The First Thalesian Full-Day Workshop Programme: GPUs in Finance

September 11th, 2009

Department of Computing, Imperial College

Brought to you in collaboration with

NVIDIA SUPERMICRO nag

and computer facilities provided for use by

Imperial College Department of Computing and the **Oxford Supercomputing Centre**.

1 Executive Summary

This workshop will focus on **Monte Carlo Pricing Libraries on High Performance Multicore Architectures**. Lectures and hands-on laboratory exercises will be provided by:

- Prof. Claudio Albanese (King's College, London)
- Thomas Bradley (NVIDIA)
- Prof. Mike Giles (Mathematical Institute, Oxford University)
- Prof. Paul Kelly (Imperial College, London)
- Dr. David Thomas (Imperial College, London)
- Dr. Robert Tong (NAG)
- Gernot Ziegler (NVIDIA)

Introduction to GPU computing hardware by

- Jas Garcha (Supermicro UK)

and a NAG Q&A session and demo by

- John Holden (NAG)

2 Schedule

MORNING SESSION

08:00 - 08:45. Registration and refreshments

08:45 - 09:00. Welcome Address

09:00 - 09:30. Paul Kelly: Software engineering challenges in many-core computing

09:30 - 10:30. David Thomas: Monte Carlo methods implementation on GPUs

10:30 - 12:30. Lab Session (and coffee) : Using Pseudo-random and Quasi-random Number Generators in CUDA
Thomas Bradley & Gernot Ziegler: NVIDIA CUDA SDK
Mike Giles & Robert Tong: NAG Numerical Routines for GPUs

12:30 - 13:30. LUNCH BREAK. Introduction to GPU computing hardware by Jas Garcha and NAG Q&A session and demo by John Holden

AFTERNOON SESSION

13:30 - 14:30. Claudio Albanese: Fourth level BLAS and high performance pricing

14:30 - 16:00. Lab Session (and coffee): Claudio Albanese: OPLib, an open source library for Monte Carlo pricing implemented in CUDA

16:00 - 17:45. Mike Giles: Monte Carlo and finite difference computations on GPUs (including Lab session)

17:45 - 18:30. Panel and open discussion

3 Orientation

Welcome to the first Thalesian GPUs in Finance workshop. This programme will give you a brief introduction to the format of the day, the lecture and lab sessions and direct you to further information to help get you started. Additional lecture material will be handed out. The most important item to check is that you have your login details for the Imperial College Department of Computing machines and for Skynet, the Oxford Supercomputing Centre's GPU cluster. **You should attempt to check that you can login as early as possible to avoid delays in the lab sessions. Do not attempt to issue any commands on Skynet until you have read the usage guide- the URL is provided in the Lab Session Instructions section of this programme.**

GPUs have single-instruction multiple-data (SIMD) architectures. Programs for SIMD processors make extensive use of shared memory to communicate data between the parallel instances of the program. In contrast to serial programs on CPUs, a programmer must have a broader understanding of the issues surrounding parallel implementation and performance engineering on many-core architectures. **Paul Kelly,** will start the day by introducing you to the challenges of high performance software engineering, an overview of which is provided in the session descriptions Section on page 6. His biography is provided on page 12.

The schedule above is split up into morning and afternoon sessions. The morning session focuses on the fundamentals and the afternoon session is more application orientated. **David Thomas** will discuss how Monte-Carlo methods can be efficiently implemented on GPUs. The first lab session will provide the opportunity to experiment with **NAG** and **NVIDIA** pseudo and quasi-random number generators, which are the fundamental building blocks for Monte-Carlo based pricing applications. URLs to instructions on how to run the lab practicals are listed on page 8.

General purpose computing on graphics hardware (GPGPUs) is experiencing a very exciting growth period as we speak. Having an affordable supercomputer sitting on your desk opens up many new possibilities for developing, deploying and executing software in a business environment. **Supermicro UK** are at the forefront of supplying low-cost high performance GPU based desktop computers and lunchtime will provide the opportunity to learn more from **Jas Garcha**. **John Holden** will also be demonstrating and hosting a questions and answers session on the numerical routines for GPUs provided in **NAG**'s new library.

