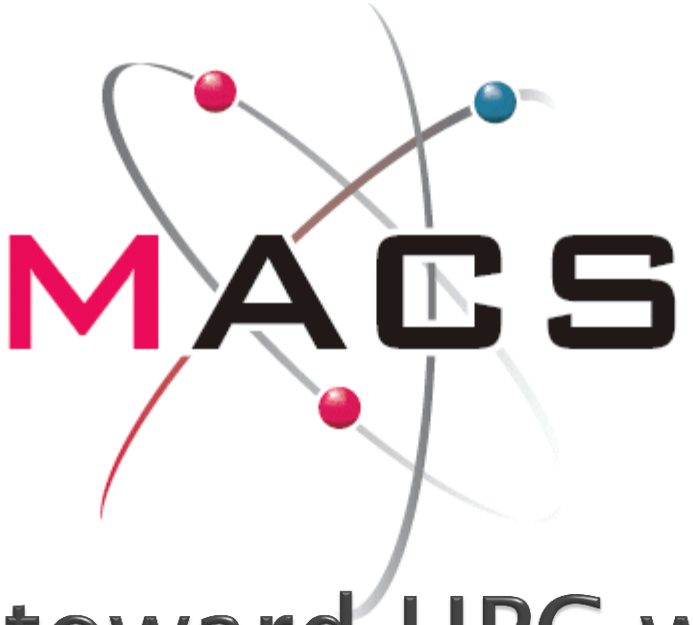



# Finance toward HPC with GPUs



NVIDIA GTC May 16th 2012


DUBLIN
MOSCOW
PARIS
BEIJING
NEW YORK
BEIRUT
SEOUL
TOKYO
SINGAPORE
SÃO PAULO
SYDNEY

# Recap from September 2010 GTC

- Murex is ready for production for all its Monte Carlo with NVIDIA Tesla GPUs.
  - The huge speed up obtained democratizes the usage of MCs, reduces the TCO of the application, while enabling the user to obtain smooth enough second order derivatives
  - Flexibility is not a problem and payoff languages can be adapted
  - Benchmark usual cases evaluation time far under the second but speed can be limited by the amount of available memory
- PDEs are a promising subject thanks to the Fermi architecture and the PCR algorithm but one needs often to run several PDEs in // to use the full power of a single GPU
- Full implementation done using OpenCL™ 1.0
- We can learn a lot from the supercomputing industry

