Perkins

Abstract

Nvidia's CUDA technology is helping improve the speed at which systems can computation complex problems. Based on C programming, CUDA allows certain programming to be done by the GPU (Graphic Processing Unit), instead of using the CPU (Central Processing Unit). CUDA allows you to unlock more processing power by accessing the GPU.

Introduction

CUDA stands for Compute Unified Device Architecture. It allows programmers to code algorithms based on C language to for execution on the GPU (graphics processing unit). CUDA was developed by Nvidia who is well known for its graphics cards. Nvidia developed and launched its first product in 1995.

Because GPU's have a "many-core" architecture, thousands of threads can be run simultaneously. CUDA can be used on all graphics cards starting with the GeForce 8 series on. These GPU's are the first to support CUDA. The 8 series supports 32-bit floating point vector processors.

The Fundamentals of CUDA

In order to understand how CUDA works, there needs to be some understanding of some of the basics. Moore's Law becomes integral when dealing with CUDA. Moore's law states that the number of transistors that can be placed on a chip will double every two years. This is applied to all different types of electronics.

In the 8 series GPU's, there are 8 processors per multiprocessor and sixteen multiprocessors per chip. This is what allows the GPU to compute threads quicker than the CPU.

API's is the application programming interface. API's are a set of functions, procedures, or classes that the operating system provides to support requests made by the individual computer programs. CUDA uses both lower level and higher level API's.

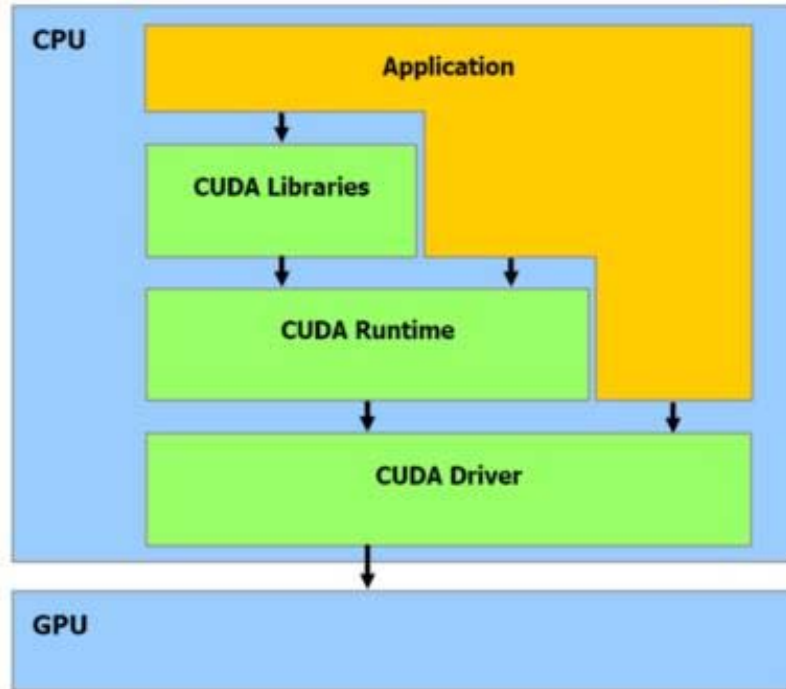


Figure 1: CUDA API's Levels (Abi-Chahla, 2008, para. 12)

On this figure, you can see the levels of the API's. The CUDA runtime API is a high level API. The CUDA Driver API is a low-level API. "Since the high-level is implemented 'above' the low-level API, each call to a function of the Runtime is broken down into more basic instructions managed by the Driver API" (Abi-Chahla, 2008, para. 14).

The Advantages of the GeForce 8

The GeForce 8 series has a combination of independent multi-processors which each contain 8 generalized processors called SP. The SP carry out the same operations like the a SIMD unit, along with 2 specialized ones called SFU's. The multi-processors use these two types to execute instructions in groups of 32 elements. Individual elements are called threads. The threads in CUDA differ from a CPU thread. These threads are a basic element of the data to be processed. When you take 32 threads, it is then called a warp.

With a GeForce 8800 Ultra, the SP and SFU's calculate algorithms at double the frequency of logic control. It can reach speeds up to 1.5 GHz. Because single operations need only one cycle, to execute a warp, you must run two cycles.

"A program, called a 'kernel' is executed in a multiprocessor on blocks of warps, which can contain up to 16 or the equivalent of 512 threads" (Triolet, 2007, para. 6). Since these threads are on the same block, they can communicate through shared memory.

