УДК 4И(АНГЛ):62

CUDA ARCHITECTURE: GPUS FOR NON-GRAPHIC COMPUTATIONS

Chubareva E.B, Kovaljova A.P.
Scientific supervisor - Associate professor Chubareva E.B.
Siberian Federal University

General purpose graphics processing units (GPGPUs) have a huge impact in everything from professional and home applications to video processing and even physics simulations. No GPU parallel computing architecture has been more in the spotlight than NVIDIA's CUDA either. The developers of this Instruction Set Architecture claim that the usage of a parallel compute engine in NVIDIA's graphics processing units enables solving many complex computational problems significantly faster than a CPU.

Since parallel computing appears to be one of the most promising tendencies in programming, we are convinced that actual knowledge of up-to-date technologies in this sphere is crucial for highly qualified specialists.

This survey was carried out in order to analyze the various facilities provided by the NVIDIA's parallel computing architecture CUDA. In our research we have tried to compare CUDA architecture not only to other GPGPUs but also to modern CPUs and to ascertain, whether the opportunities provided by this technology is competitive, estimating the range of parameters.

First of all we would like to define what CUDA is. CUDA (an acronym for Compute Unified Device Architecture) is a parallel computing architecture developed by NVIDIA. CUDA is the computing engine in NVIDIA graphics processing units that is accessible to software developers through industry standard programming languages. Programmers use 'C for CUDA' (C with NVIDIA extensions) to code algorithms for execution on the GPU. CUDA architecture shares a range of computational interfaces with two competitors - the Khronos Group's Open Computing Language and Microsoft's DirectCompute. It is also available for Python, Fortran, Java and Mathlab. CUDA gives developers access to the native instruction set and memory of the parallel computational elements in CUDA GPUs. Using CUDA, the latest NVIDIA GPUs effectively become open architectures like CPUs.

Unlike CPUs however, GPUs have a parallel "many-core" architecture, each core capable of running thousands of threads simultaneously - if an application is suited to this kind of an architecture, the GPU can offer large performance benefits.

When comparing NVIDIA's GPUs with support of CUDA architecture to CPUs we should first of all admit that a CPU is a general purpose device that is entrusted with various functions while the range of problems solving by GPU is much more restricted.

In addition to this the cores of CPU are designed to execute one thread of instructions at a maximum speed, the GPU, on the other hand, is designed to execute a large amount of parallel threads simultaneously and as fast as possible as well. Although modern CPU has from two to four cores, since the CPU is representing the MIMD (Multiple Instruction Multiple Data) model, each core is executing one thread at a time, and due to the base restrictions of this model increasing quantity of cores does not mean aliquot increase of computation speed. Video chips operations are originally simple and parallel. Thus, the computations can be executed irrespective to the other threads. Furthermore, GPUs have an enormous amount of actuating units that are easily loaded.

What is more CPU and GPU have different approach to memory access and caching. All this distinctions make GPUs more accommodated to solving the task in case the task is easily parallelized. Of course using CUDA is not the only way to make your GPU carry out non-graphic computations. Since that, it seems unfair not to compare CUDA to other solutions provided by different manufacturers.

On the surface of it open standards that use OpenGL seem to be more preferable due to their universality and portability as an opportunity to use one code for different platforms is rather attractive. But on the other hand these methods have a lot of disadvantages: they are much less flexible and not as convenient in use. Furthermore, using these standards means getting no advantage of specific features of a particular video card such as fast shared memory.

Another competitor of CUDA architecture is CTM - a low-level programming interface developed by ATI (now AMD Graphics Products Group), aimed at enabling GPGPU computing. The main difference between CMT and CUDA is that, unlike CUDA, with CTM AMD/ATI have opened up the low-level ISA so that their graphics products can be programed directly in assembly language. The idea here is that a development community will develop the sorts of libraries and higher-level tools that NVIDIA is providing in a prepackaged but closed form with CUDA. People who want to do math coprocessing with NVIDIA's parts will have to rely on the quality, stability, and performance of the company-provided driver, whereas CTM lets you roll your own interface to the hardware if you don't like what's on offer from AMD/ATI.

All in all, CUDA has several advantages over traditional general purpose computation on GPUs (GPGPU) using graphics APIs:

1. Scattered reads – code can read from arbitrary addresses in memory.
2. Shared memory – CUDA exposes a fast shared memory region (16KB in size) that can be shared amongst threads. This can be used as a user-managed cache, enabling higher bandwidth than is possible using texture lookups
3. Faster downloads and readbacks to and from the GPU
4. Full support for integer and bitwise operations, including integer texture lookups.

In order to make sure that usage of NVIDIA GPU acceleration can really fasten the computations we have decided to test it by executing applications based on easily-parallelized algorithm. That is why we chose ElcomSoft Distributed Password Recovery which is believed to be the most technologically advanced password recovery product currently available. The main advantage of this application is that the computations can be executed both on the CPU and NVIDIA GPU.

The hardware on which the product was tested:

CPU: Intel Core 2 Duo E4600 (2400MHz, LGA775, 2048Kb, 800MHz).

GPU: NVIDIA GeForce GT 240 (96 CUDA cores, 128-bit, 8 ROP, 550/1800)

Having tested this password recovery system on various types of applications (Microsoft Office, OpenOffice, etc) executing computations both on CPU and GPU we can definitely say that NVIDIA video card can be called a winner in this “competition”. In some cases it was ten times faster in finding the right password than Intel CPU. Of course we must admit that this kind of tasks is perfect for CUDA architecture because of its internal parallelism, never the less the results are impressing.

To sum up, we can tell that CUDA architecture is a very perspective technology in fields like video and audio encoding, oil and gas exploration, product design, medical imaging, and scientific research

. Of course difficulties connected with producing effective code should be taken into account when deciding whether it is well worth using this technology in your research and hardly can we tell that GPUs will be able to substitute CPUs. Still CUDA has a right to exist as a self-contained architecture for parallel programming. Furthermore, due to the fact that it can be called one of the most effective ways of using personal computer for parallel computing, we are convinced that developing applications using CUDA technology can become a widespread way of fastening computations in different researches carried out on a PC.