# Feeling Partial

## Tackling the challenges of PDEs in the brave new world of GPUs in finance

Beyond huge advantages in terms of efficiency and long-term costs the steady adoption of GPU architecture in finance is bringing with it a shift, not only in the way the industry thinks about hardware, but also the thought given to the logical flow of the computational process and, potentially, the kinds of models utilized in quantitative finance.

The treatment of Partial Differential Equations (PDEs) in finance provides an ideal case in point. It clarifies and addresses issues in the old CPU/serial mode. It highlights the massive increases in efficiency of the GPU approach. Most importantly it illustrates the shift in thinking that occurs as soon as you step into the GPU arena; a shift that could ultimately lead to new methodologies in quantitative finance as a whole.

Daniel Egloff will be familiar to most readers of Wilmott as the founder of QuantAlea, a niche consulting shop providing avenues for Tier 2 banks and hedge funds to gain greater operational efficiencies. The consultancy runs a small development team, mainly based in China, whilst Dr. Egloff is in charge of consulting, business development and the quant side of things from the company's base in Central Europe. QuantAlea focuses on high performance derivative pricing, risk management, and the development of quantitative investment and arbitrage strategies. "QuantAlea started to use GPUs from very early on," explains Egloff. "We developed a close relationship with NVIDIA and somehow developed the whole expertise in GPU computing over the last four to five years. NVIDIA is an important strategic partner. They innovate GPU computing with their CUDA parallel computing platform. I can also take back some of the expertise that we gain in the field, for example where GPUs work, what needs to be

changed. In the early days of their CUDA compiler we could suggest several improvements and changes, which gradually made their way into CUDA. NVIDIA was always very responsive and keen to incorporate needs from the business side."

The power and potential of the GPU architecture is something Egloff is intimately familiar with. On the previous generation Tesla C1060 architecture a speedup of a factor of 25 to 40 was easily achievable over an optimized CPU version - a very well-implemented version in C/C++ with proper memory access patterns on the CPU and no performance penalties caused by additional software abstraction layers. In order to achieve a factor of 25 you need 500 to 1000 PDE pricings to batch as a single-parallel task. Such a large number is realistic for options on a major index or in a risk setting where many scenarios have to be processed. The speed increase can go up to a factor of 40 if you have 2,000 problems or more. "You would download these 2,000 pricing problems to the GPU, solve them there and get the result back, and it would typically take you roughly one to two seconds."

For two-dimensional problems like the Heston stochastic volatility PDE solver, the situation is even more interesting. Here a C1060 can already get a speedup of a factor of 20 to 30. The more modern C2050 based on the new Fermi architecture is twice as fast, reaching a factor of 70 to 80. Fermi architecture is able to process larger problems. The older Tesla cards could efficiently process up to 500 grid points per dimension. Beyond that limit, it gets tricky because of the limits of shared memory. On the new Fermi architecture a user can solve pricing problems with approximately 500 to 2,000 grid points in both dimensions. "That's a really big system," Egloff laughs. These calculations are performed with double precision much more efficiently than the previous Tesla cards.

## Theory

"PDEs are the bread and butter for simple exotics or American vanilla evaluation enabling the user to obtain fast and accurate prices and sensitivities even in a world without GPUs." Pierre Spatz, Head of Analytics at Murex, outlines the scenario. Having met with success in transitioning Monte Carlo methods into a GPU setting, Murex, a leading provider of trading and risk management solutions to the capital markets, were keen to take the next step; reprogram their PDEs in order to allow their user base to utilize the same hardware across their whole portfolio. "The job was tricky since, on the one hand we needed to adapt our numerical methods to take advantage of the multiprocessors of the GPU by going from Thomas algorithm towards a Parallel Cyclical Reduction (PCR) algorithm. In many cases we also needed to artificially add some parallelism – by computing simultaneously several options or at least an option and all its sensitivities – simply to take full advantage of the huge available computing power of the GPU."

