GPU-Based High-Performance Computing in Science and Economics

Ivan Plander

Keywords:

graphics processing unit (GPU), general purpose GPU (GPGPU), high-performance computing (HPC), data parallelism, SIMD and SPMD models, CUDA programming model, GPU acceleration of matrix multiplication, green computer, flops per watt.

Abstract:

In the past few years, a new class of high-performance computing (HPC) systems has emerged. These systems employ unconventional processor architectures – such as cell accelerator and graphics processing units (GPUs) – for heavy computations and use conventional central processing units (CPUs) mostly for non-compute-intensive tasks, such as I/O and communication. General Purpose GPUs (GPGPUs) appear for scientific computing. A new concept is to use a GPGPU as a modified form of stream processor. This paper gives an overview of the state-of-the-art of the developments and applications in GPU-based high-performance computing for all platforms: applications, hardware and software technologies, languages and development environments.

Introduction

Computer system architectures today are rapidly advancing and diversifying. A decade ago, most programs ran on conventional, single-core processors capable executing one thread at a time. In the last ten years, those simple CPUs have been replaced by an array of multi-core CPUs, dedicated accelerators that are actually capable of powerful, general-purpose computation. This trend is likely continuing, making it difficult to predict the architectures that will be available in as little as five years or what programming models will be best suited to exploit them.

In the past few years, a new class of high-performance computing (HPC) systems has emerged. These systems employ unconventional processor architectures – such as Cell processor (or field programmable array) (Kahle, J.A. et al. 2005) and graphics processing units (GPUs) (NVIDIA Tesla 2011) – for heavy computations and use conventional central processing units (CPUs) mostly for non-compute-intensive tasks such as I/O and communication.

Today's communications technologies and scientific advances in computer hardware and software are forcing a dramatic change and acceleration in all areas of sciences, engineering and economics. Many scientific and economic applications require the acceleration of linear algebra operations for solution large problems, such as 100 000×100 000 matrix multiplication, inversion etc. In the economics, it should be solved large economic simulation models represented by large matrices. The aim of this paper is to give an overview of the state-of-the-art of the

developments and applications in GPU-based high-performance computing for all platforms: Applications, hardware and software technologies, languages and development environments.

Complexity of computing in science, engineering and economics is rapidly increasing. In the past years complexity of simulations running on the HPC systems requires high processing speed, high level of parallelism (systems with thousands of processing elements), envisioned to reach millions of threads of parallelism and availability of parallelism in algorithms. Architecture problems to be solved for high-performance computing are:

(a) to state limits to manageable levels of parallelism and programming models allowing high performance and efficiency,

(b) to determine the number of cores that can be used to building a single computer and its heterogeneity (CPU/GPU),

(c) to specify the fundamental limits to increasing space dimensions of interconnect,

(d) design considerations for I/O and storage subsystems for huge amounts of data,

(e) to reduce as much as possible the power consumption but also enable the design of even faster computers.

Architectures for extreme-scale computing use the following approaches: *New key technologies:* new near threshold voltage operations, non-silicon memories and photonics.

Multi-core processors are no longer the future of computing they are present day reality. A typical mass-produced CPU features multiple processor cores while a GPU (Graphics Processing unit) may have hundreds or even thousands of cores. With the rise of multi-core architectures has come the need for advanced programming and to program massively parallel processors.

General Purpose Graphics Processing Unit

The Graphics Processing unit, or GPU, has been an integral part of most home computer systems and game consoles for several years. Efforts for ever more realistic games have driven its development from a simple 2D accelerator for graphics-based applications to an extremely powerful unit aimed at 3D games.

The raw computational power of the modern GPU has, in recent years, led to explosion of interest in its use for numerically intensive computing beyond the graphics domain. This interest is demonstrated by the release of dedicated General Purpose GPUs, or GPGPUs, by manufacturers such as NVIDIA) (NVIDIA Tesla 2011) and AMD) (AMD Radeon 2011). The adoption of GPGPU computing by the HPC community is shown by the fact that two of the top four machines in the latest Top 500 list June 2011 employ GPGPUs (Fig. 1).