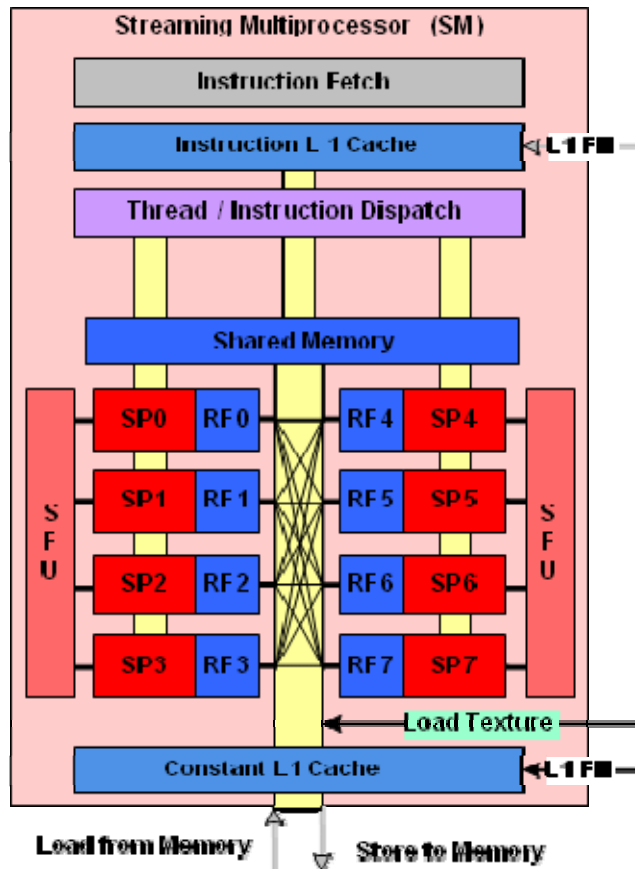


Figure 2: Schema of a multi-processor (Triolet, 2007, para. 5)

To maximize the GeForce 8 type CPU, the programming and data needs to be written in a way that the GPU can use the highest number of threads while staying within the hardware limits. Triolet gives the following technical data for a GeForce 8 GPU (2007 para. 10):

- Threads per SM: 768
- Warps per SM: 24
- Blocks per SM: 8
- Threads per block: 512
- 32 bit registers per SM: 8192
- Shared memory per SM: 16 KB
- Cached constants per SM: 8 KB
- 1D textures cached per SM: 8 KB

Depending on the developer, you can set up the number of threads in a block then the blocks into grids of blocks. A GeForce 8 class can execute up to 2 million instructions.

It should also be noted that it is possible for CUDA to interface directly with OpenGL and DirectX (Arun, 2007, para. 38). With the texture units open in CUDA, they are cached. This takes advantage of the cached memory that is normally just sitting idle.

Expansion

CUDA is still in the process of development. It is version 2.0. Because it is free at this time, several institutions are teaching and working on CUDA. Some of the institutions that are teaching this are as follows:

- Caltech
- Clemson University
- Georgia Institute of Technology
- Johns Hopkins
- Kent State
- North Carolina State
- Perdue University
- University of Colorado
- University of Texas, Austin

Conclusion

CUDA is still in its infancy. Since it's inception in early 2007, Nvidia has made steps to continue to improve it. It had adjusted the performance, flexibility, and its options. As with anything new, there are still bugs and new functions to be added. With time and input by other developers, CUDA will be the leading programming language.

Fedy Abi-Chahla (2007) goes on to explain on his personal feelings in regards to CUDA:

On a more personal level, the impression I got from our first steps with CUDA was extremely positive. Even if you're familiar with the GPU's architecture, it's natural to be apprehensive about programming it, and while the API looks clear at first glance you can't keep from thinking it won't be easy to get convincing results with the architecture. Won't the gain in processing time be siphoned off by the multiple CPU-GPU transfers? And how to make good use of the thousands of threads with almost no synchronization primitive? We started our experimentation with all these uncertainties in mind. But they soon evaporated when the first version of our algorithm, trivial as it was, already proved to be significantly faster than the CPU implementation. (para. 34)

CUDA has opened a new door to the evolution of computers. As CPU's, motherboards and graphics card improve, they will eventually start accessing CUDA to help increase their speed. CUDA will not fall to the way side as other languages have. It builds on one language and evolves into another language.

References

- Abi-Chahla, F. (2008, June 18). Nvidia's CUDA: The end of the CPU? Tom's Hardware.com. Retrieved September 20, 2008, from <http://www.tomshardware.com/reviews/nvidia-cuda-gpu,1954.html>
- Arun. (2007, February 16). NVIDIA CUDA introduction. Beyond 3D.com. Retrieved September 20, 2008 from <http://www.beyond3d.com/content/articles/12/>
- Triolet, D. (2007, August 9). Nvidia CUDA: Practical uses. Behardware.com. Retrieved September 20, 2008 from <http://www.behardware.com/art/imprimer/678/>