Egloff explains the challenge that Spatz and his team would have faced arising from the fact that PDEs, in general, are hard to parallelize, particularly one-dimensional PDEs because there is little parallelism to be exploited. For example the tri-diagonal system to be solved in implicit schemes can be solved efficiently with a serial algorithm called the Thomas algorithm. However there are algorithms for fine-grained parallel infrastructures based on the variation of parallel cyclic reduction. "We started to use them already back in 2007 and they turned out to be very efficient. (For more details see Egloff's Wilmott articles (Sept and Nov 2010) The important point is that one is able to solve in parallel a system of N unknowns in about log N steps but that's not adding too much parallelism as Pierre already mentioned. For one-dimensional PDEs this does not really pay off unless the system is very large. However the PDEs in Finance tend to be of moderate to small in size, a few hundreds of grid points is usually accurate enough."

Egloff went in the direction whereby he augmented parallelism in the sense of solving multiple PDEs in parallel. "For instance, you have an index, the index moves, a lot of quoted options are outdated and need to be recalculated. That's quite a common business application. Batching these recalculations easily leads to hundreds of PDEs to be solved. For American options, where PDEs are the primary solution strategy, yet another complication appears. The commonly used Brennan Schwartz algorithm is inherently serial so you need to do something else. I normally suggest an operator splitting approach that is fully parallel and can be implemented nicely on GPU as well."

With these two technologies armed, first the PCR algorithm and second this operator splitting Egloff says that it is possible to solve a huge variety of pricing problems; "Barriers and knock-ins, knock-outs, Americans, window barriers, to various American style contracts. To get a high quality solver you obviously have to pay attention to the discretization, the boundary conditions and other implementation details. But that is down to expertise, which is the same for a GPU or CPU implementation. If you think that the PCR algorithm might also work for multi-core systems you will be disappointed. PCR is good for fine-grained parallel systems with many threads in the order of numbers of discretization points. The CPU has not yet enough threads in parallel such that the benefit of PCR would kick in."

So far we've discussed one-dimensional problems. What about problems with two and more dimen-

**On the new Fermi architecture a user can solve pricing problems with approximately 500 to 2,000 grid points in both dimensions. "That's a really big system," Egloff laughs**

sions, like pricing with a stochastic volatility model or small baskets? "Here the pricing problem has significantly more parallelism, which can be exploited by Alternating Direction Implicit (ADI) methods." Egloff explains, "ADI does multiple sweeps along a single dimension, keeping the others fixed. A sweep breaks up into multiple independent problems, one for each grid point of the sweeping dimension. Hence, a single sweep leads to significant parallel work. Typically multiple sweeps need to be done. Our experience is that already in two dimensions this is sufficient to provide enough parallel computing load on a modern GPU. There is no need any more to batch multiple option pricing problems and solve them at once."

In the traditional CPU setting there is no need for batching. Egloff explains the reason why it has to be done in a GPU setting.

"Let's step back and have a look at the PCR algorithm, which is used to solve the tri-diagonal systems: usually it is set up so that one thread processes one row of the system. To resolve the dependency between the rows, PCR depends on communication between the threads. This poses a severe restriction on how the threads can be executed: they have to run as a block of threads, running all on the same multi-processor of a GPU to get access to the same shared memory. Now a GPU has typically 30 multi-processors. This means that a single one-dimensional PDE can only utilize about 3.3% of the overall GPU capacity. The only way out is to solve 30 or more problems on different multi-processors. This also complicates the software implementation because additional logic is required for a clever scheduling of the pricing problems."

This illustrates one fundamental difference between Monte Carlo and PDE methods: in plain Monte Carlo, each sample is independent and doesn't need to know anything about the other samples. "For PDEs you solve systems of linear equations, which is a global problem: the rows of these systems are dependent, there is no efficient fine-grained parallel algorithm unless the threads can communicate between each other. From this point of view the hardware structure with the different types of memory and the grouping of threads into thread blocks and grids of thread blocks is a useful invention, which we owe to the progressive engineering at NVIDIA."

## In Practice

The level of sophistication that this kind of capacity opens up to players who previously would have found the costs prohibitive is remarkable. Egloff was asked for his own observations on the sorts of use cases at Tier 2 and Tier 3 banks and hedge funds.

