# Tipping point: The performance benefits of GPUs put an elephant on the scales

*Graphics processing units are set to finally tip the balance in favour of rewriting legacy mathematical code. Murex shares the benefits of such an in-depth overhaul – gains in performance, a standardised programming language, easier optimisation and improved precision – which have placed the solutions provider ahead of the game and fully GPU-compliant*

Organisations including software vendors, banks and insurance companies that produce and maintain code are enticed on a regular basis to rewrite their legacy mathematical code with the aim of optimising it or adapting it to recent technology advances in machine hardware or coding languages.

Initiatives of this kind rarely see the light of day when the benefits are compared to the eventual cost of implementing it. However, the availability of graphics processing units (GPUs) will finally tip the balance in favour of an in-depth overhaul of code. In our opinion, with the aid of GPU, there are four main reasons for code to be rewritten:

- First, the performance gains are incomparable to those achievable today, even those that can be envisaged in the coming years.
- Second, the one-year-old standardised coding language (OpenCL™) shows that GPU is a sustainable development field.
- Third, a closer relationship between the code and the machine makes code optimisation easier.
- Last, but not least, a quasi-paradox: more precise numerical algorithms that are commonly used in financial mathematics.

**Performance gain**

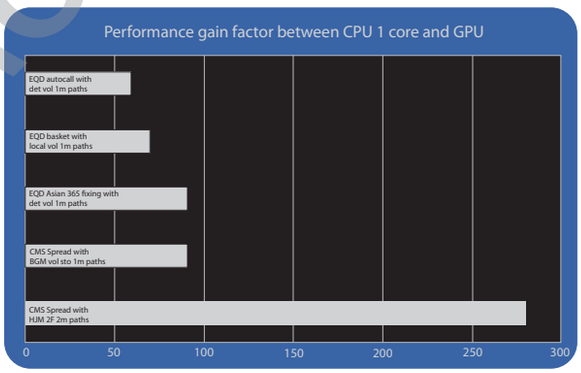
Monte-Carlo simulations are the obvious beneficiaries of GPU technology. The financial world is a heavy user of the Monte-Carlo method for exotic pricing – path-dependent products, multi-asset payoffs or, quite simply, models that have more than three factors – and, for other fields like value-at-risk, potential future exposure.

Such a resolution method is particularly well-suited to GPU architecture because:

- each path performs the same operations;
- each path is independent from other paths; and
- the calculation of the paths' average and their distributions can also be parallelised.

Depending on the type of model or underlying (for example, single stock or constant maturity swap index), performance gains using Murex analytics range between 60 and 250 times faster (see figure 1).

1 Monte Carlo performance gains based on different underlyings and a comparison of models using a Xeon CPU and NVIDIA Tesla C2050 GPU




Today, GPUs are significantly faster than central processing units (CPUs) and this situation will not change in the coming years. Even more, the gap between GPU and CPU continues to widen, as it has over the past three years. Throw grid computing into the mix, and performance gains can be multiplied with a grid server of GPUs.

**What can be done with all this spare capacity?**

Big market players known to be early adopters of this solution have taken advantage of GPUs to execute near-time heavy risk analysis, previously only possible on a daily basis, and have pushed Monte Carlo over a million paths to have stabilised Greeks.

**Partial differential equation, above all else**

Our first-preference resolution method is analytical formulae and the default choice is evidently partial differential equation (PDE); Monte-Carlo simulations are a last resort.

Indeed, with the same amount of calculations, the PDE will always be more precise with smoother sensitivities. This resolution

method is particularly well-suited to high-volume first-generation exotic payoffs like auto-callables and Bermudan swaptions. Moreover, merely adapting the Monte Carlo for GPU will lead us to heterogeneous hardware architecture – different hardware for PDE and Monte-Carlo calculations.

Nonetheless, calculating the PDE on GPU faces two main challenges:

- First, one PDE is not enough to make full use of GPU cores.
- Second, normal resolution methods for PDE rely on an inverted tridiagonal matrix – which typically cannot be parallelised – like, for example, the Gaussian elimination method.

The first challenge can be addressed by modifying the calculation sequence: the present value of the payoff and its numerous sensitivities will be evaluated simultaneously rather than sequentially. The resolution of the second challenge relies on new matrix-inversion techniques like parallel cyclic reduction[1] (PCR), albeit more computationally intensive, these techniques can be parallelised.

The resulting performance gain is in the region of 40 for one-dimensional PDEs and eight for two-dimensional PDEs. In addition, NVIDIA's latest Tesla 20-series Fermi provides particularly promising capabilities for PDE resolution using GPU due to the increased shared memory and higher double-precision performance.