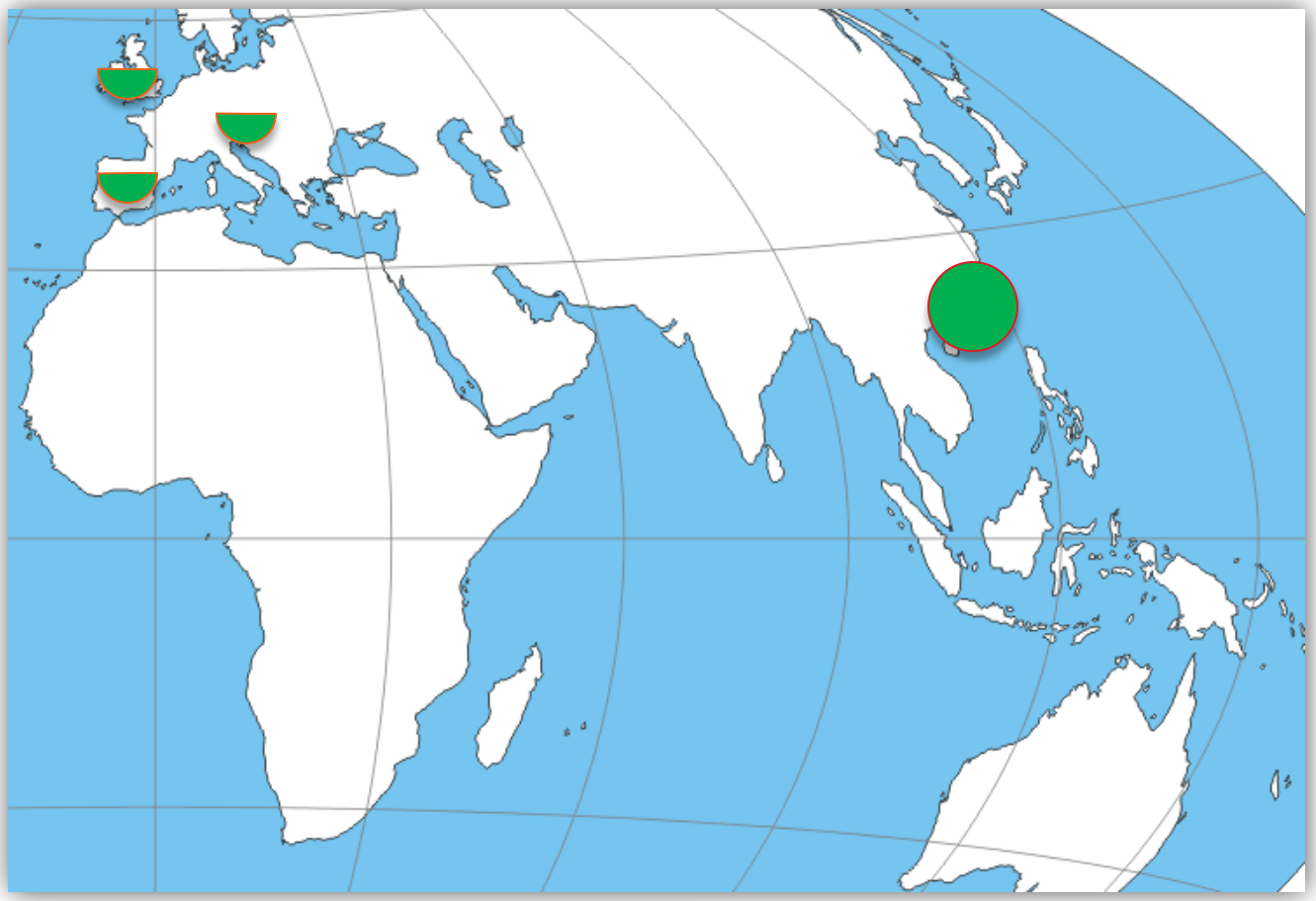

Time to compute over 150 sensitivities of a cancellable range accrual on a CMS 30Y & 3M basis (30 year Constant Maturity Swap, reset quarterly) priced with a Markov functional PDE model. The SABR input smile, mean reversion and term structure of volatilities are numerically recalibrated for each market data bump.


Time to price a 15 year cancellable range accrual on a CMS (Constant Maturity Swap) Spread, using a 2-factor HJM (Heath-Jarrow-Morton) model with 1 million Monte Carlo simulation paths. The model uses a full term structure for volatilities and includes calibrating the correlations.

"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 with the eventual cost of implementing it. However, the availability of GPUs will finally tip the balance in favour of an in-depth overhaul of code," predicts Pierre Spatz, head quant at trading and risk system supplier Murex in Paris

# 2011 What's new

# We are live with Macs & GPUs



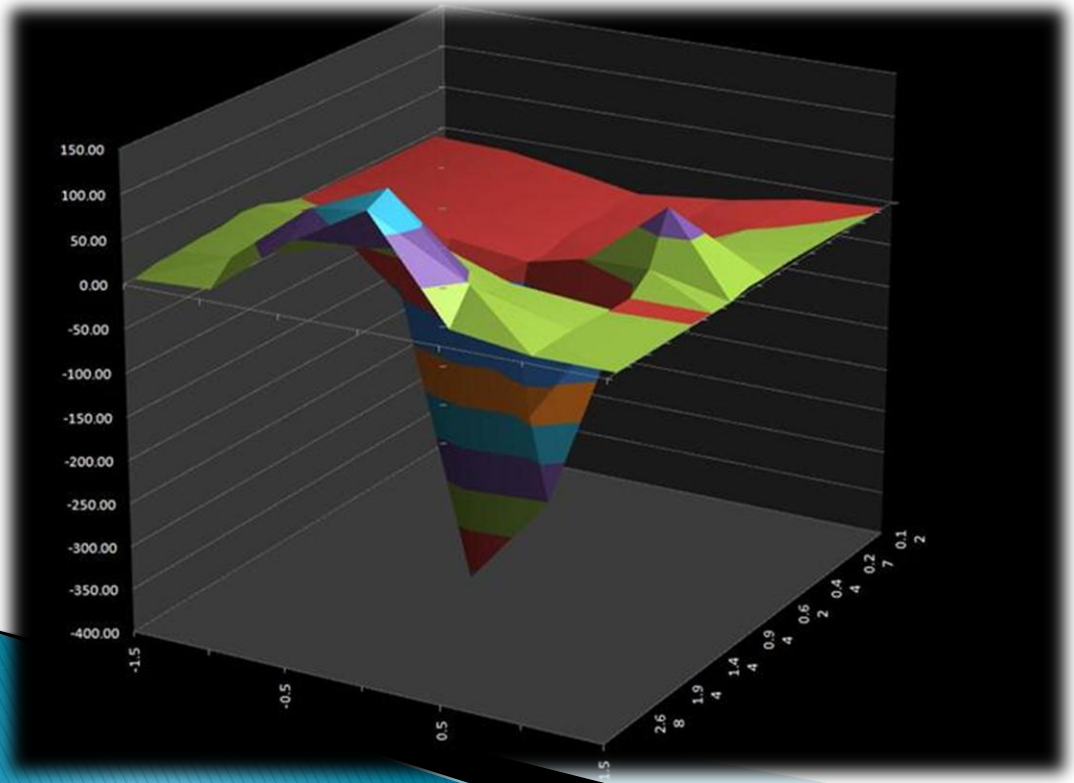
Live

Already Scheduled Or UAT

# 2011 Achievements

## Calibration of local volatility

- Direct application of 1D PCR PDE Solver
  - Solve for the central point using only 1 SM
  - Solve for sensitivities in // assigning 1 or several scenarios to each SM
- Mixed with fast // PDEs & MCs on GPU one can process in a few seconds a local volatility trade with many numerical sensitivities including super vega bucket




till

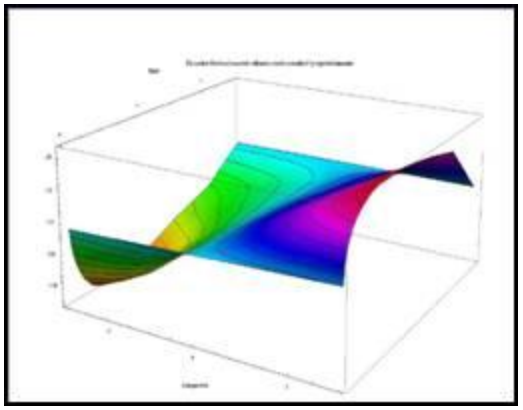
X 80

Legacy 1 core
& no SSE

# 2011 Achievements

## Calibration of the markov functional model

- Markov functional is equivalent to the local volatility model for 1 factor interest rates avoiding adjusters or other risky tricks
- Many of our customers are eager to replace HW by Markov but its calibration is purely numerical based – Iterative gauss Integrals  with many exponential embedded – and its performance was often considered as a show stopper
- A Full 2 level calibration – caplets & co-terminals – is solved in far less than a second on a GPU but is not compute intensive enough to feed one big GPU
- Here again, solve several calibrations for premium and sensitivities on the same GPU to obtain the best performance.