Rank	Rmax [petaflops]	Name	Processor cores	Company
1	8,162	K computer	548 352 SPARC 64	Fujitsu, Japan
2	2,566	Tianhe-1A	186 368 Xeon 5670, NVIDIA Fermi **GPU**	National computing center Tianjin, China
3	1,759	Jaguar Cray-XT5	224 162 Opteron 6-core	Cray Inc. USA
4	1,271	Nebulae	120 640 Xeon, NVIDIA Tesla 2050 **GPU**	National Supercomputing Centre ShenZhen, China
5	1,192	TSUBAME 2.0	73 278 ProLiant SL390, Xeon 6CX5670, NVIDIA **GPU**	GSIC Center, TokyoInstituteoof Technology, Japan

***Fig. 1**: Top 500 List, June 2011*

A Graphics Processing Unit (GPU) is an accelerator, sometimes called a co-processor, designed to carry out specific graphics tasks faster than the main CPU in the system. It contains one are more microchips designed with a limited number of algorithms in mind. Graphics operations may be split into two types – vector- based operations and raster operations. Vector-based operations are the manipulation of the so called graphics primitives – that is objects such as lines, circles and arcs. A raster, or bitmap, is a structure representing individual image pixels such as those displayed on screen. Raster operations are the manipulation of such bitmaps in various ways such as scrolling a background image between display frames or overlaying a moving sprite over a background.

GPUs were developed during the years 1970s and 1980s, however only in the mid 1980s the first mass-market personal computer appeared including a dedicated chipset capable of taking care of all the graphics functions. The chipset included the famous blitter chip named after the acronym for Block Image Transfer. The blitter was responsible for manipulating large amounts of data corresponding to bitmap images. As well as a personal computer being a popular machine for playing games the advanced graphics capabilities led to its use in video processing, production and scene rendering. However, only in the 1990s the development of GPUs began in earnest. It was in this period more advanced GPUs for IBM-compatible PCs started to be developed. The first of these were simple 2D accelerators, aimed at speeding up the performance of the user interface of the Windows operating system. At about the same time 3D computer games were starting to became popular, leading to the development of GPUs specifically aimed 3D graphics processing. These accelerators became available in games consoles and on the PC thanks to graphic cards. The race was on. Fuelled by the thirst for ever more realistic computer games, the development of 3D GPUs has continued apace ever since. Today market is largely dominated by two companies, AMD and NVIDA. Modern graphic cards are responsible for the many different operations involved in producing a graphic scene which are commonly refered to as rendering pipeline. The input to the pipeline is in the form of information about primitives, typically polygons or triangles. The rendering process transforms and shades the primitives and map them onto to screen for display. The typical pipeline steps are: Geometric vertex generation, Vertex processing, Primitive generation, Primitive processing, Fragment generation or rasterization, Fragment processing, Pixel operations or composition.

The rendering pipeline lends itself to a form of processing called *stream processing*. A stream of data is passed through a series computational stages. The operations within each stage are performed locally and independently on each element within the data stream. GPUs in such a way allow to exploit the inherent *task-parallelism* of streaming (different processor resources being devoted to different stages of the pipeline).

Several stages of the rendering pipeline also lend themselves naturally to another form of parallelism: *data-parallelism*. That means, each geometric vertex or image pixel can be processed independently of the others, but using the same algorithms, in the other words using the common Single Instruction Multiple Data (SIMD) approach. The data parallel approach has consequently evolved form SIMD to a more complicated Single Program Multiple Data (SPMD) model, as hardware capabilities have increased. In the SPMD model, different branches may be followed within a rendering stage for different sections of data.

Modern GPUs typically contain tens or even hundreds of processing units, each unit further containing several Arithmetic Logic Units (ALUs) able to exploit the SIMD characteristic of much of the processing.

The fact that modern GPUs typically contain hundreds of ALUs leads to units of processing power over 1 Tflops, several times that of a typical CPU, for example NVIDIA GPU GeForce GTX580M (NVIDIA 2011). This performance naturally led to an interest in using them for

computationally-intensive problems outside the traditional graphics domain. Manufactures now release products aimed specifically at the HPC market, in particular AMD FireStream (AMD 2011) and the (NVIDIA Tesla 2011). In the latest Top500 list (TOP500 supercomputing sites 2011) (Fig. 1), two of the fastest four machines employ GPGPUs. The top spot taken in this list is by the Japanese RIKEN supercomputer with a performance of 8,162 petaflops (10^{15}) and second one the Chinese Tianhe-1A system with a performance 2,57 petaflops.