2 Inversion of a tridiagonal matrix with the PCR method

$$
\begin{array}{rrrrrrrrrrrrrr}
2a & - & b & & & & & & & & = 1 & & \times \tfrac{1}{2} \\
-\,1a & + & 2b & - & 1c & & & & & & = 1 & & \times 1 \\
 & - & b & + & 2c & - & d & & & & = 1 & \times \tfrac{1}{2} & \times \tfrac{1}{2} \\
 & & & - & 1c & + & 2d & - & 1e & & = 1 & \times 1 & \\
 & & & & & - & d & + & 2e & - \; 1f & = 1 & \times \tfrac{1}{2} & \times \tfrac{1}{2} \\
 & & & & & & & - & 1e & + \; 2f - \; 1g & = 1 & & \times 1 \\
 & & & & & & & & & - \; 1f + \; 2g & = 1 & & \times \tfrac{1}{2}
\end{array}
$$

$$
\begin{array}{rrrrr}
+\,1b & - \tfrac{1}{2}d & & = & 2 \times \tfrac{1}{2} \\
-\,\tfrac{1}{2}b & + 1d & - \tfrac{1}{2}f & = & 2 \times 1 \\
 & - \tfrac{1}{2}d & + 1f & = & 2 \times \tfrac{1}{2}
\end{array}
$$

$$
\tfrac{1}{2}d = 4
$$

## A standardised language

The release of OpenCL™ in April 2009 as a standardised programming language on GPU is a real breakthrough. Derived from CUDA C, the language developed by NVIDIA, Open CL has become a market standard. It brings sustainability to the GPU solution by liberating it from hardware-vendor dependencies, even rendering it compatible with multicore CPU x86 and Power.

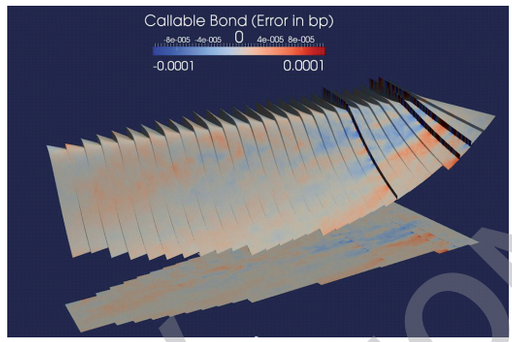
The production cycle of software (code-writing, functional validations, documentation, maintenance, support team training), being slower than that of hardware vendors, was sine qua non to massive investment in a development team – a condition that has now been satisfied.

Finally, all original equipment manufacturers such as Hewlett Packard, IBM® or Dell™ are now integrating passively cooled GPU modules into their servers, in compliance with all data-centre requirements. In effect, GPU is no longer reserved for teenage gamers, but has become an industry-ready, battle-tested, sustainable product.

## Closer to the machine

Programming in typical programming languages that run on CPUs hide fundamental complexities as they depend on the compiler – the use of the machine is not explicit. Such programs are therefore difficult to optimise apart from the optimisation of the algorithm

3 CPU versus GPU on a one-dimensional PDE for a callable bond




itself. Relying on the compiler is not sufficient. This is not the case for programs using GPU, despite – and thanks to – the fact they are more difficult to write since they are closer to the hardware, they become easier to optimise.

Keep in mind three areas where the benefits are most evident: cache management, synchronisation and explicit operations on vectors.

## More precision

The quasi-paradox: allowing the use of computationally intensive but more precise algorithms in PDE while still benefiting from performance gains. As an illustration, take the common example in finance of a two-factor model solved in PDE. The use of alternating direction implicit/vector-splitting techniques is required to eliminate cross-term derivatives. Such predictor/corrector techniques can lead to poor precision of the resolution method.

More precise methods exist, such as iterative methods, but are more computationally costly. However, such methods can be parallelised and can therefore run on GPU, still giving performance improvements. At the end of day, we have our cake and eat it too: more precision and better performance for two-dimensional PDE.

Two iterative methods have attracted our attention to date:

- Multigrid[2] – the use of a smaller (coarse) PDE to resolve a large (fine) one
- The Schwarz method[3] – sandwiching the PDE grid in sub-grids and resolving them in parallel.

## Conclusion

Dividing hardware cost by 10, power consumption by 20 and valuing financial products 200 times faster sums up the amazing benefits of going massively for GPUs and explains why Murex has undergone such an important effort for the past three years. We have already rolled-out this technology for exotic products and potential future exposure and will soon include vanillas.

Murex will release the result of this work at the GPU Technology Conference 2010, organised by NVIDIA in San José from 20–23 September, under the title *Practical Methods beyond Monte Carlo in Finance*.

More info: gpu@murex.com

[1] *S Yao Zhang, Jonathan Cohen & John D Owens, "Fast tridiagonal solvers on the GPU", in* Proceedings of the 15th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming (PPoPP 2010), *pp. 127–136, January 2010. DOI: 10.1145/1693453.1693472*

[2] *William L Briggs*, A Multigrid Tutorial

[3] *Martin J Gander*, Schwarz Domain Decomposition Methods in the Course of Time, *University of Geneva, February 2009*