Designing a GPU-Based Counterparty Credit Risk System

Patrik Tennberg, TriOptima

March 19th 2015

triCalculate

TriOptima

NEW YORK LONDON STOCKHOLM SINGAPORE TOKYO

Agenda

- A new methodology for counterparty credit risk calculations
- System Overview
- Architecture
- ComputeEngine - CUDA made easy
- ValuationEngine
- SimulationEngine

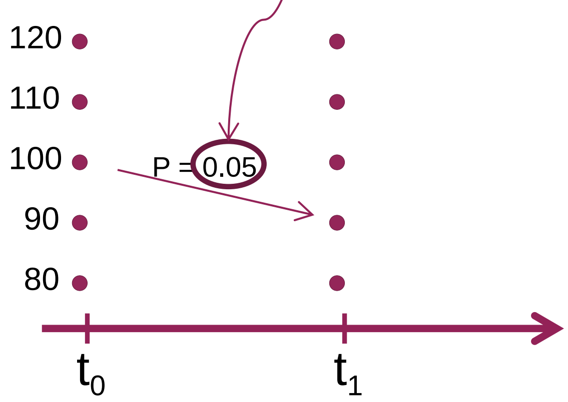
TriOptima

A new methodology for counterparty credit risk calculations

TriOptima

Valuation

	120	110	100	90	80
120	0.9	0.07	0.03	0	0
110	0.03	0.9	0.05	0.02	0
100	0.01	0.05	0.88	0.05	0.01
90	0	0.02	0.05	0.9	0.03
80	0	0	0.03	0.07	0.9



- Discretized time and space
- Market factor dynamics described through transition probability matrices
- Matrices can be used for both:
  - Backward induction to price derivatives
  - Stepping forward in Monte-Carlo simulation for counterparty credit risk
- Calculation builds on matrix algebra
  - Very fast implementation using modern GPU technology

TriOptima

Valuation, cont.

- Start from a general model for the underlying

$$dS_t = \mu_{a_t} dt + \kappa(t)\left(\theta(t) - S_t\right)dt + \sigma(t)S_t^{\beta(t)} dW_t + \alpha(t)S_t\left(dN_t - \lambda(t)dt\right)$$

- Use probability theory to generate the transition probability matrix at a (very) short time period

$\longleftarrow S_{dt} \longrightarrow$

	120	**110**	**100**	**90**	**80**
120	0.99	0.01	0	0	0
110	0.01	0.98	0.01	0	0
100	0	0.01	0.98	0.01	0
90	0	0.02	0.01	0.98	0.01
80	0	0	0	0.01	0.99

- Multiply the transition matrix by itself to generate longer period matrices

Advantages

- Consistency in market dynamics
  - Traditional approaches using one dynamic for MC generation and another dynamic for pricing (implied by standard pricing models)
- Realistic models for market dynamics
  - Numerical approach means that you are not confined to models with analytical solutions
  - Caters for wrong-way risk
- Simple implementation of new products
  - Only the pay-off profile need to be described
- Very fast calculations when designed for new hardware
  - All prices for all paths is pre-calculated during the valuation step
  - Enables many more MC simulations (100,000 scenarios) which also increase accuracy