General Purpose GPU Scientific Computing

To make best use of the GPGPUs and achieve a performance which makes their use more worthwhile than a standard CPU in scientific computing, significant challenges must be addressed.

The primary issue is that of the application's scope for parallelism: it must demonstrate sufficient data-parallel characteristics such that it can be mapped on the GPU architecture and make full use of the many processing cores available.

The second challenge is making efficient usage of the GPU memory through the application's memory access patterns, where several problems may be encountered. The first problem to address is that of copying data between the main memory of the machine hosting GPU and device itself. This transfer is quite expensive therefore such transfers should be minimized whenever possible. Types of applications likely making the best use of GPGPUs: (a) Ensure that the application has substantial parallelism, (b) Ensure that it has high computational requirements, i.e. a *high ratio of arithmetic operations to memory operations*, (c) Prefer throughput over latency.

General purpose GPU computing is the use of a GPU to do general purpose scientific and engineering computing. The model for GPU computing is to use CPUs and GPUs together in a heterogeneous co-processing PU. The GPU has evolved over the years to have teraflops of floating point performance. NVIDIA revolutionized the GPGPU and accelerated computing world in 2006 – 2007 by introducing its new massively parallel architecture called CUDA. The CUDA architecture consists of hundreds of processor cores that operate together to crunch through the data set in the application.

Success of GPGPUs in the past few years has been easy of programming of the associated CUDA parallel programming model. In this model the application developers modify their application to take the compute-intensive kernels and map them to the GPU. The rest of application remains on the CPU. Mapping a function to the GPU involves rewriting the function to expose the parallelism in the function and adding C keywords to move data to and from the GPU. The developer is tasked with launching 10s of 1000s of threads simultaneously. The GPU hardware manages the threads and does threads scheduling.

The CUDA parallel hardware architecture accompanied by the CUDA parallel programming model provides a set of abstractions that enable expressing fine-grain and coarse-grain data and task parallelism. The programmer can choose to express parallelism in high-level languages, such as C, C++, Fortran or driver APIS, such as OpenCL (Khronos 2011) and Direct X-11 (Microsoft 2010) (Fig. 2). The users should understand alike the basic concepts of parallel programming and the GPU architecture. Case study demonstrates the development process, which begins with computational thinking and ends with efficient programs.

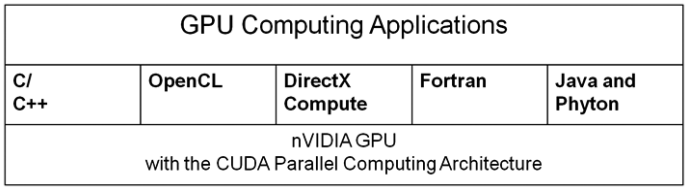

Fig. 2: *A set of software development tools along with libraries and middleware*

GPU Acceleration of Linear Algebra Operations

Many scientific applications require the acceleration of linear algebra operations, which are quite well suited for GPU architectures. The CUDA software development toolkit includes an implementation of the basic linear algebra subprograms (BLAS) library ported to CUDA (Cublas). The most expensive operation is the matrix multiplication (Watson, M. A 2010).

GPU acceleration of matrix multiplications can be realized using cleaving algorithm. Consider the matrix multiplication **C = A B,** where **A** is an $(m \times k)$ matrix, **B** is a $(k \times n)$ matrix, and **C** is an $(m \times n)$ matrix. We can divide A into a column vector of $(r + 1)$ matrices

$$A = \begin{pmatrix} A_o \\ A_l \\ \vdots \\ A_r \end{pmatrix}$$

where each entry $\mathbf{A}_i$ is a $(p_i \times k)$ matrix, and

$$\sum_i^r p_i = m.$$

In practice, all the p_i will be the same.

