GROUP RISK MANAGEMENT

High Performance Counterparty Risk and CVA Calculations in Risk Management

Dominique Delarue
Azim Siddiqi
20th March 2013

Diversity Integrated Global Long term vision Transversality
Independent Anticipation Expertise State-of-the-art

Overview

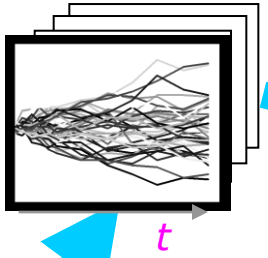
- Counterparty Risk Primer
- The Problem
  - Scale
  - Scale Explosion
- The Solution
  - Challenges
  - Pricing Ecosystem
  - Breaking the Cube
- Moving to Kepler
- Summary

Counterparty Risk for Dummies

Market data & Calibration parameters

	EUR	USD	HKD	NZD
EUR	1.000	0.874	6.816	2.075
USD	1.144	1.000	7.800	2.374
HKD	0.147	0.128	1.000	0.304
NZD	0.482	0.421	3.285	1.000
CAD	0.719	0.628	4.899	1.491
AUD	0.592	0.517	4.032	1.227
GBP	1.637	1.430	11.157	3.396
CHF	0.674	0.589	4.593	1.398
NOK	0.128	0.112	0.875	0.266
DKK	0.135	0.118	0.917	0.279

Market Scenarios



(1) Market Parameter Simulation

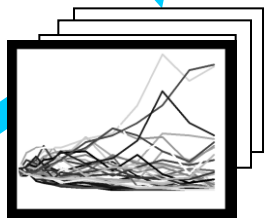
FX rates, interest rates, equities, commodities, credit spreads and ratings

(2) Revaluation of deals

- mostly full reval
- 'Norms' for exotics

Deal data

FWFXS	05/08/1997	0	2929553	USD
ASTSW	05/12/1997	5	20000000	DEM
AMORT	15/06/1998	6.4	562129	USD
AMORT	28/08/1998	6.6	7500000	USD
IRSWP	15/10/1998	7.9	300000000	HKD
EATOP	21/07/1999	0	1000000	DEM
AMORT	24/08/1999	0	50000000	DEM
AMORT	23/09/1999	0	267380	USD



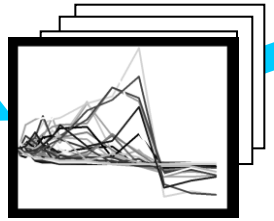
Simulated Deal PVs

Netting, Collateral & Referential data

FORD MOTOR CREDIT COMPANY	US
FORD MOTOR COMPANY BRASIL LTI	BR
FORD TAIWAN SERVICES LTD	TW
LE FORT LIFE LIMITED (FCP AMP HE	AU
FORMOSA PLASTICS CORP	TW
FORTIS BANQUE FRANCE	FR
FORTIS BANK SA NV	BE
FORTIS BANK NEDERLAND NV	NL
FORTIS OPCVM RISKY (CRFI SYSTE	FR

(3) Aggregation & Collateral Simulation

- by Global Credit Line



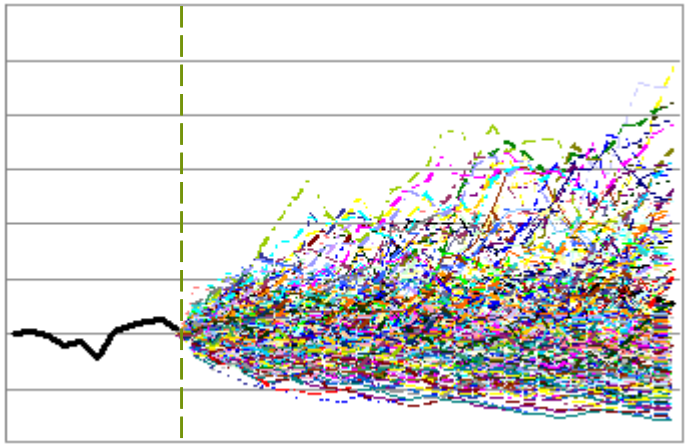
Aggregated PVs

(4) Front-End Analysis Tools

- risk monitoring
- credit adjustments
- *stress analysis*

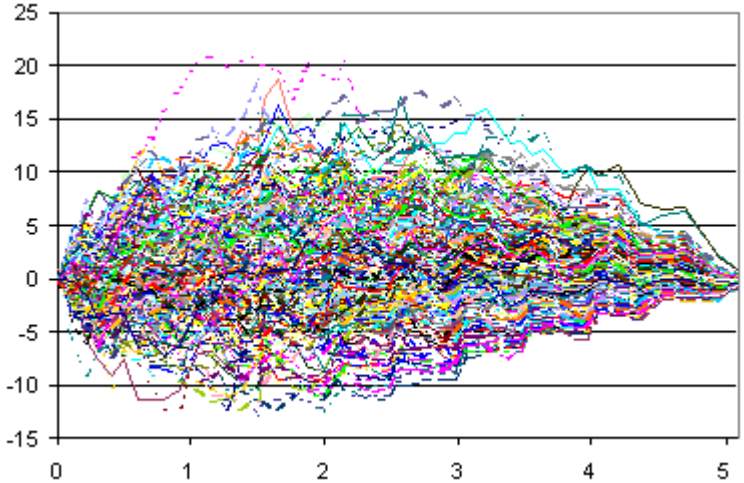
Monte Carlo Process

Market parameter (for ex. Euribor 3 months)