The method is developed by Claudio Albanese

www.albanese.co.uk

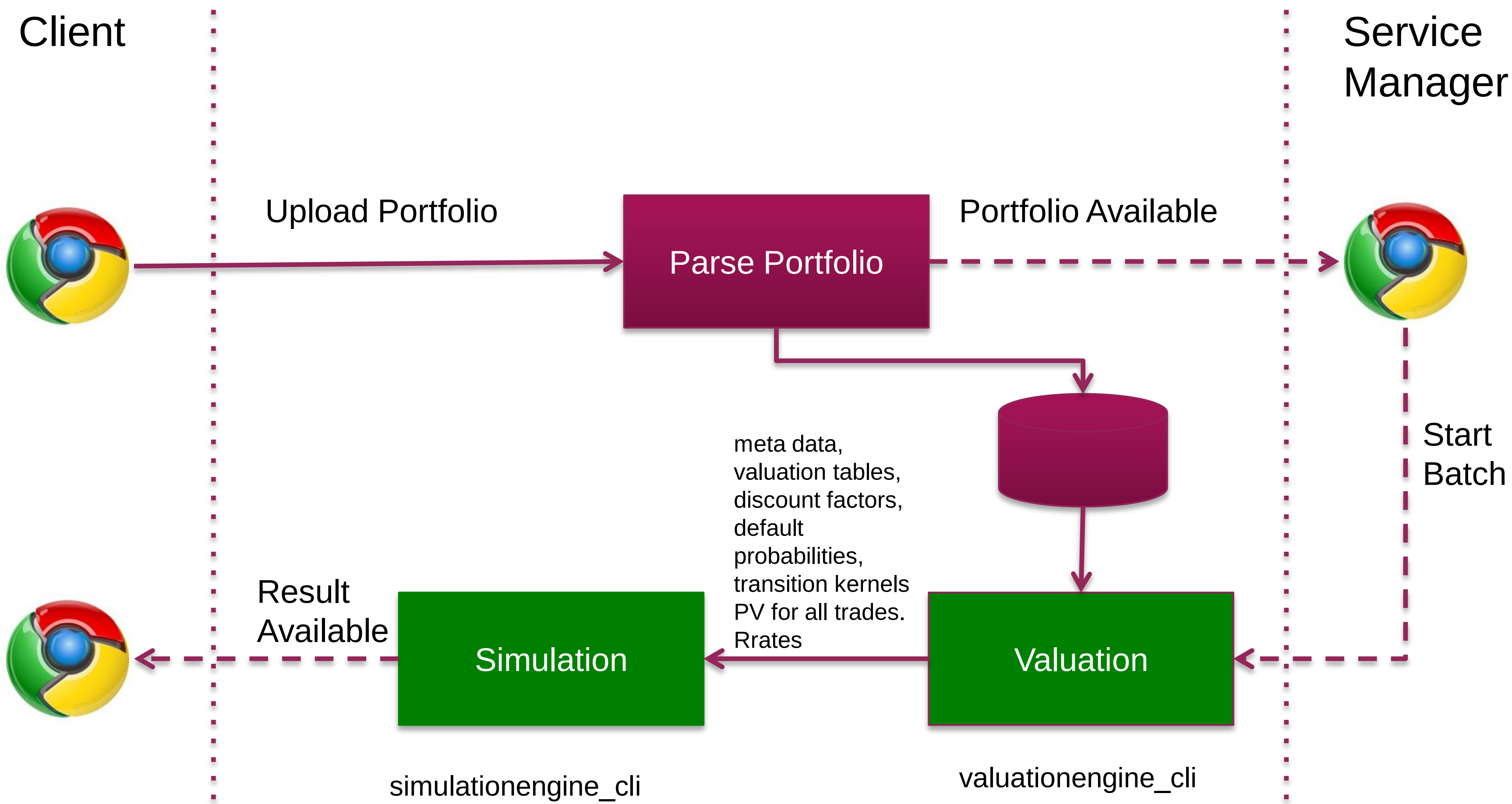
More information regarding the method can be found here:

Coherent global market simulation and securitization measures for counterparty credit risk

TriOptima

System Overview

triCalculate Overview



TriOptima

Coverage and Features

Coverage:

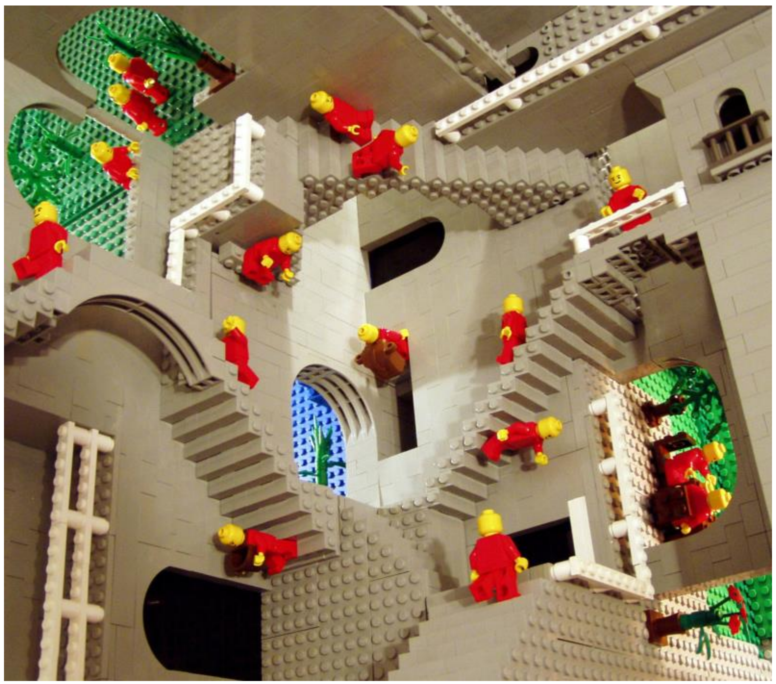
- Rates
  - Swap
  - Forward Rate Agreement
  - Swaption with cash delivery
  - Swaption with physical delivery
  - Cap / Floor
- Inflation
  - Swap / Zero Coupon Swap
- FX
  - Forward
  - Swap
  - European Option
  - Barrier Option
- More is on the way