Although the scientific computing community has spent decades developing efficient software for running numerically intensive code on parallel computers, running financial pricing applications on GPUs is a much more recent endeavour and demands new development directions. **Claudio Albanese** will start the afternoon session by describing a novel layer of software building blocks that are required for high performance financial derivative pricing, referred to as Basic Linear Algebra Subroutines (BLAS) level 4. BLAS currently exists as a three layer structure of basic linear algebra routines: vector operations (BLAS 1), matrix-vector operations (BLAS 2) and matrix-matrix operations (BLAS 3). Claudio will explain why **BLAS 4** routines are needed for high performance pricing and how they are implemented in **OpLib** – a set of libraries and benchmarks for high performance pricing routines using a combination of lattice and Monte Carlo methods. Further details are provided on page 9.

The first afternoon lab session will give you the chance to experiment with OPLib. The goal is to understand how a small set of highly optimized base routines can provide a very comprehensive spectrum of pricing applications across all asset classes in higher level languages. You should refer to lab session description on pages 9-10.

Finally, **Mike Giles** will discuss the use of finite difference techniques together with Monte Carlo methods for derivative pricing on GPUs when closed form pricing solutions aren't available. His lecture notes will be provided separately in printed form and are also available on the workshop's website. URLs to additional material are also listed on page 11 below. Mike has been instrumental to the evolution of GPUs in finance and his website provides an excellent reference.

The workshop will close with a panel discussion. This is a good opportunity to share your experiences with using GPUs for financial applications and raise questions about the future of this exciting growth area. I'm sure, like us, you are mindful of OpenCL and the forthcoming many-core Intel CPUs and that practicality is just as important as speed when it comes to adopting CUDA as a tool for implementing finance models. History has shown time and time again that through sharing these experiences, we can all make informed choices about choosing the right tool for the task at hand.

Please help us to make the workshop a knowledge sharing experience by providing feedback on our **blog** (http://blogs.thalesians.com/gpu) and sign up for future workshops, seminars and other related events.

4 Session Descriptions

09:00 - 09:30. Paul Kelly: Software Engineering Challenges in Many-core Computing

This talk is a review and manifesto, looking at what makes building high-performance software hard. I'll talk about some of the correctness and performance issues, with a specific focus on how these concerns are destructive to software quality - how performance hacking breaks software abstractions and reusability. My research has attacked this problem from multiple perspectives; the main aim of this talk is to set out our agenda for software technologies that allow performance engineering to be isolated from higher-level algorithmic development - by delivering pluggable domain-specific, or library-specific, optimisation tools. Our recent examples of this include a collaboration with The Foundry Ltd, on image processing in visual effects post-production, an active library for dense and sparse linear algebra, and program generation tools for finite-element CFD.

09:30 - 10:30. David Thomas: Monte Carlo methods implementation on GPUs

Getting the best performance out of GPUs requires one to understand the architectural differences between GPUs and CPUs, and the way that these differences change the relative costs of computation, storage, and parallelism. To demonstrate these differences, this session follows the development of a simple Monte-Carlo simulation for pricing baskets of derivatives over a number of underlying assets, where the underlyings following a multi-variate normal distribution. During the development process we examine the different types of physical storage types available, and how their interactions with the threads can guide the allocation of high-level data to physical storage.

We also focus on one example of algorithmic-level optimisation within the pricing simulator, where we show how rethinking the process of uniform random number generation can suggest completely different algorithms, designed just for GPUs. In particular, the fast shared memory and tightly-coupled intra-warp parallelism allow us to use multiple threads to work together on a single random number generator. This allows us to derive a new type of generator that provides both a large period (2^1024) and excellent statistical quality, while only requiring six instructions per sample with a generation rate 37 Gwords/sec.

References

- Research on RNGs for GPUs (http://www.doc.ic.ac.uk/~dt10/research/rngs-gpu-uniform.htm)

10:30 - 12:30. Lab Session (and coffee) : Using Pseudo-random and Quasi-random Number Generators in CUDA

Mike Giles, Robert Tong (NAG), Gernot Ziegler (NVIDIA) and Thomas Bradley (NVIDIA)

In this Lab session, Mike, Robert, Gernot and Thomas will be guiding you through the pseudo and quasi-random number generators (RNGs) provided in NAG's numerical routines for GPUs library and the NVIDIA CUDA standard development kit (SDK).