Today

time



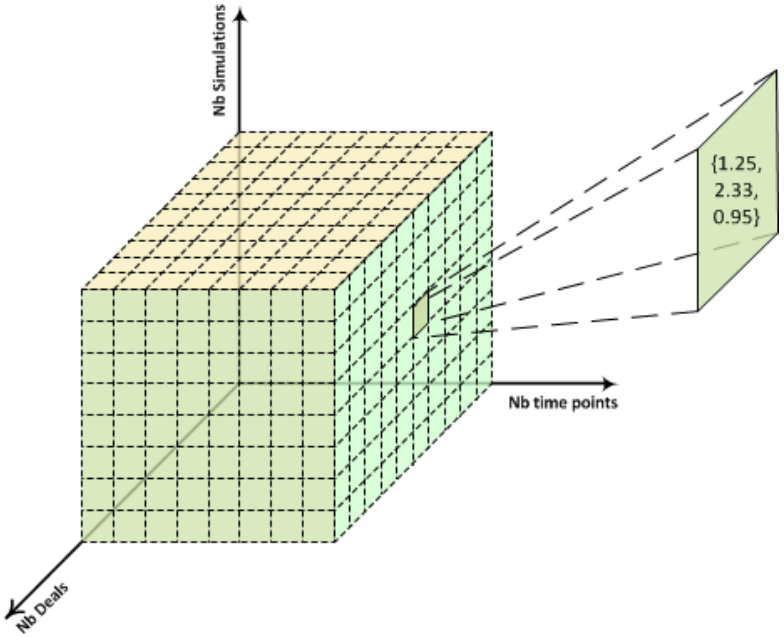
- Simulation process ('Monte Carlo')
  - Paths (values for different future dates) are simulated for all the market parameters that influence the value of the transaction
  - Correlation against other market parameters (IR and FX) is taken into account during the simulation process.

- Re-pricing of the transaction
  - The interest rate swap value is recalculated for the future dates of all the scenarios
  - In some scenarios, the value will be positive (typically when the interest rates increase), in others, the value will be negative

Problem - Scale

- For all counterparties and for each scenario

  1,000,000 x 5,000 x 400
  = 2 trillion calculations
  (8TB of floats)



…and 500 TB of simulations were consumed in this process

But that's ONE scenario. We need HUNDREDS !!!

Problem – Scale Explosion

- Regulatory Requirements
  - More stress testing
  - Detection of Wrong Way Risk
    (e.g. stressing Turkish Lira to assess impact on Turkish banks)
- Risk Management
  - Exposure Explained
    Incremental Stress Scenarios for Complex Stress Scenario
  - Add/Remove deals from counterparties

Problem – Scale Explosion

- CVA Calculation is simple in itself

$$CVA_{Regulatory} = LGD_{MKT} * \sum_{i=1}^{T} \max\left(0, e^{-\frac{S_{i-1} * t_{i-1}}{LGD_{MKT}}} - e^{-\frac{S_i * t_i}{LGD_{MKT}}}\right) * \frac{EE_{i-1} * D_{i-1} + EE_i * D_i}{2}$$