till

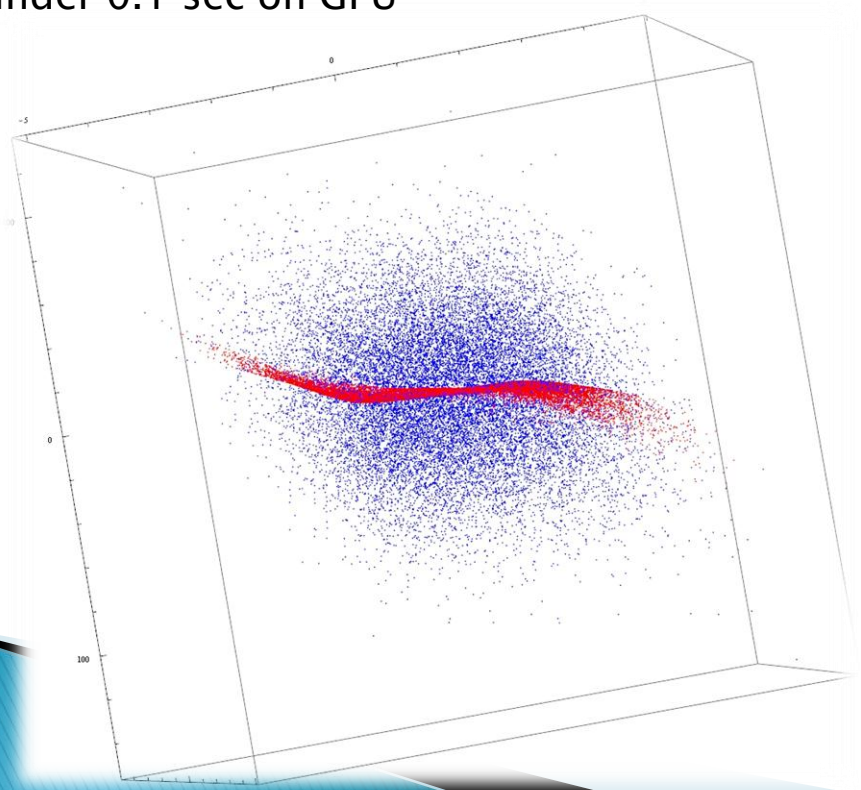
X 90

Legacy 1 core
& no SSE

# 2011 Achievements

## Local regression for early exercize & PFE – NBody like –

- Payoff shape agnostic method that works till 3D x Values and which does not create fancy extrapolations
  - Clusterize the stochastic cloud of points
    - Bandwidth driven
  - Build many interpolation functions instead of one regression function
    - Compute driven
- Unusable on CPU – 20 sec for the worst case – against under 0.1 sec on GPU