Features:

- Collateral / GAP Risk
- Sensitivities (IR, FX and Credit delta)
- Support for Path Dependency

TriOptima

Architecture



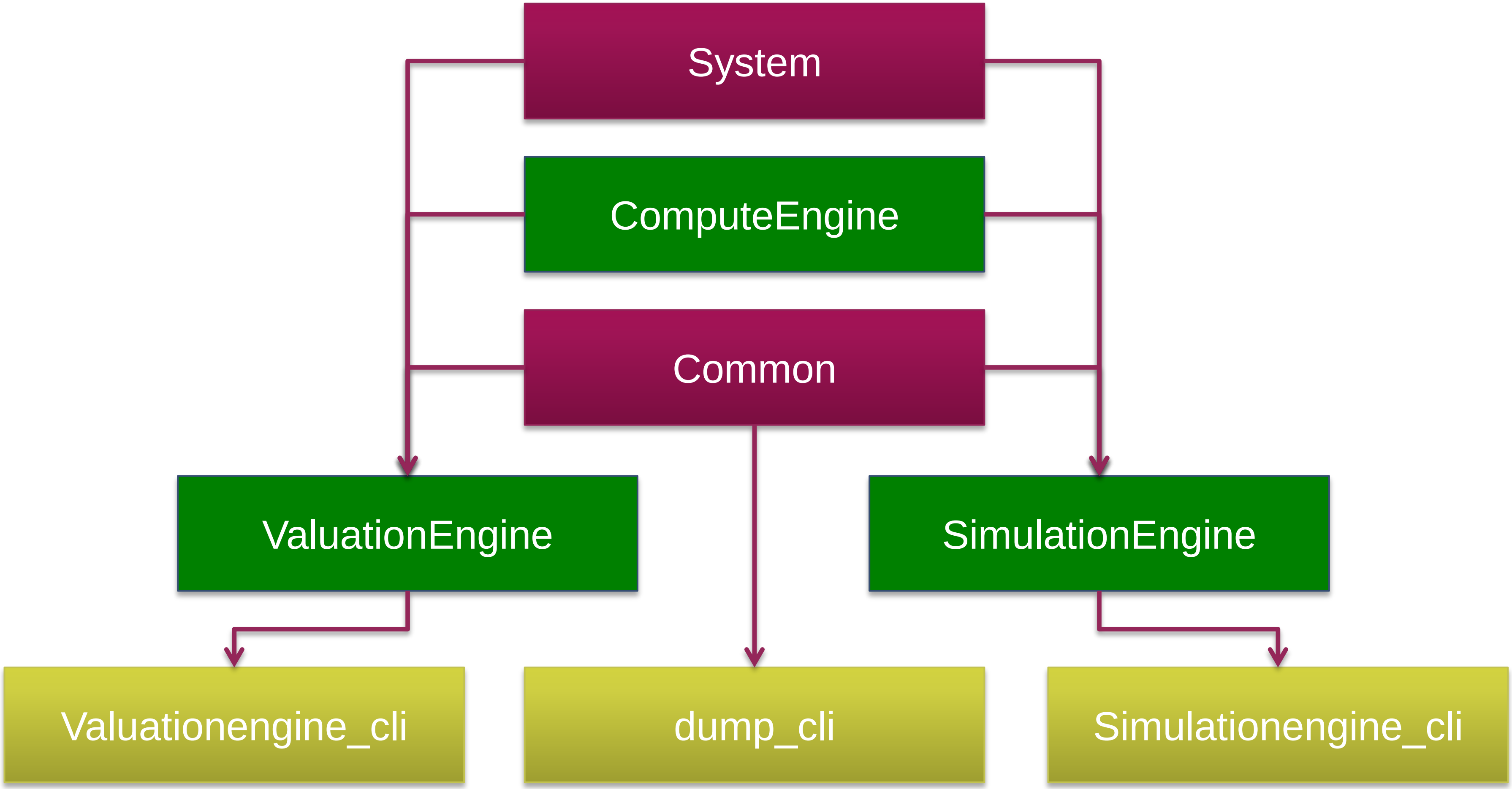
Copyright © A. Lipson 2003

Architectural Goals

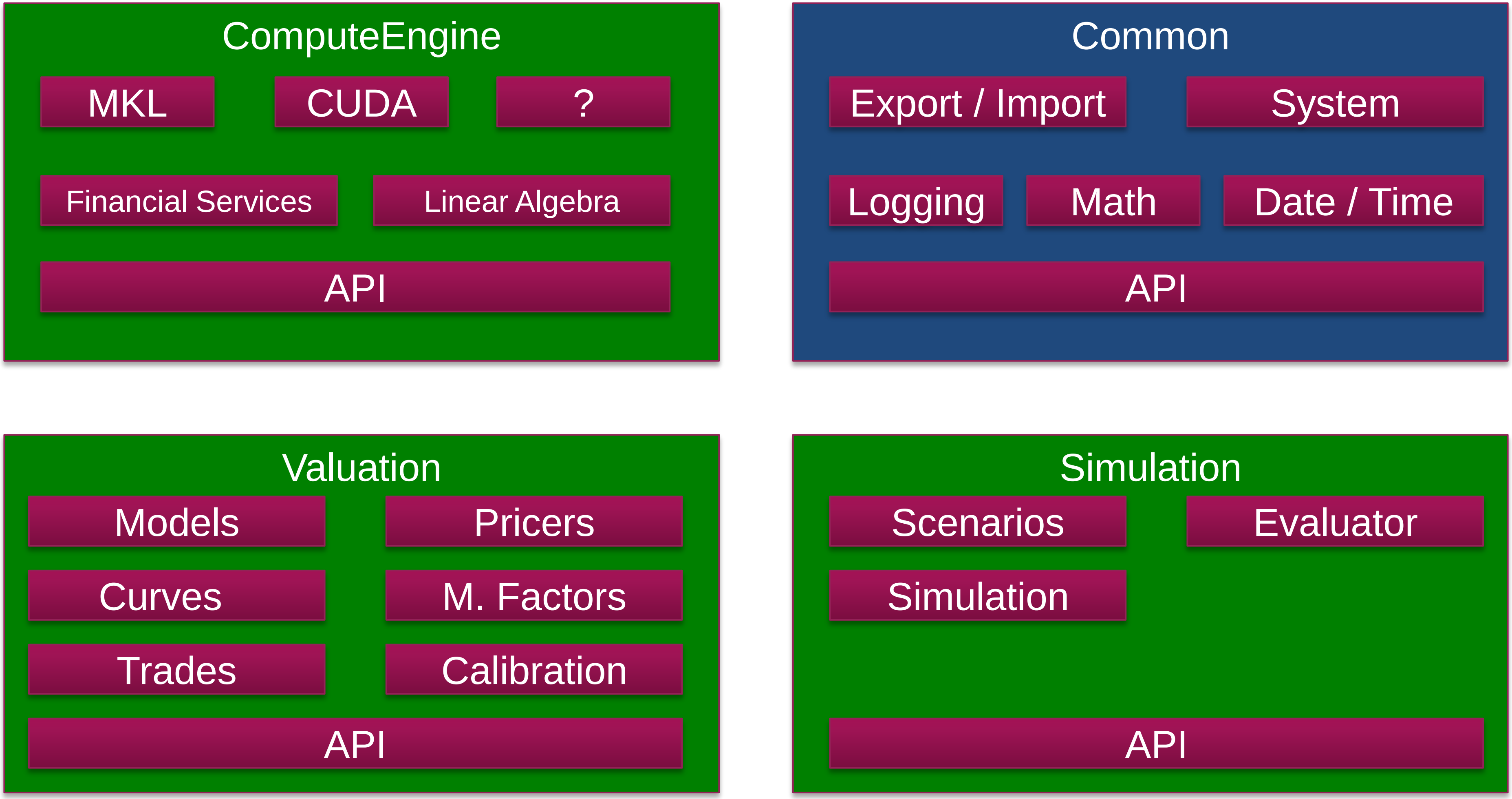
- Device Agnostic
  - CPU (MKL) and CUDA
- Portable
  - Linux (Prod), OSX and Windows (Development)
- Extendible
- Simple and natural programming model
  - Application code has no knowledge about devices, threads and other complicated stuff
- Testable
  - 800+ tests, executed at every code commit
- Fast Enough!