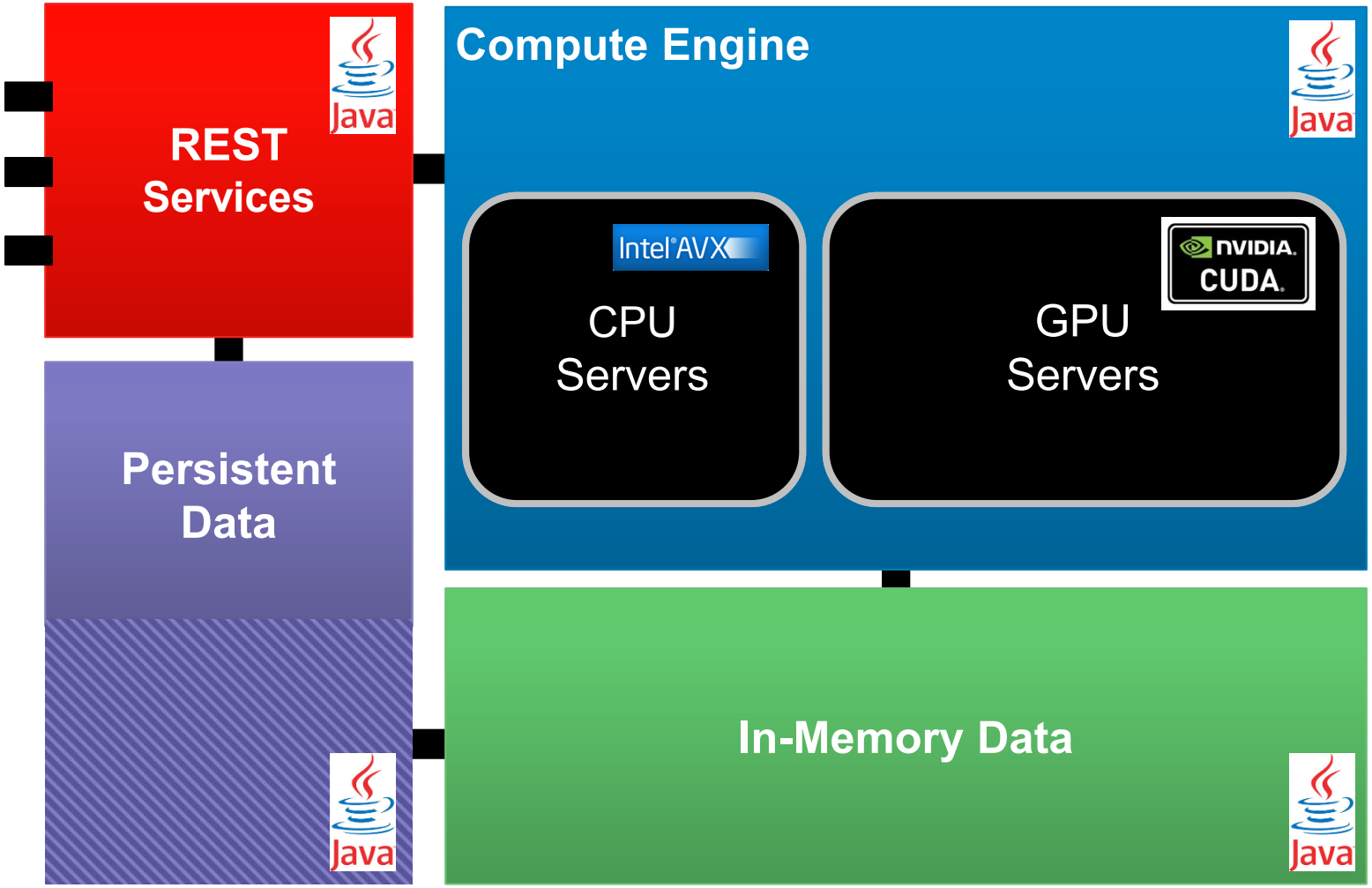
- But CVA needs to be hedged


you're
gonna need
a bigger
boat

Solution - Challenges

- Current Counterparty Risk System
  - CPU Compute Farm (a few thousand cores)
  - Low level of Parallelization
  - C++ Code
- Objective
  - Do at least 50 x more for the same cost
- Re-think of the whole solution
  - Parallelization on multiple levels (counterparty, deal, instrument, market data, time point, path)
  - Re examining data, calculation and latency problems