1D
X 100
Legacy 1 core
& no SSE

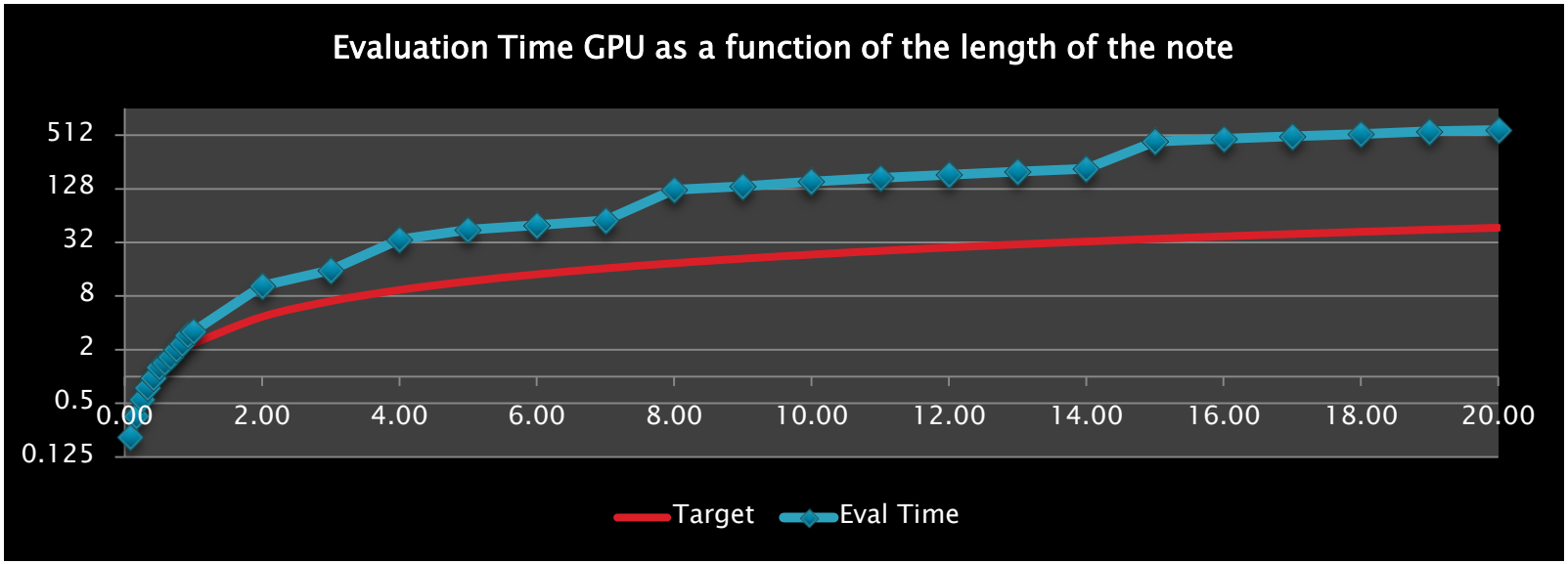
3D
X 300
Legacy 1 core
& no SSE

# Is it enough ?

# Another optimization story

## Analysis of a bad simple test case (I)

- N Years note paying a fixed coupon when the weighted spread between two assets performances crosses a barrier with daily checking or otherwise at expiry
- 1M Paths and a Black Scholes like model to reduce compute intensity and highlight the problems

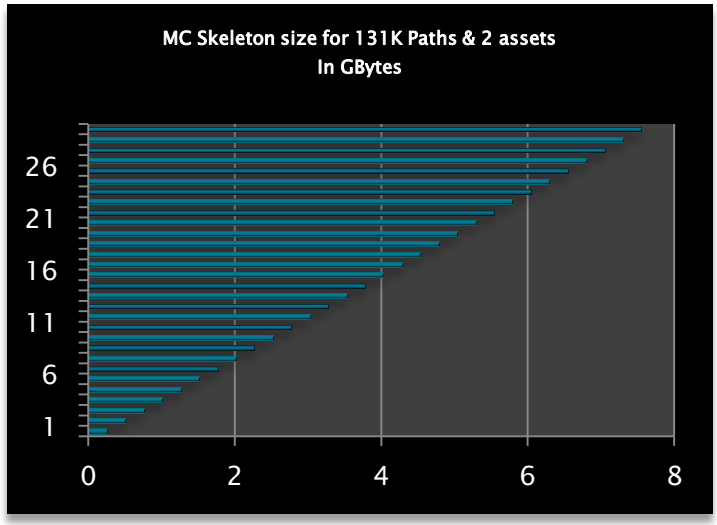


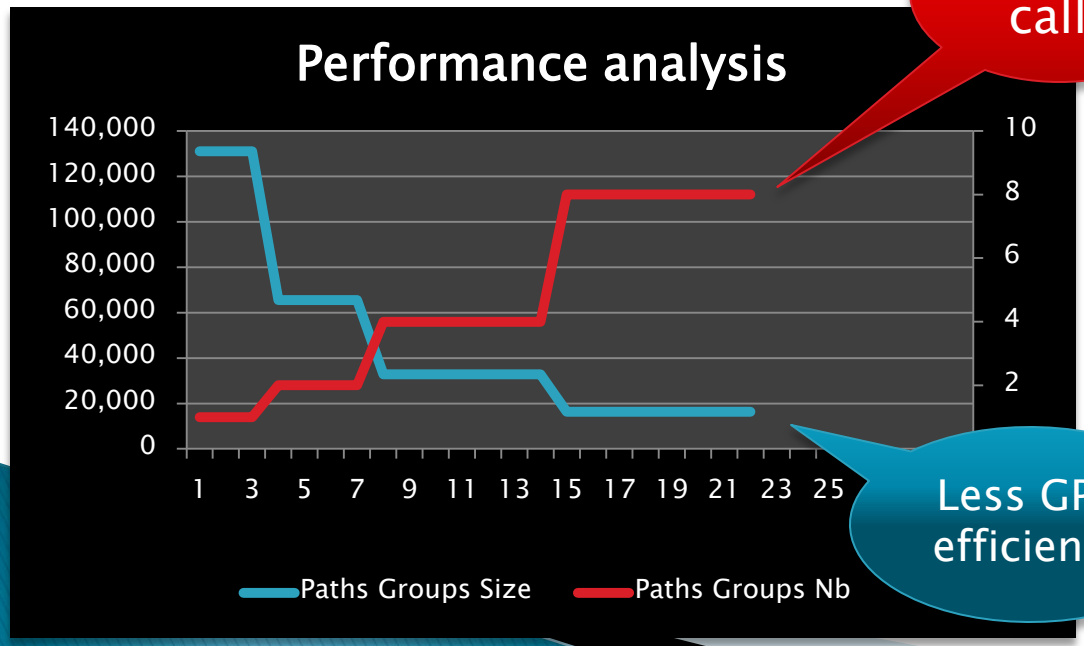

- The payoff has a huge number of fixings so we expect an evaluation time proportional to the length of the product – which is already bad – but we obtain something worse !

