OxfordNumerics

Scientific consultancy, mathematical modelling and software development

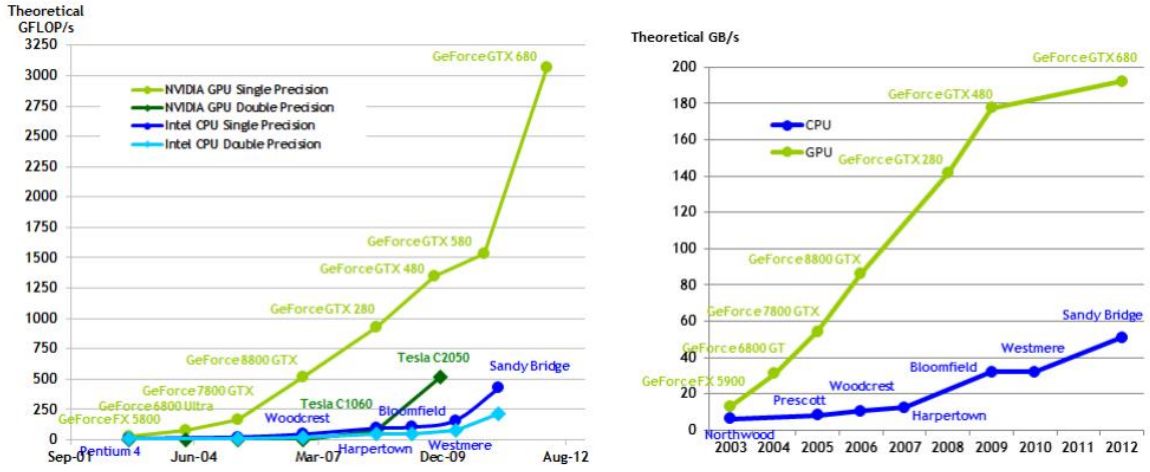
Davina House (Suite 407)
137-149 Goswell Road
London EC1V 7ET
United Kingdom
Telephone: +44 (0) 20 7490 5786
Web: www.oxfordnumerics.co.uk
E-mail: info@oxfordnumerics.co.uk

General Purpose Graphical Processing Unit (GPGPU) Computing approach can offer a powerful and cost-effective way of dramatically improving computational speed of computer simulations. By parallelising the computations and utilising the consumer graphics hardware now present in most modern computers, speed increases of one to two orders of magnitude can potentially be achieved when compared with the more common single-threaded CPU approach.

Achieving this is not without its challenges – we briefly outline what these are with a summary of how we would go about employing the approach for a particular problem and why we are uniquely suited to the task at hand.

Introduction to GPGPU

Recent years have seen increasing interest in the concept of GPGPU computing. This can be attributed to three main factors. The first is the increasing disparity between the floating point computing power provided by GPUs and CPUs as shown in the following diagrams:



Floating-Point Operations per Second and Memory Bandwidth for the CPU and GPU (source: Nvidia Cuda C Programming Guide version 4.2)

The theoretical floating point performance and memory transfer speed of the GPU is many times higher than that of CPUs. Combined with the relatively low price of the hardware and its wide availability in modern computers makes using GPGPU computing an attractive proposition.

The second factor is that along with the increase in power there has been a shift in the design of GPUs, away from hardwired devices designed to perform a few specific graphical operations and towards massively parallel general purpose computing devices. This added programmability has allowed the computing power such GPUs offer to be harnessed for tasks other than graphics.

The final factor is the development of higher level languages for programming these GPUs. Initially coding GPUs required the developer to write in an assembly language specific to the device in question, however recently it has become possible to develop in higher level languages, often derived from C, significantly simplifying and accelerating the development process.

Programming tools

There are no strict requirements for the code used as a starting point for the application of the GPGPU approach. The most common is perhaps C/C++ code as the language affords broad functionality and high performance but starting with existing code in Fortran, Java, Pascal or Basic is entirely possible.

For the GPGPU code itself there are three main Application Programming Interfaces (APIs) currently available: NVIDIA CUDA, OpenCL and DirectCompute. Each API affords a set of advantages and disadvantages and our current framework utilises OpenCL. Due to the considerable number of similarities between the APIs it is possible to switch between them should the project requirements demand so.

Main challenges

There are a number of challenges associated with successfully adopting the GPGPU compute solution to any particular problem. More so than in the case of a conventional serial solution to the problem realising the potential performance of the GPGPU approach requires an understanding of the interaction between the overall algorithm, numerical methods used and low-level implementation details such as memory and computational task management. Some of the main issues are:

- The algorithm needs to be parallelized – not all algorithms lend themselves readily to parallelization without modification (e.g. space-directional or time-explicit schemes)

- The number of parallel computations must be large with preferably homogeneous run times. GPGPU compute approach works best when the number of parallel computations is in the 1000s with each taking approximately the same time – otherwise the benefits quickly diminish. This can be sometimes be addressed by adjusting the numerical approach or selective workgroup grouping.

- To achieve significant performance increases, code must be written specifically to target the GPU architecture. Both computation tasks and memory allocation must be such as to mesh well with the hardware architecture in terms of size, grouping and cross-dependencies.

- Whilst a subject area under active development and research, this is still a relatively new approach. As such, compilers, debuggers and other programming tools are not yet fully refined and automated, requiring a higher level of understanding of hardware and driver behaviour to derive significant performance increases.

OxfordNumerics

What we can do

Each problem considered for the application of the GPGPU compute method is different and the solutions will vary considerably on a case-by-case basis. However, there are some common steps to a successful solution:

- Analyse the algorithm. This is a necessary and most important first step when considering applying the GPGPU approach to an existing solution. Typical steps here are:
    - Identify parallelizable sections
    - Do performance timings to see whether parallizing the code will lead to performance improvements
    - Look at the memory usage of the algorithm to see if it is suited to running on the GPU

- Adjust the numerical approach to better suit GPU implementation if required. Not all algorithms and approaches are readily parallelizable or suitable to run on a GPU – this may be down to the nature of the numerical solution, memory management or a combination of causes. Often these obstacles can be overcome by adjusting the numerical methods used or maybe higher-level algorithm or numerical approach.

- Implementation: Write the parallelized code, integrate with the rest of computations, validate against existing results.

- Refinement and expansion. After the initial performance goals are achieved, the approach can be iterative – sequentially lifting bottlenecks that become apparent after the initial improvement, enhancing solution performance, deeper adjustments to numerical approach and higher-level algorithm.

Why us

We as a company are uniquely suited to the application of the GPGPU approach to solving both new and existing problems. Apart from our personal interest in the subject area, we have:

- Extensive integrated background in physics, applied mathematics and computing combined with a proven history of commercial project delivery

- Record of successful application of GPU compute methods to existing algorithms and deployment of commercialized software with GPU compute sections

- Experience of integration of GPU compute with 3D visualization, with other simulations as part of a workflow and with Graphical User Interfaces as part of commercial packages