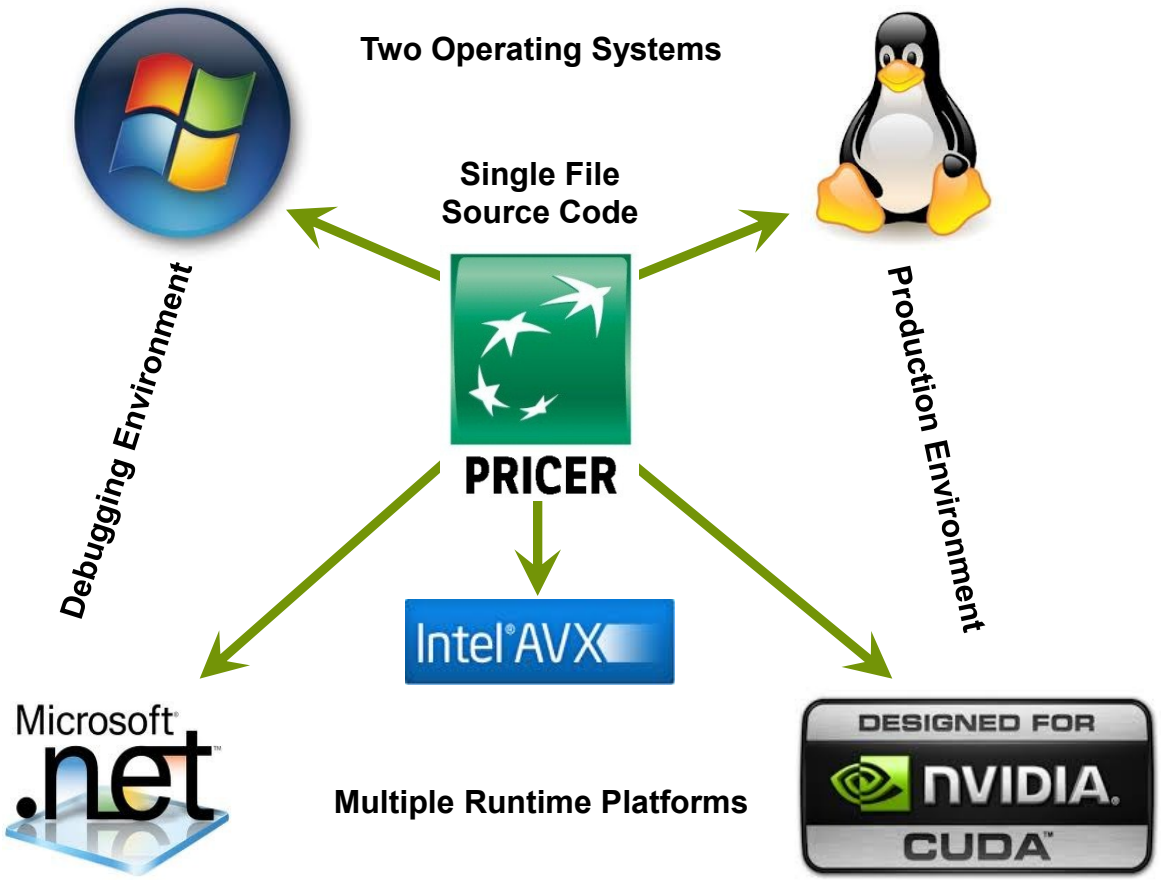
Solution - Overview



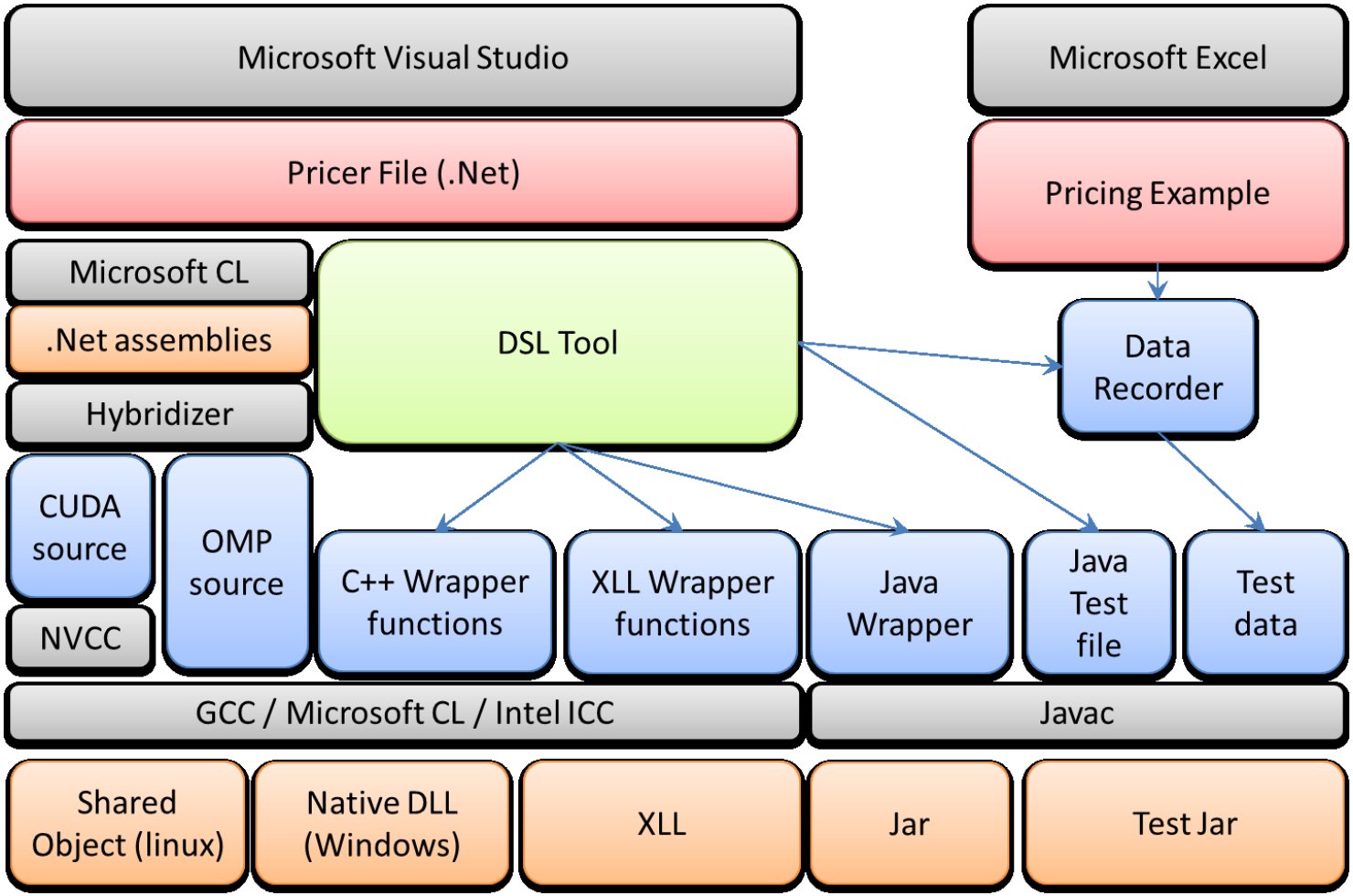
Solution - Calculation Engine

- Java
  - Negligible performance penalty compared to C++
  - Development Productivity
  - Maturity of Java for large applications
- JNA (Java Native Access)
  - Integration of native libraries and java
  - Stubs generated by the C# module
  - No measurable footprint
- NIO (New IO)
  - Market Data is managed outside the JVM (native)
  - Manage dozens of GB outside the JVM heap

Solution - Beyond Hybrid



Solution - Pricing Ecosystem



Solution - Pricing Ecosystem

- High Level Language for pricing and simulation
  - C# used
- Rationale
  - Higher Productivity (Managed language)