# Another optimization story

## Analysis of a bad simple test case (II)

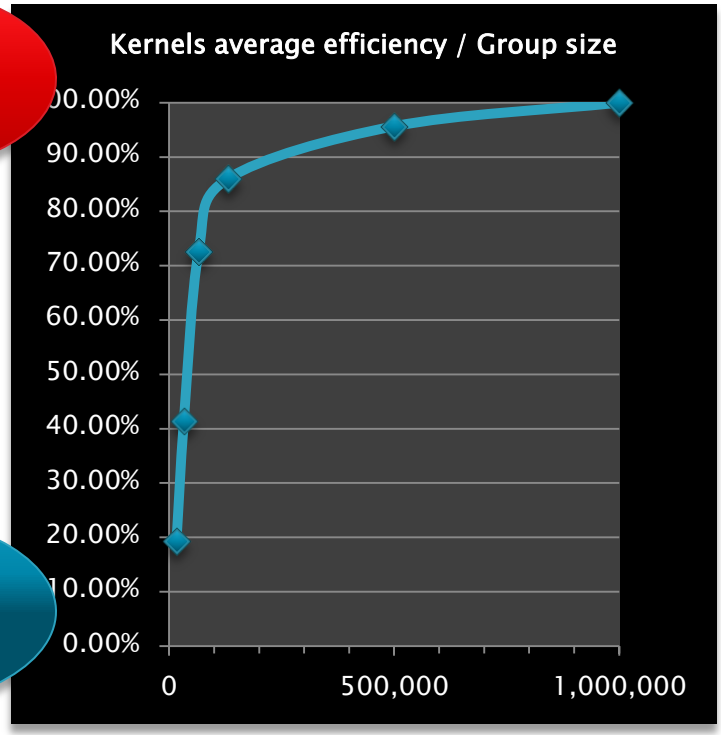
- It is not very compute intensive
  - Exp, random numbers and Brownian bridge are the only heavy kernels
- It is memory bandwidth intensive and GPUs are better than CPUs at this level but not tremendously
- It is global memory bound – skeleton kept in memory to be compliant with scripting per date and Full Brownian bridge –
- The size of the parallel groups has to be decreased while increasing the number of fixing dates while the number of calls to the scripting language and the kernels has to be increased since they are called for each group of paths




More calls

Less GPU efficiency



# Solution Part I

## Be hybrid

```
Host to Device Bandwidth, 1 Device(s), Paged memory, direct access
   Transfer Size (Bytes)        Bandwidth(MB/s)
   33554432                     3027.1

Device to Host Bandwidth, 1 Device(s), Paged memory, di
   Transfer Size (Bytes)        Bandwidth(MB/s)
   33554432                     2514.9
```

Independent of the length of the product

- Till now we have used the host only for the sequential code and reduced to the maximum PCI express usage by keeping in GPU memory everything. It has worked fine for most cases but we have reached the limits.
- **Let's use the host as a memory buffer.**
  - Build as much paths in GPU memory and store them in the host while computing the next ones to hide PCI express latency. The whole skeleton can be easily stored in the host.
  - Once done load the paths subparts by group of dates while computing the payoff on the previous dates. Dates are represented by vectors and are far smaller than the skeleton. We can treat all the paths at the same time and find back the same level of efficiency as with a reduced number of dates.
- Enable us to keep exactly the same algorithm as the production one
- Easy to implement on paper but harder in practice because the memory is not well ordered between the different tasks either on GPU & CPU
- But it breaks the barriers relatively to the global available memory. It is only a question of buying a host with enough memory

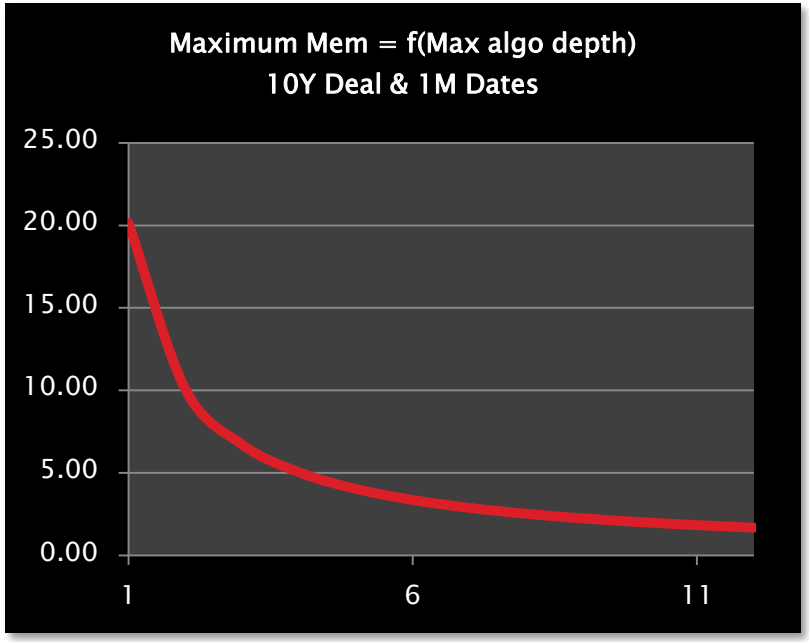
| Asset or Variable size for 1 date In MBytes | |
|---|---|
| 131K Paths | 0.52 |
| 1M Paths | 4.19 |

1K smaller than the skeleton

# Solution Part I Better Alternative

## Change the algorithm for Mem to be in O(Path*Log(Dates))

- Instead of generating all the random numbers at the same time
  - generate first a skeleton with BB for N intermediate dates $\{D_i\}$.
  - Then in the forward step generate by group between $D_i$ and $D_{i+1}$ and forget the past but keep dates for $D_i$.
- **Pros & Cons** :
  - Implementation of American option is harder
  - It enables us to multiply by N the size of the groups when N is small enough
  - It can be mixed with previous solution if we have so many dates or assets or paths that the host memory will be overloaded
  - The storage and retrieves are easier since the values stored are already well ordered in memory
- **Careful** :
  - Random number generation order will depend of the depth of the algorithm and the available memory card. Use a **M**assively **P**arallel **R**andom **N**umbers **G**enerator like CURAND XORWOW – or DE SHAW PHILOX – **billions** of independently generated series – to obtain the same value on different hardware
- Solution applicable to many models which need small discretizations like local volatility or SV-BGM