1) For detailed instructions on how to log on to the Oxford supercomputing centre facilities, you should refer to guidelines below.
2) You should consult pages 1-2 of the lab sessions instructions below for how to use the NAG pseudo-random number generator.
3) Links to the NAG GPU library and CUDA SDK documentation are also provided below.

The NAG GPU library provides an implementation of Pierre L'Ecuyer's **MRG32k3a** - a multiple recursive random number generator for the GPU. This generator has a suitably long period in double precision which is sufficient for most scientific applications. The skip-ahead capability of MRG32k3a is particularly suitable for parallel implementation and can take advantage of the GPUs highly parallelized architecture.

The prac2.cu code calls `gpu_mrg32k3a_normal`, which generates a sequence of normal random numbers. The output should look similar to:

```
[cudaXX@compC007 prac2]$ bin/release/prac2
NAG GPU normal RNG execution time (ms): 83.077003 , samples/sec:
2.311109e+09 Monte Carlo kernel execution time (ms): 8.961000
Average value and standard deviation of error = 0.41755106 0.00048179
```

Mike Giles' LIBOR example (libor.cu) demonstrates the usage of `gpu_mrg32k3a_normal` for a LIBOR market model simulation.

```
LIBOR example output:
[cudaXX@compC007 libor]$ bin/release/LIBOR_example_S_Release
GPU time (No Greeks) : 39.938999 msec CPU time (No Greeks) :
56885.074219 msec average value v = 48.95407269 average error =
0.00007025
```

The NVIDIA CUDA SDK 2.3 demonstrates various examples of CUDA implementations of single precision parallel pseudo and quasi-random number generation:

- The **MersenneTwister** project demonstrates an efficient implementation of Matsumoto's Mersenne Twister uniform random number generator. This example also implements a Box-Muller routine for transforming the uniform distribution into a standard normal distribution.

```
cudaXX@compC007 ~]$
   /opt/cuda/2.3/sdk/C/bin/linux/release/MersenneTwister
(remember to qsub -IVX first when using Skynet)
```

- The **SobolQRNG** project demonstrates an efficient parallel implementation by Mike Giles of Joe and Kuo's Sobol' quasi-random generator. This example also implements Acklam's polynomial approximation of the inverse cumulative distribution function for transforming uniform quasi-random numbers to standard normal numbers. Joe and Kuo report that their Sobol' quasi-random number generator exhibits a uniformity property, known as 'property A', in up to 16900 dimensions.

```
cudaXX@compC007 ~]$
   /opt/cuda/2.3/sdk/C/bin/linux/release/SobolQRNG
```

- The **quasirandomGenerator** project demonstrates efficient parallel implementation of Niederreiter quasi-random number sequences. The Monte Carlo Option pricing example uses Niederreiter sequences.

```
cudaXX@compC007 ~]$
   /opt/cuda/2.3/sdk/C/bin/linux/release/quasirandomGenerator
```

Instructions

- Lab session instructions (http://people.maths.ox.ac.uk/~gilesm/cuda/imperial_workshop.pdf)

Guidelines

- Guidelines for Skynet usage (http://people.maths.ox.ac.uk/~gilesm/cuda/skynet_notes.pdf)

References

- Introduction to NAG Numerical Routines (http://www.nag.co.uk/numeric/GPUs/gpu_g05intro.pdf)
- NAG Numerical Routines for GPUs (http://www.nag.co.uk/numeric/GPUs/doc.asp)
- NVIDIA CUDA Programming Guide 2.3 (http://developer.download.nvidia.com/compute/cuda/2_3/toolkit/docs/NVIDIA_CUDA_ProgrammingGuide_2.3.pdf)
- NVIDIA CUDA Best Practices Guide 2.3 (http://developer.download.nvidia.com/compute/cuda/2_3/toolkit/docs/NVIDIA_CUDA_BestPracticesGuide_2.3.pdf)

13:30 - 14:30. Claudio Albanese: Fourth level BLAS and high performance pricing

A large class of generic stochastic processes which are not necessarily analytically solvable but are still numerically tractable can be described by giving transition probability kernels over a contiguous set of time intervals. From the numerical viewpoint, this procedure is highly e
ffective on current microchip architectures as kernels can be conveniently evaluated using GPU co-processors and then used for scenario generation while storing them in CPU caches. This presentation describes the pricing methodology and a mathematical framework for Finance based on direct kernel manipulations, i.e. operator methods. We also discuss a number of techniques based on measure changes to accomplish tasks such as variance reduction and sensitivity calculations. Numerical experiments are included along with performance benchmarks. Source code is distributed under GPL license in a library named OPLib.