TriOptima

Overview


System
ComputeEngine
Common
ValuationEngine
SimulationEngine
Valuationengine_cli
dump_cli
Simulationengine_cli

High Level Architecture



TriOptima

The ComputeEngine

NVIDIA CUDA

made easy

TriOptima

ComputeEngine – Low Level API

- Device Management
  - *ceGetDeviceCount, ceEnumDevices, ceCreate*DC
- Memory management
  - *DataHandle, ceAllocateData, ceFreeData*
  - *Supported Types: Arrays, Vector, Matrix, Float, Double, Integer*
  - Devices has their own memory manager
- Operations
  - *Linear Algebra:* e.g. FastExp, Floor, Multiplication (MS, MV, MM)
  - Financial Operations: e.g. ceAddCashFlows, ceGetDailyDiscountFactors
- Asynchronous execution
  - *ceAddJobToQueue*

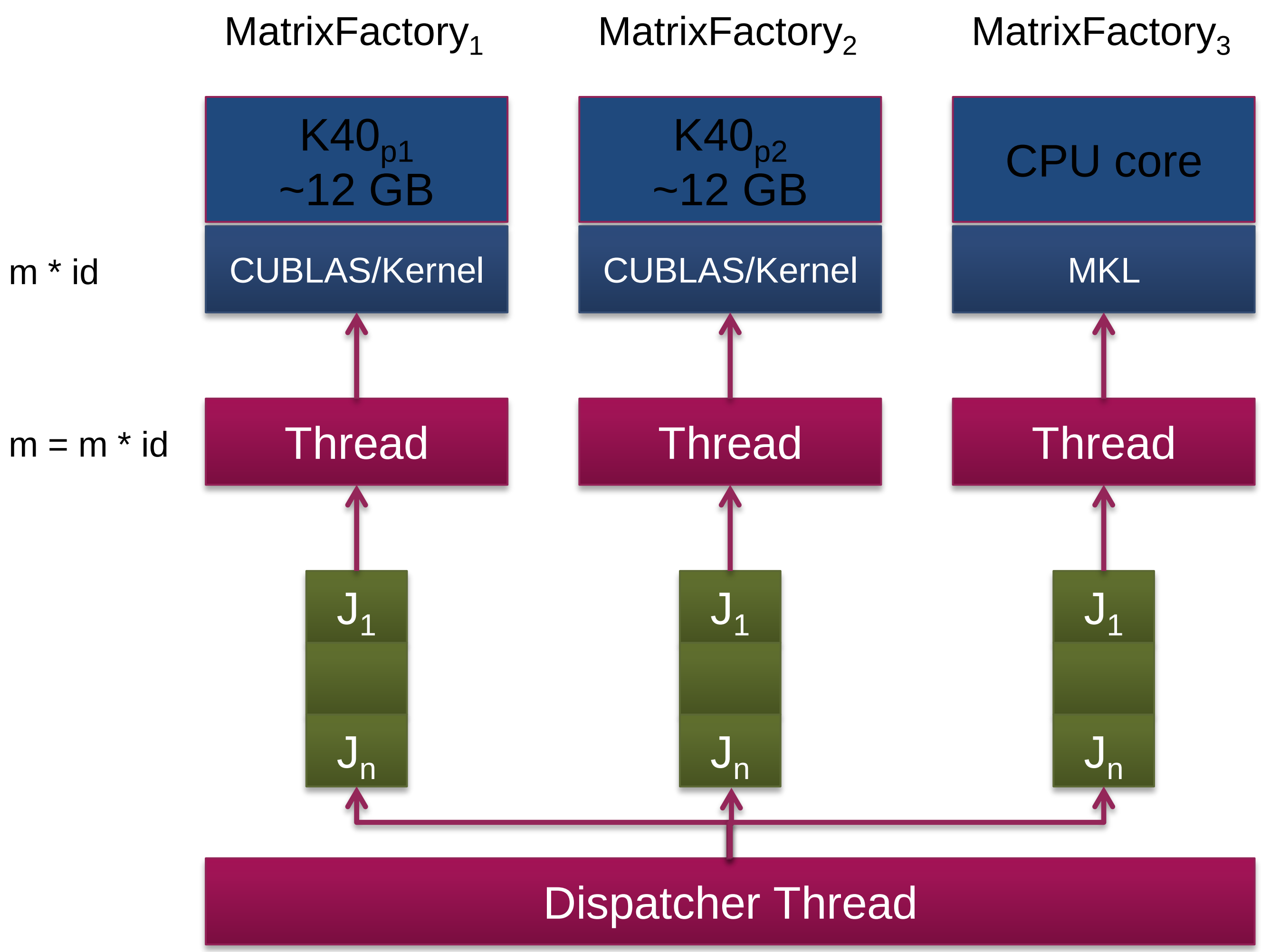