In a similar manner, can be divided **B** into a row vector of $(s + 1)$ matrices **B =** ($\mathbf{B}_0$, $\mathbf{B}_1$, ... $\mathbf{B}_s$), where each $\mathbf{B}_j$ is a $(k \times q_j)$ matrix and

$$\sum_j^s q_j = n$$

Again, all the q_j will be the same. Then the outer product of this two vectors can be formed:

$$C = \begin{pmatrix} A_o \\ A_l \\ \vdots \\ A_r \end{pmatrix} (B_o \;\; B_l \;\; \cdots \;\; B_s) =$$

$$= \begin{pmatrix} A_o B_o & A_o B_l & \cdots & A_o B_s \\ A_l B_o & A_l B_l & \cdots & A_l B_s \\ \vdots & & \ddots & \vdots \\ A_r B_o & A_r B_l & \cdots & A_r B_s \end{pmatrix}$$

Each individual element $\mathbf{C}_{ij} = \mathbf{A}_i \mathbf{B}_j$ is a $(p_i \times q_j)$ matrix, and can be computed *independently*. Generalizing this to a full implementation for the GPU we get a SIMD matrix multiplication. The

p_i and q_j values can be chosen such that each sub-multiplication fits within the currently available GPU memory. Each multiplication can be organized through the GPU and assembled on the CPU.

Although the SIMD and SPMD models are supported by modern GPUs to make best use of hardware code-branching should still be minimized as much as possible. If *a* code branch occurs all threads must execute both branches, what is suboptimal for performance.

Conclusions

The main reason for using accelerators is because of the need to increase application performance to either decrease the compute time, increase the size of science problem that can be computed, or both.

However, reasons other than pure performance improvements are starting to influence the deployment of HPC resources. As the size of conventional HPC systems increases, their space and power requirements and operational cost quickly outgrow the available resources and budgets. Thus, metrics such as flops per machine, flops per watt of power, or flops per dollar spent on the hardware and its operation are becoming increasingly important. Accelerator-based HPC system looks relatively attractive considering this metrics. The June 2011 Green 500 List of the world´ s most energy efficient supercomputers shows the ranking in(The Green 500 List 2011).

As a multi-billion dollar industry, the commodity games market will continue to be main driver of GPU development. Despite hardware systems (GPU-based high-performance computers) availability, however, the computational science community is currently split between early adopters of accelerators and skeptics. The early adopters´ main concern is that new computing technologies are introduced frequently, and users simply don´t have time to chase after developments that might fade away quickly. With introducing application accelerators, new languages and programming models are emerging that eliminate the option to port code between „standard“ and „non-standard“ architectures. The community fears that these new architectures will result in the creation of many code branches that are not compatible or portable.

On the other side GPUs have evolved to the point where many real-world applications are easily implemented on them and run significantly faster than on the multi-core systems. Future computing architecture will be hybrid systems with parallel-core GPUs working in tandem with multi-core CPUs.

References

- AMD FireStream (2011). *AMD FireStream 9250 GPU Compute Accelerator,* from http://www.amd.com/us/products/workstation/firestream.aspx
- AMD Radeon (2011). *AMD Radeon HD7970 Graphics,* from http://www.amd.com.
- Kahle, J.A. et al. (2005). *Introduction to the Cell Multiprocessor*. IBM J. Research and Development,Vol.49, Nos.4-5, pp.589-604.
- Khronos (2011). *Open CL – The open standard for parallel program heterogeneous systems,* from http://www.khronos.org/opencl/.
- Microsoft (2010). *DirectX 11 – Microsoft Windows,* from http://www.DirectX 11. microsoft.com.
- NVIDIA (2011).*Specification and benchmarks of the NVIDIA GeForce 570M graphics card and notebooks,* from http://www.geforce.com
- NVIDIA Tesla (2011). *High Performance Computing – Supercomputing with Tesla GPUs,* from http://www.nvidia/object/tesla computing solutions.

- The Green 500 (2011). *The Green500 List – June 2011,* from http://www. green 500.org/lists/2011/06/top/list.php.
- TOP500 supercomputing sites (2011). *June 2011 – Top 500 Supercomputers;* from http://www.top500 June 2011.org.
- Watson, M.A.(2010). *Accelerating Correlated Quantum Chemistry Calculation Using Graphics Processing Units.* Computing in Science and Engineering, July/August, pp. 40-50.

Ivan Plander
A. Dubček University of Trenčín in Trenčín
Študentská 2, 91150 Trenčín, ivan.plander@tnuni.sk