14:30 - 16:00. Lab Session (and coffee): Claudio Albanese: OPLib, an open source library for Monte Carlo pricing implemented in CUDA

Performance results to be reviewed and discussed during the Laboratory session are divided into GPU and CPU benchmarks for kernel calculation and scenario generation.

My conclusion based on the equipment I experimented with is that GPUs show an impressive factor 15-20 gain on kernel calculations. Sustained performances on real life problems are: Tesla 860: 180 GF/sec Tesla 1060: 340 GF/sec Xeon 5460: 15 GF/sec Xeon 5500 (Nehalem): 11 GF/sec. This kernel benchmark perhaps is biased in favour of GPUs because I spent a lot of time there developing BLAS Level 4 routines in CUDA such as SGEMM4 and SGEMV4 that operate on tensors. On the CPU side instead I am still using standard BLAS. I attempted to design a CPU side SGEMM4 by queuing MKL calls with no success and will leave that as a challenge for the audience.

On the Monte Carlo scenario generation side instead I worked a lot at optimizing both GPUs and CPUs. I concluded that the two need to be optimized with radically different strategies to exploit the very

different memory/cache configurations. Results on stochastic volatility models on lattices with 512 sites are: Tesla 860: 100 million eval/sec Tesla 1060: 230 million eval/sec Xeon 5460: 180 milion eval/sec Xeon 5500, (Nehalem): 680 million eval/sec An "evaluation" here is a single period draw from a generic Markov chain. For instance, if an interest rate scenario involves generating a curve semiannually over 20 years, that corresponds to 40 evaluations. In my metric, CPUs outperform GPUs by a factor 3 at scenario generation.

The reason for the under-performance is that I am interested in generic processes, not solvable ones which can be coded using mostly registers and shared memory. For instance, the Black-Scholes example in the SDK only needs Box-Mueller transformations. Instead, I give myself a family of cumulative transition probability kernels stored as large matrices in global memory that I need to invert at every period in the simulation by performing a binary search. The factor 3 advantage CPUs show is due to the fact that kernels fit snugly into Level3 cache. On GPUs instead, as far as I understand, one is forced to use global memory for random uncoalesced access. Kernels would fit in shared memory if this was as large as 2 MB per thread block, but we are far from that mark. Instead, on recent CPUs we do have 2MB/core of cache.

Another reason for the difference is that I am using hash tables CPU side to speed up the calculation of inverse probability distribution functions. Since CPU cores are MIMDs with respect to each other, they can branch independently and can take advantage of this. Hash tables however would worsen the performance GPU side because of the overhead due to asymmetric branching in thread blocks and because ultimately thread blocks would default to the worse case scenario. CPUs accelerate slightly [a 10%] in single precision MC calculation, mostly because of better cache management with smaller data structures. I can't use SSE2 there because I would have to eliminate hash tables in order to avoid uneven branching . On kernel calculations instead CPUs have about double speed in single than in double thanks to SSE2.

The ideal solution for orchestrating a Monte Carlo pricing engine is thus a hybrid one, with GPUs computing kernels by fourth level BLAS and CPUs generating scenarios and valuing payoffs. The two tasks would take about the same time in a balanced application. The calculation of sensitivities does not require generating scenarios and valuing payoffs repeatedly for each bumped input, but one can simply generate a new set of kernels and then value Radon-Nykodym derivatives numerically. The method is very robust even for second order derivatives and cross-gammas. Finally, backward induction solutions for callables and calibration should obviously be GPU driven.

Development status

I've just posted OPLib 1.0 RC5, still a beta but a nearly final release of a set of libraries and benchmarks for high performance pricing routines using a combination of lattice and Monte Carlo methods.