  - Learning Curve (Different level of abstractions)
  - Integration (Java, Excel, Python)
- Hybridizer™ Technology
  - Product from Altimesh (Florent Duguet)
  - Generate C++/OpenMP and Cuda/C code from C#

Example: Fixed Cash Flow

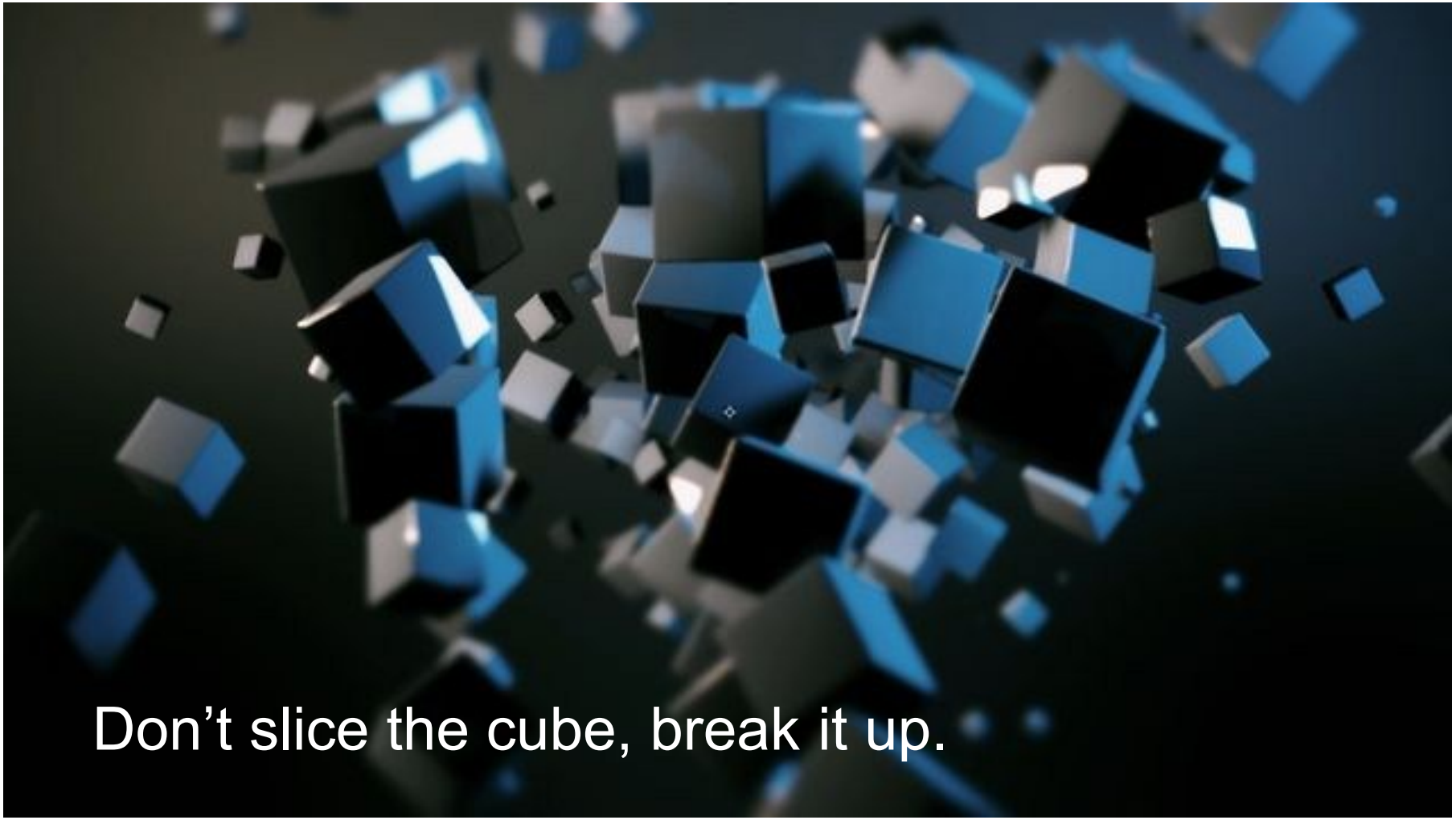
$$PV(i,j) = \sum_{k}^{nb\ of\ CF} Notional_k * YearFraction * Rate_k * D_j(t_i, T_k)$$

```
public class FixedCashFlowsPricer
{
    // Pricer Data
    public int cashFlowCount;
    public float[] paymentDate;
    public float[] value;

    // Market Data
    public YieldCurve yieldCurve;

    public void GridPV(PricerResult result, int simFrom, int simTo, int timePointFrom, int timePointTo)
    {
        int timePoint, simId;
        float price, simulDate;

        for (timePoint = timePointFrom + blockIdx.x; timePoint < timePointTo; timePoint += gridDim.x)
        {
            simulDate = result.GetTimePointDate(timePoint);
            for (simId = simFrom + threadIdx.x ; simId < simTo ; simId += blockDim.x)
            {
                price = 0.0f;
                for (int k = 0; k < this.cashFlowCount; k++)
                {
                    if (simulDate >= paymentDate[k]) continue;
                    price += value[k] * yieldCurve.GetDiscountFactor(simId, timePoint, paymentDate[k] - simulDate);
                }
                result[simId, timePoint] = price;
            }
        }
    }
}
```