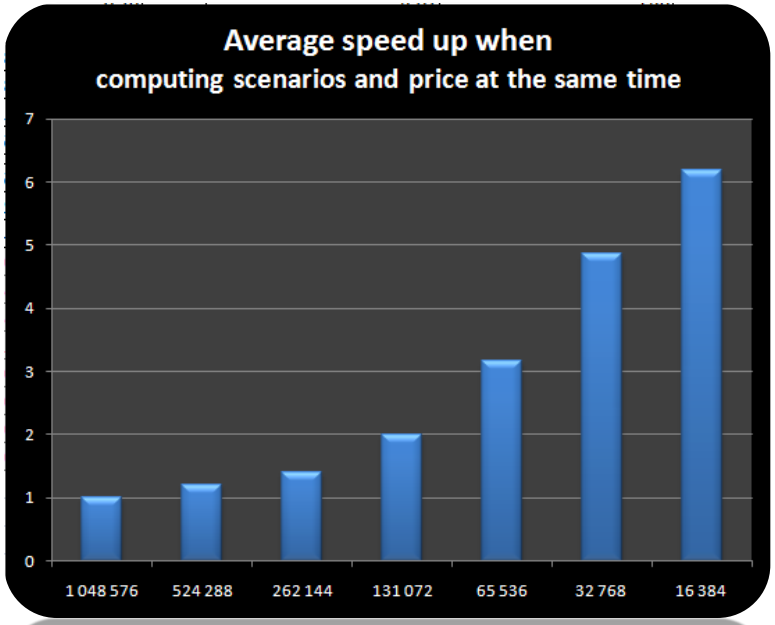



| Evaluation time 1M Paths | 10Y note | 20Y note |
|---|---|---|
| Legacy GPU | 50.5 | 140.6 |
| Target | 15.3 | 30.2 |
| Achieved | 15.3 | 30.3 |

# Solution Part II

## Efficient usage of the GPUs

- The issue of the memory is solved and we can treat all paths in one call most of the time
- We have seen that the GPU is not even fully efficient when we have 100 000 paths and that 500 000 or 1 000 000 is better but we often need far less
- We know also that next GPU generations will have more cores and that this efficient number of Paths will only increase with time




**Let's be ready today & compute enough sensitivities or scenarios inside the same MC run to feed the GPU**

Monte-Carlo Skeleton for N Paths
& a subset of M dates

Keep M big enough
to reduce number of calls

Reuse skeleton as much as possible to increase compute intensity

Local variables of size N
We can have many

Kernel launch time is the same for a small or a big N

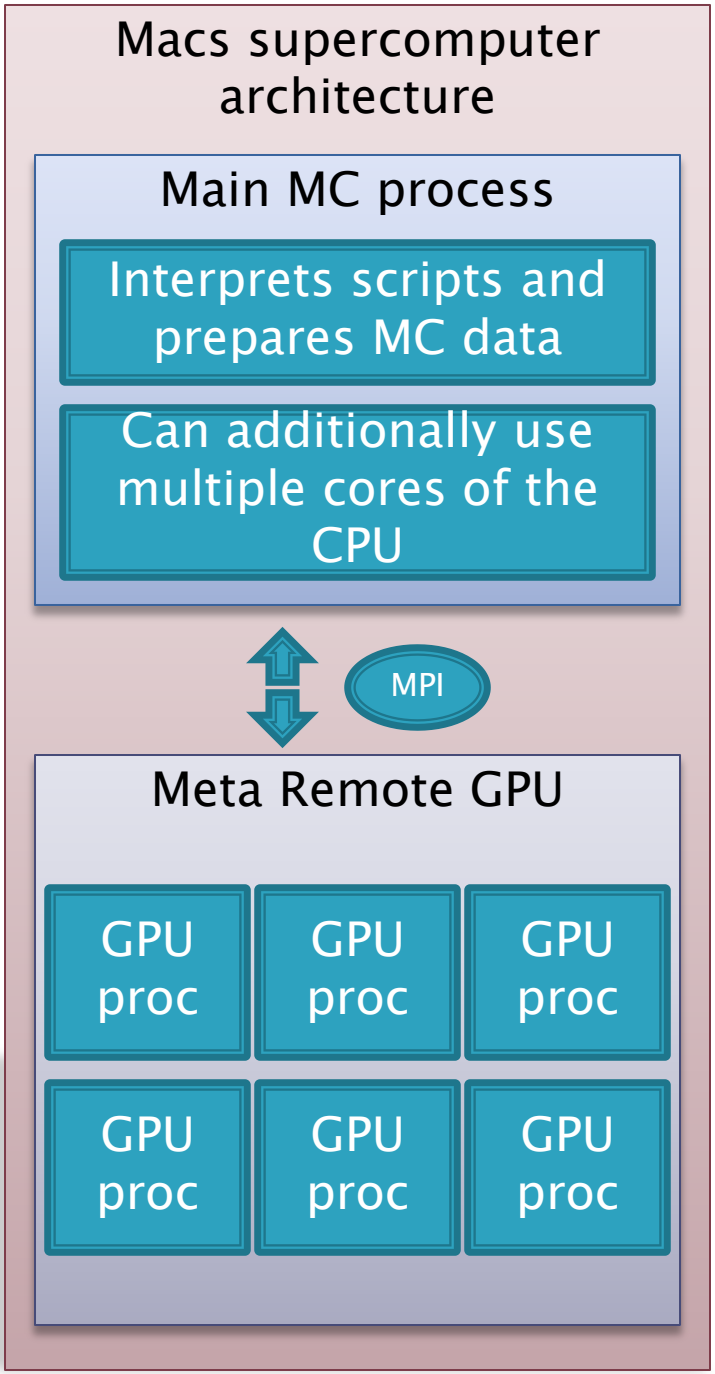
Global memory usage for a Monte Carlo

# Solution Part III

## Multiple GPUs

- To be fully efficient the GPU code needs to receive queries with premium & scenarios
- If there are too much scenarios the evaluation time will go down at some stage
- Split the task on the grid could work but we would
  - Redo common things like script evaluation
  - Loose some parallelization opportunities
  - Have Issue reusing central point information so useful for Greeks
  - Add latency by going back and forth on the network – mind the usage of clouds – and add useless message encryption/decryption