References

- Overview of Oplib (http://www.level3finance.com)
- Download (http://www.level3finance.com/oplib/OPLibs_1_0_RC_5.zip)

16:00 - 17:45. Mike Giles: Monte Carlo and finite difference computations on GPUs (including Lab session)

The lecture slides for this talk are supplemental to this booklet. They are also available online through the URL below along with the instructions for his integrated lab session (see page 3) and guidelines for using Skynet.

The laplace3d practical sample output:

```
[cuda01@compC007 laplace3d]$ ./bin/release/laplace3d_new



Grid dimensions: 256 x 256 x 256
Using device 0: Tesla C1060
Copy u1 to device: 31.299999 (ms)
dimGrid  = 8 32 1
dimBlock = 32 8 1
10x GPU_laplace3d: 88.652000 (ms)
Copy u2 to host: 74.970001 (ms)
10x Gold_laplace3d: 1950.051025 (ms)
rms error = 0.000000
```

References

- Lab Session instructions (page 3) (http://people.maths.ox.ac.uk/~gilesm/cuda/imperial_workshop.pdf)
- Lecture slides (http://people.maths.ox.ac.uk/~gilesm/talks/thalesian.pdf)

Guidelines

- Guidelines for Skynet Usage (http://people.maths.ox.ac.uk/~gilesm/cuda/skynet_notes.pdf)

5 Speaker Biographies

Claudio Albanese is a Visiting Professor in the Financial Mathematics Group at King's College and an independent consultant at level 3 Finance. He received his doctorate in Physics from ETH Zurich, following which he held post-doctoral positions at New York University and Princeton University. He was Associate Professor in the Mathematics Department of the University of Toronto and then Professor of Mathematical Finance at Imperial College London.

Thomas Bradley MEng(Hons) MIEE graduated with a first-class MEng degree in Computer Systems Engineering from the University of Bristol in 2000, having also completed the final year of the Diplôme d'Ingénieur at l'École Nationale Supérieure de Télécommunications in Brest, France. He worked as processor architect for video encoding processors at STMicroelectronics before moving to ClearSpeed Technology plc to lead architecture development for general purpose parallel processors. Since then he has specialised in High Performance Computing software development at ClearSpeed and now at NVIDIA.

Mike Giles is a Professor of Scientific Computing, a member of the Oxford-Man Institute of Quantitative Finance, and Associated Director of the Oxford e-Research Centre. He focuses on improving the accuracy, efficiency and analysis of Monte Carlo and finite difference methods. He is interested in various aspects of scientific computing, including high performance parallel computing, and in the last couple of years has been working on the exploitation of graphics cards for scientific applications in both finance and computational engineering.

Paul Kelly has been on the faculty at Imperial College London since 1989. He teaches courses on compilers and advanced computer architecture. The main focus of his Software Performance Optimisation research group is to extend compiler techniques beyond what conventional compilers can do, by exploiting application domain properties - in particular, recently, with regard to exploiting many-core, SIMT and SIMD hardware. He is serving as Program Committee co-Chair for the 2010 ACM Computing Frontiers conference, and chaired the Software track of IPDPS in 2007.

David Thomas is a post-doctoral research associate in the Department of Computing in Imperial College, working mainly with FPGAs, as part of the Custom Computing group. His two main research interests are random number generators for FPGAs, and also financial computing using FPGAs. He has published some 25 articles on computational aspects of Monte Carlo simulations for finance applications, random number generators and applications of reconfigurable computing.

Robert Tong is a technical consultant for NAG where he is involved with library development and has worked on data approximation methods, including wavelets and radial basis functions, in addition to

applications in finance. Robert previously held the post of Research Fellow in Applied Mathematics at the University of Birmingham, where his focus was on the role of numerical software and mathematical modelling in the context of the failure of structures due to extreme events. He received a PhD from the University of Bristol (UK) in the area of applied mathematics and scientific computing, following a first degree in mathematics.

Gernot Ziegler (MSc/civ.ing.) is an Austrian engineer with an MSc degree in Computer Science and Engineering from Linköping University, Sweden. He pursued his PhD studies at the Max-Planck-Institute for Informatics in Saarbrücken, Germany, where he specialized in GPU algorithms for computer vision and data-parallel algorithms for spatial data structures. He now works at NVIDIA.