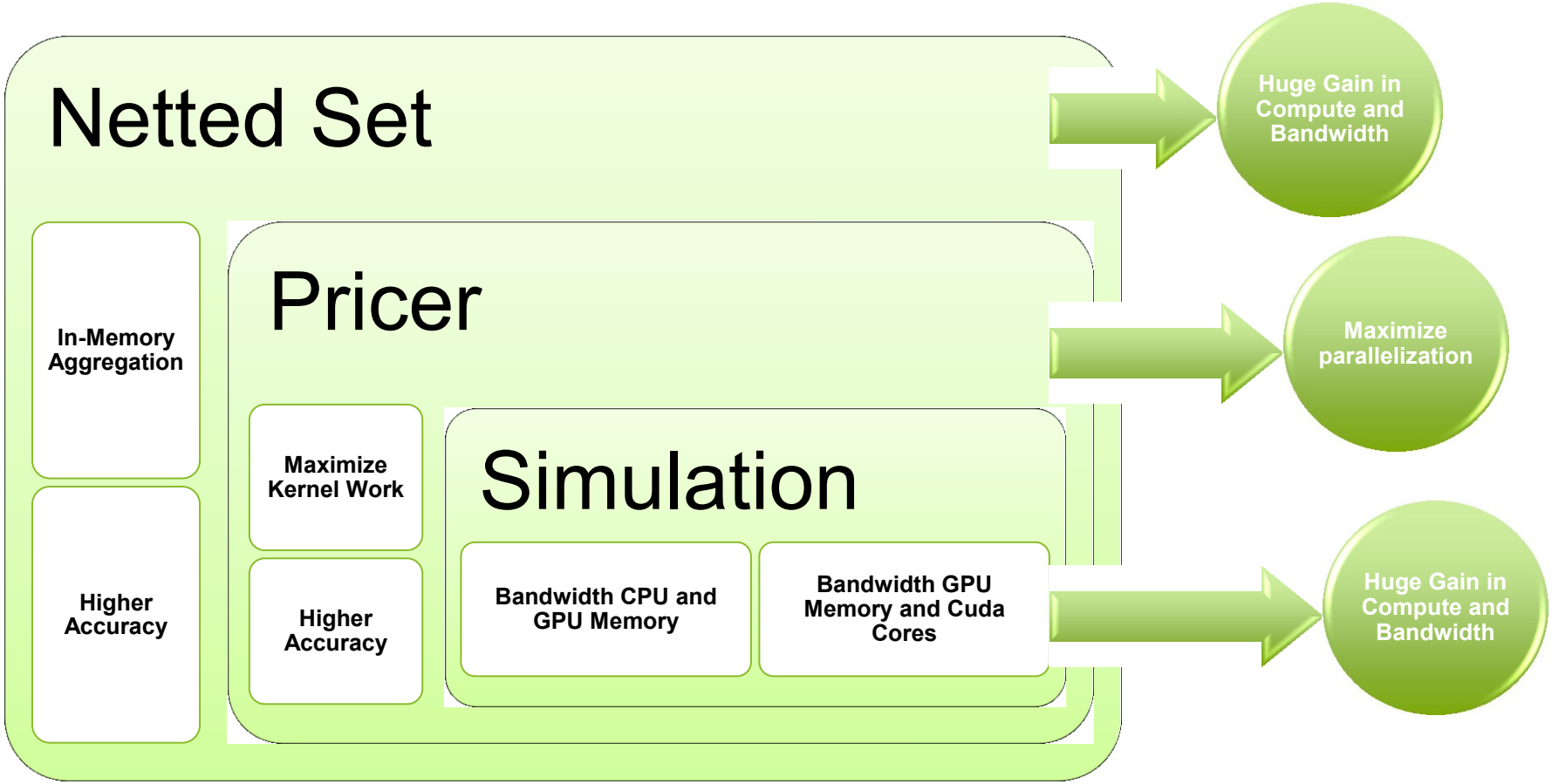
Solution - Pricing Ecosystem

- Work distribution is done using cuda concepts
  - Blocks : maps to OMP threads or cuda blocks
  - Thread : maps to FPU register unit and/or cuda-threads
- C# to Cuda
  - Use of different types of memory (shared, global, registers)
  - Access to library functions and existing code snippets
- C# to OMP
  - Basic C/C++ code generation
  - Code Vectorisation (SSE and AVX)

Back to the Cube Problem



Breaking the Cube up



Calculating Big Counterparties

- LCH
  Clearing House
  - Mainly IR Swaps
    161,025 deals, 5.5 M. Cash Flows
  - Cube
    161,025 x 1,000 x 400 = 64,4 Billion calculations

Pricer Table 28m ago

	pricer	nbCalls	nbCurrency	nbOfInstruments	delta
1	CASH	2	2	203	0.01
2	EONIA_FLOATING	1	1	1485	0.05
3	FIXED_LEG	17	17	2056032	5.92
4	OVERNIGHT_FLOATING	2	2	1675	0.07
5	STANDARD_FLOATING	17	17	3484961	42.85

- Calculation
  - 161,025 deals
  - 1,000 simulations
  - 400 time points
  - 1 M2090
- Performance
  - Overall (262s)
  - T0 Pricing (150s)
    1,075 deals/s
  - MC Pricing (49s)
    3,300 deals/s
    113,000 CF/s

Moving to Kepler

- Current Hardware
  - Dual NVIDIA M2090
  - Running large calculations
  - Production like environment
- Continuous Approach
  - Can constantly test and verify
  - Not afraid of changing code/trying things out
  - High level of confidence in our approach

➢ We have been able to try out Kepler

BUT our initial tests did not match expected performance

What was the limiting factor?

How bound are you?

Problems are split into two categories:

- Compute Bound
  - Limited by the compute power
- Memory Bound
  - Limited by the cuda cores to gpu memory bandwidth

How bound are you?

Problems are split into two categories:

- Compute Bound
  - Limited by the compute power
- Memory Bound
  - Limited by the cuda cores to gpu memory bandwidth