"Well, there are actually a few interesting areas of application. The first is obviously you want to trace the market and price options on liquid indices very quickly, with the least amount of hardware. That's the usual and most evident use case. The next use case is risk management. Having the capability to solve PDEs so quickly it is now possible to also use PDE methods in a scenario based risk system. There is one important difference between real time pricing and risk: in real time pricing you can often calibrate once and price hundreds of options. In risk you generally have less options on the same underlying but you want to scale over the number of scenarios. In the extreme case you

# The importance of PDEs in finance

## SciComp presents a review of the major points to consider

When the dimensionality allows, PDE methods are preferred to Monte Carlo methods. The PDEs of finance are with rare exception parabolic, which allows isolation of the region of interest and accurate solutions even when only approximate boundary conditions are available. An entire grid of prices is computed at once along with sensitivities to spot variables (e.g., Deltas, Gammas and Thetas). Sensitivities to process parameters (e.g., Vegas) follow closely related PDEs that may be solved simultaneously, and with attention to discretization can be ensured to be smooth and accurate, even for rough payoffs. American exercise requires only minor modifications to a European style code and is computationally efficient. Fokker-Planck PDEs are extremely useful for model calibration.

However, the fact that an entire grid of prices is computed simultaneously becomes a disadvantage for PDE methods for more than one dimension, i.e.; 'the curse of dimensionality'. With modern CPUs and even GPUs, this generally occurs at the third factor, where 'factor' includes both underlying stochastic factors of the model and any parameters carried along to record path dependencies, (e.g., a range accrual fraction). Beyond this point, Monte Carlo methods are generally required despite problems of relative inefficiency and bias for early exercise, and potentially large variances of sensitivities for rough payoffs.[1]

Traditional clusters are not well suited for execution of multi-dimensional PDE codes across cluster nodes due to node-to-node communication overhead. Therefore the possible acceleration may be limited to number of CPU cores per node.

## Advantages of deploying a GPU based solution

Efficient use of GPU architecture requires keeping thousands of threads busy simultaneously. For one factor PDE models significant acceleration over CPU is possible only by 'stacking up' many (perhaps thousands) of separate PDE solutions. Two-factor PDE solutions can often achieve significant acceleration on GPUs depending on grid sizes. But three factor PDE codes are best suited for GPU solution since CUDA thread grouping (3D blocks of threads and 3D grid of thread blocks[2]) allows a natural mapping of three dimensional domains to CUDA C/C++. Full double precision (DP) support (IEEE 754-2008 compliant) along with the increased number of DP units in NVIDIA Fermi-architecture GPUs allows large accelerations over CPU solutions while obtaining identical numerical results. Efficient and accurate three-dimensional ADI PDE solvers (second order accurate and unconditionally stable for positive definite correlation matrices) can be utilized in a massively parallel fashion. Large reductions in datacenter footprint and power consumption are possible while maintaining the same computational capabilities.

SciComp's code synthesis engine, SciFinance® automatically generates C and C++ pricing model source code from users' custom model specifications written in a concise, high-level language. The code is stand alone cross-platform C/C++ code with no run-time licenses or links to vendor binary libraries.

SciFinance has been extended to produce CUDA source for execution on NVIDIA GPUs. The Monte Carlo CUDA code module, released several years ago, is now followed by a PDE CUDA code module. The PDE module concentrates on two and three factor

calibrate a model for each pricing calculation. To better leverage the GPU you might also want to bring the calibration also to the GPU. This is perfectly feasible. For instance calibrating a local volatility model with Dupire's formula is fully data parallel; and the alternative methods of Andersen and Brotherton-Ratcliffe is very close to solving a PDE, which can again be tackled with the PCR method."

The risk management use case is particularly interesting. Egloff observes a tendency to have the same models for pricing on the trading desks and risk management. When it comes to risk management, many Tier 2 and Tier 3 banks use flat volatility and simple Black-Scholes models so that the pricing can be done analytically in closed form. This allows such institutions to run their Monte Carlo simulations relatively quickly and the data management gets significantly simpler. "Integrating front office pricing into risk systems usually leads to an explosion of required compute capacity," says Egloff, "the model calibration must be executed for every risk scenario. Bringing down the whole calibration and pricing work-flow is substantially more work but when you do it, GPU computing really starts to pick up, you gain much more."