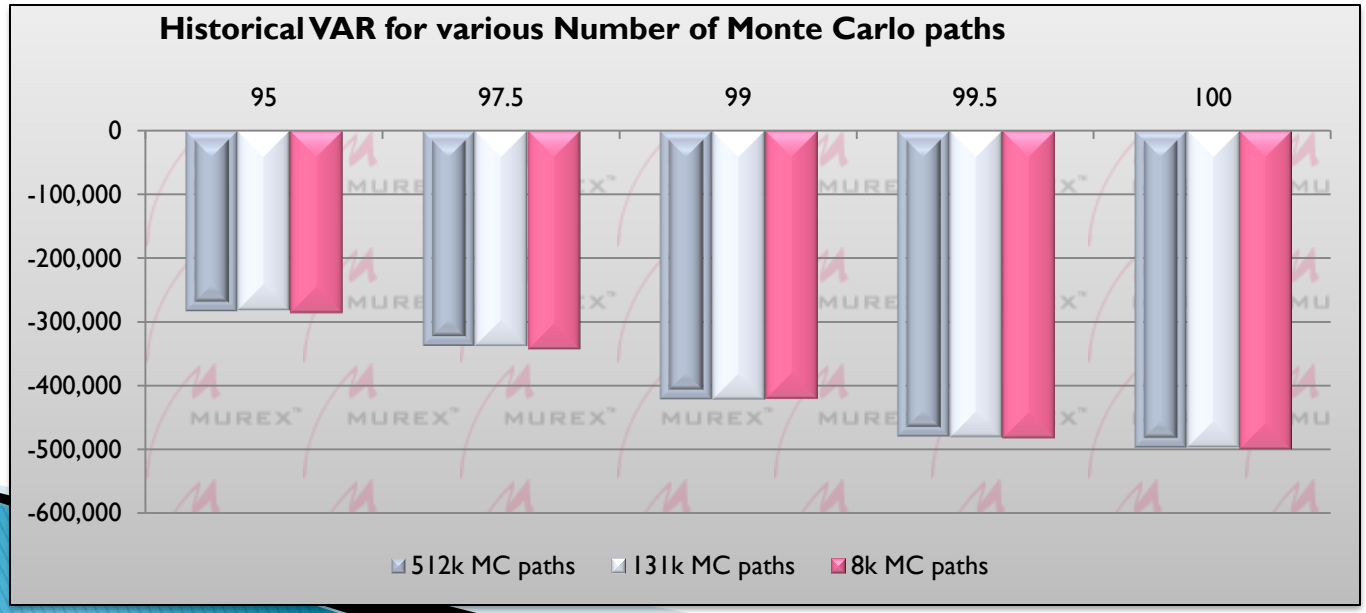
- Instead use several GPUs
  - Straight forward extension of previous solution with several GPU MPI slave processes
  - Slave processes only run the OpenCL/Cuda kernels and are very light
  - Non blocking communication
  - Solution works with clusters and not only multiple GPU computers
  - Increase tremendously
    - The amount of onboard memory
    - Bandwidth
    - Compute
  - Mind again random number generation to stay scalable



# GPU Cluster Benchmark

## Front office Risk & VAR on a real case

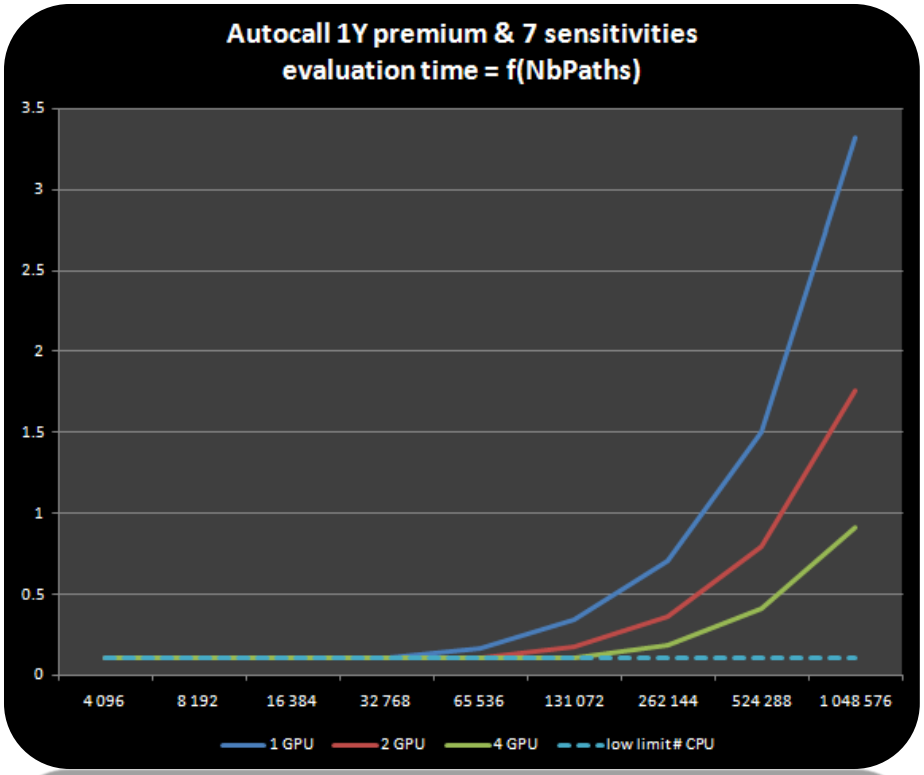
- Daily Autocall 1Y on 2 assets with Monte Carlo
  - A structure very standard in Asia and Europe
  - Our customers want to have more than 1 000 of these in there portfolio and risk manage them in a quasi real time way without investing in more than a rack of low cost computers
- For front office risk management they will use between 100K & 200K scenarios
- For VAR 8K scenarios are enough since the premium only matters
- Hopefully the bank would like to have a real time intraday incremental VAR in a way to define limits and pre-deal check on the VAR