And a more subtle one…

- Latency Bound
  - Limited by the latency between cuda cores and GPU memory

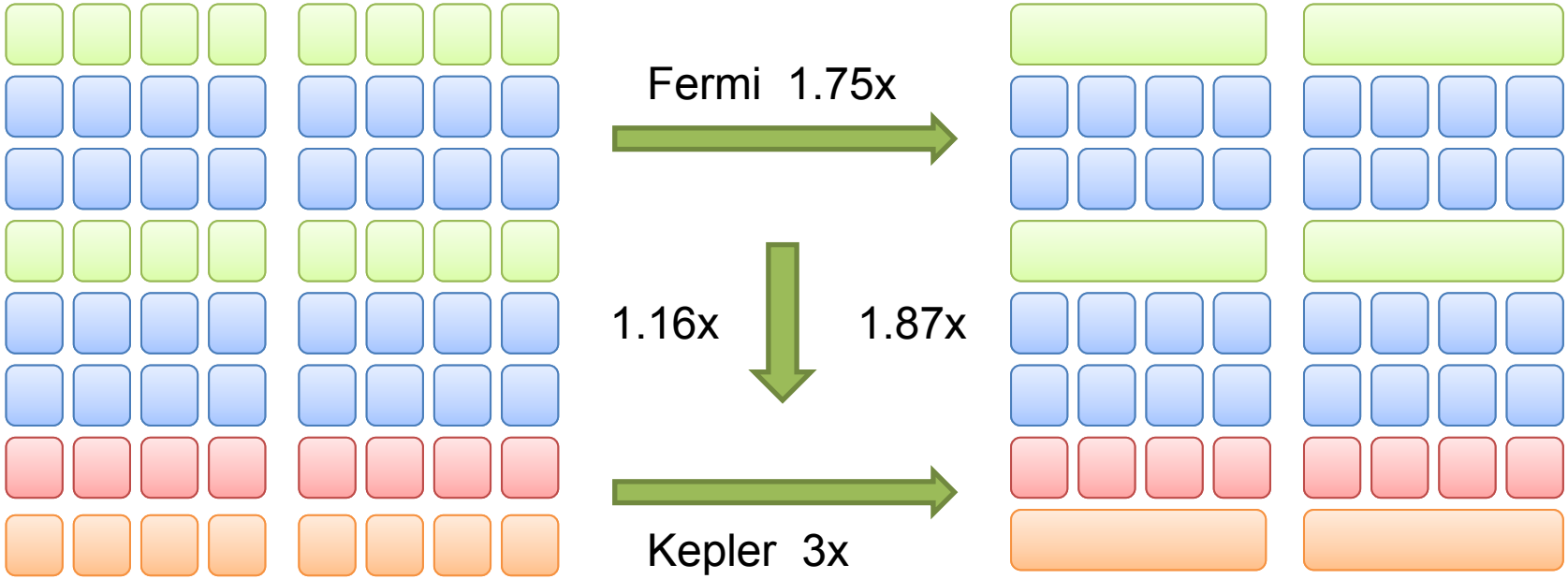
Nvidia Kepler

- Collaboration with Nvidia
  - John Ashley and Thomas Bradley (and his team)
  - Nvidia identified our problem was **latency bound**
- LDG
  - Simple use of Texture Memory (from memory address)
  - Keyword __ldg
  - Impact on pricing code

```
[IntrinsicFunction("hybrid_ldg")]
public static float ldg(float val) { return val; }
```

- Vectorization on GPU
  - Use of float4 instead of float
  - Allow to pack memory reads into one request (128 bits)
  - Impact on pricing code (much heavier, on going work)

LDG, float4 on Fermi and Kepler



16 LOADS – 8 STORES

4 LOADS – 2 STORES

Calculation Results

$$PV(i,j) = \sum_{k}^{nb\ of\ CF} Notional_k * YearFraction * Rate_k * D_j(t_i, T_k)$$

- Fixed Cash Flow Pricing
  - 500 cash flows
  - 1,000 path

	time (ms)
AVX (dual E5-2640)	40.3
CUDA (M2090)	7.5
CUDA (K20)	4.0

X5.3

Is the boat bigger?

- Nightly runs
  - 12x more runs than current system
  - Cost is 5x less than current setup

  ⇒60x more calculations for the same cost

- And the boat can get even bigger…
  - Less sensitive to the number of deals
  - Scale linearly with number of instruments
  - Reuse of data for different instruments

Summary

- Counterparty Risk and Stress Scenario is A Big Problem
- Had to rethink the solution
- Use a hybrid technology stack
- The Cube is broken, not sliced
- Continuous approach

Summary

- Kepler can allow us to run 2x as many scenarios as Fermi
- Kepler investigation has helped us gain new insights
- Can now do massive parallel computations
- Well positioned to leverage future improvements in processor technology

The End

- Thank You
- QA

No.17 European group

all sectors[1]

190,000 employees

representing more than 150 nationalities

Present in nearly 80 countries

€94.4bn

of consolidated equity as at 31/12/2012

23 million clients

and more than 7,000 branches

across its retail network

A leading global bank

(1) Based on the Group's FY11 net income, a European ranking in the Stoxx 600 index universe

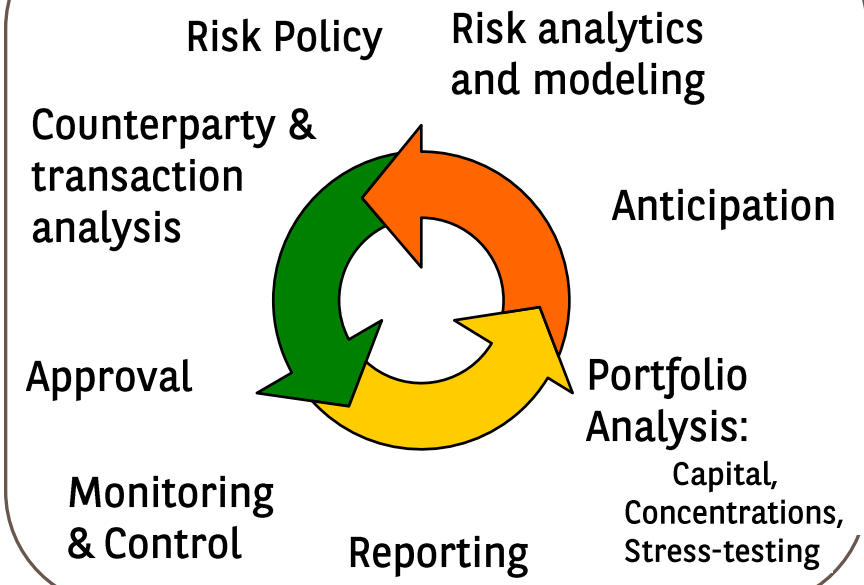
Group Risk Management

Risk Coverage

- **Credit risk**
  - Corporate
  - Retail
- **Market & Counterparty risk**
- **Liquidity risk**
- **Insurance risk**

Operational Risk and Reputation risk are placed under the responsibility of Conformity

Risk Cycle



Main Missions	
	■ To advise Bank Management on risk policy definition
	■ To contribute as a “second pair of eyes” that risks taken on by the Bank align with its policies
	■ To report and alert Bank Management of the status of risks to which the Bank is exposed