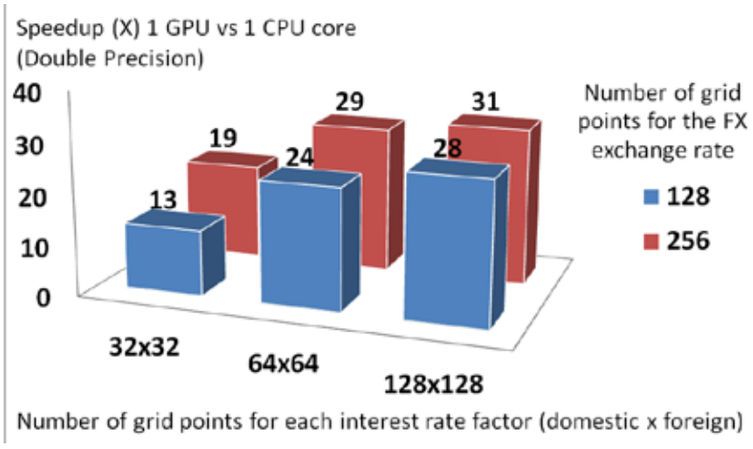
So what does the freeing up of resources and the speed advantage allow for these Tier 2 and Tier 3 institutions as a result? Egloff points to a Tier 2 bank targeting to replace a 750 core cluster with a few dozen GPU boards. "It's hard to compare in money, because third-party vendors do not yet have GPU accelerated software libraries for pricing, which are ready to integrate in a straightforward way. The main lack is the full workflow from data preprocessing to calibration to pricing and Greeks calculation. Not to say that the whole chain of calculations must be configurable to meet specific client needs. Therefore, most clients rely on bespoke implementations, which lead to somewhat higher project costs."

This aspect of things is being approached from both ends of the chain. From the bottom end a lot of interesting support libraries are popping up, like Thrust from NVIDIA bringing C++ Standard Template Library concepts to the GPU, or some public domain libraries in various fields like random number generators and linear system solvers, covering some of the numerical groundwork.

"From the work being done now," says Pierre Spatz, "we feel that the GPU solution is particularly effective for listed option market screening, local vol like PDE based calibrations and layer path dependent products like target redemption notes or ratchets."

"In the medium term, NAG may become an interesting provider of math algorithms on the GPU," says Egloff. "In the middle, third party pricing library vendors continuously extend their GPU coverage. Last but not least banks have understood the benefits of GPUs and how they can fit into their IT architecture. They work from the top end of the business requirements, the data preparation and processing, and rethink the IT architecture to utilize the power of GPUs. In this process they need to change their data models and workflows, and extend their IT systems, such that for example a batching of pricing request becomes feasible. Today GPU applications are mainly dedicated isolated solutions. In two to three years we'll see bigger changes and more widespread GPU applications."

**Figure 1. Acceleration (CPU execution time / GPU execution time) of a cancelable PRDC swap model for various grid sizes. The hardware specifications for this example are: Dell Precision T7500 workstation with Intel Xeon X5677 @ 3.46 GHz, dual socket CPU, 24 Gb of memory, and a single NVIDIA Tesla C2075 GPU. The testing environment was Windows 7 Enterprise 64-bit OS, MS Visual C++ 9.0 (2008 SP1) 64-bit, CUDA 4.0 64-bit, with full double precision code compiled for CUDA compute capability 2.0.**




PDE solutions where accelerations of about 15X to 40X have been obtained, depending upon dimensionality and grid sizes. As with serial (CPU) PDE code generation with SciFinance, users have complete control over discretization schemes, solvers, grid building, boundary conditions, and the like through built-in keywords. Alternatively these choices may be defaulted to SciFinance's PDE solution knowledge base.

As an example of the acceleration obtained, we use a cancelable knockout power reverse dual currency (PRDC) swap. The model features Gaussian dynamics for both interest rates and lognormal dynamics for the exchange rate (FX) including local volatility. While there are more complex models in use, this one is relatively standard and useful for comparisons.

Figure 1 shows the acceleration (CPU execution time/GPU execution time) obtained for three different interest rate grids, 32x32 to 128x128, and for each of two FX grids, 128 and 256. The GPU timings are for the entire pricer code (initialization + memory transfers + GPU kernels). For the smallest grids GPU acceleration is less significant. The GPU kernels are so fast that it is difficult to fully amortize the initialization and memory transfer overhead. The larger grids, which are probably of more interest in terms of accuracy, show accelerations of about 25X to 30X as GPU kernel times dominate the total.

## Notes

1. Su, Q. and Randall, C. 2007. General Monte Carlo Greeks in Practice. *Wilmott* magazine, November.
2. 3D grid of thread blocks support was introduced in CUDA 4.0