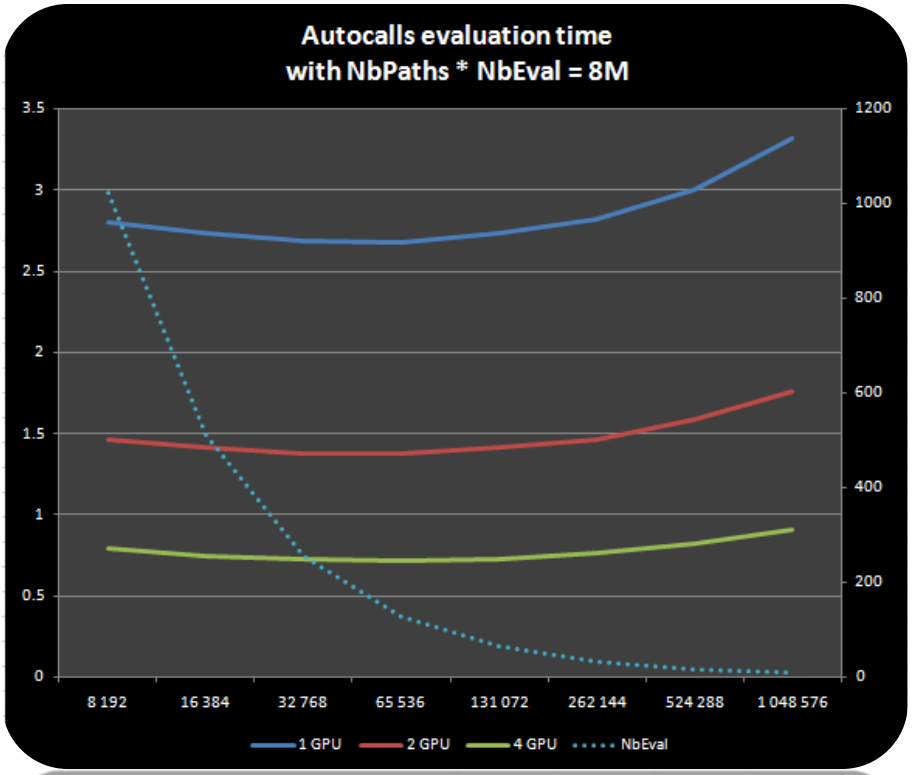


# GPU Cluster Benchmark

## time in secs per deal

Full solution Available Now




- In less than a second traders
  - can go easily to 1M paths for the computation of their risk
  - check their IncVAR on more than 1K historical scenarios

# Gedankenexperiment

## Back-testing

- Our first GPU server in May 2009 was a Supermicro/Bull branded twin x86 server linked to a Tesla 1070 box with 4 GPUS
- Its not considered as best of breed today but we could have achieved very good speed already using MPI before Fermi & Kepler for cases which do not need a lot of double precision


| | 1 Tesla C1060 | 1 M2090 | 1 Tesla S1070 | 2 M2090 |
|---|---|---|---|---|
| Cuda Cores | 240 | 512 | 960 | 1024 |
| Single Precision GFlops | 933 | 1500 | 3732 | 3000 |
| Memory GB | 4 | 6 | 16 | 12 |
| Memory Bandwidth GB/sec | 102 | 200 | 408 | 400 |
| **Autocall 1Y Premium & 7 greeks** | 8.04 | 3.31 | 2.09 | 1.76 |

# Recap, conclusions & perspectives

## GPU usage is a real change of paradigm for finance

- We have broaden widely the usage of GPUs to a brand new set of problems from vanillas, to calibration & V@R
- While regulators impose more & more compute intensive monitoring procedures, we consider that the fight for performance is the only way forward if we want to keep financial innovation going
- Thanks to broad optimizations and the usage of clusters we can tune the hardware to give whatever results in quasi-real time at a low TCO while the focus has changed from Grids that give the throughput toward real HPC & Clusters that give throughput but also minimum latency
- Thanks to kernel isolation and MPI the MACS architecture is ready to scale again with the next generations of GPUs to come but also with greener & hybrid solution like the one exposed by the Carma & Mont Blanc Projects

| | |
|---|---|
| CPU | NVIDIA Tegra 3 Quad-Core ARM A9 |
| GPU | NVIDIA QuadroTM 1000M with 96 CUDA Cores |
| Memory | • CPU Memory: 2 GB<br>• GPU Memory: 2 GB |
| Peak Performance | 270 Single Precision GFlops |
| CPU – GPU Interface | PCIe x4 Gen1 link |
| Network | 1x Gigabit Ethernet |
| Storage | 1x SATA Connector |

# Q & A