TriOptima

ComputeEngine – High Level API

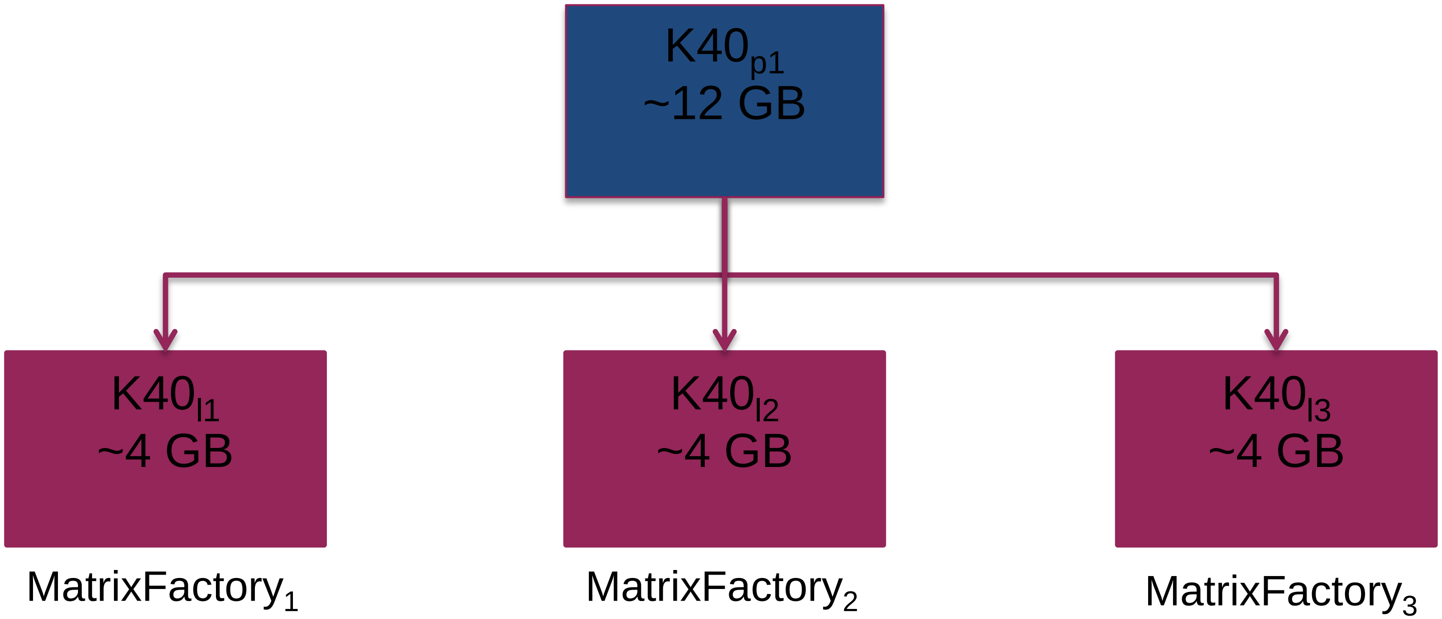
```
typedef Matrix<float> FloatMatrix;

MatrixFactory mf(DeviceType::CUDA); // DeviceType::MKL

FloatMatrix m = mf.CreateMatrix(3, 3, 1.0f);

m *= 0.005f;

FloatMatrix id = mf.CreateIdentityMatrix(3);

m = m + id;

m.FastExp(3);
```

- A matrix factory represents a device
- A matrix factory knows how to create data types (e.g. vectors, matrices, etc.) on a specific device.
- All operations on data types are executed on a specific device without memory transitions

Devices / MatrixFactory / Execution



Logical vs. Physical Devices



Logical Devices gives good occupancy even when you saturate the GPU with relatively small data sets

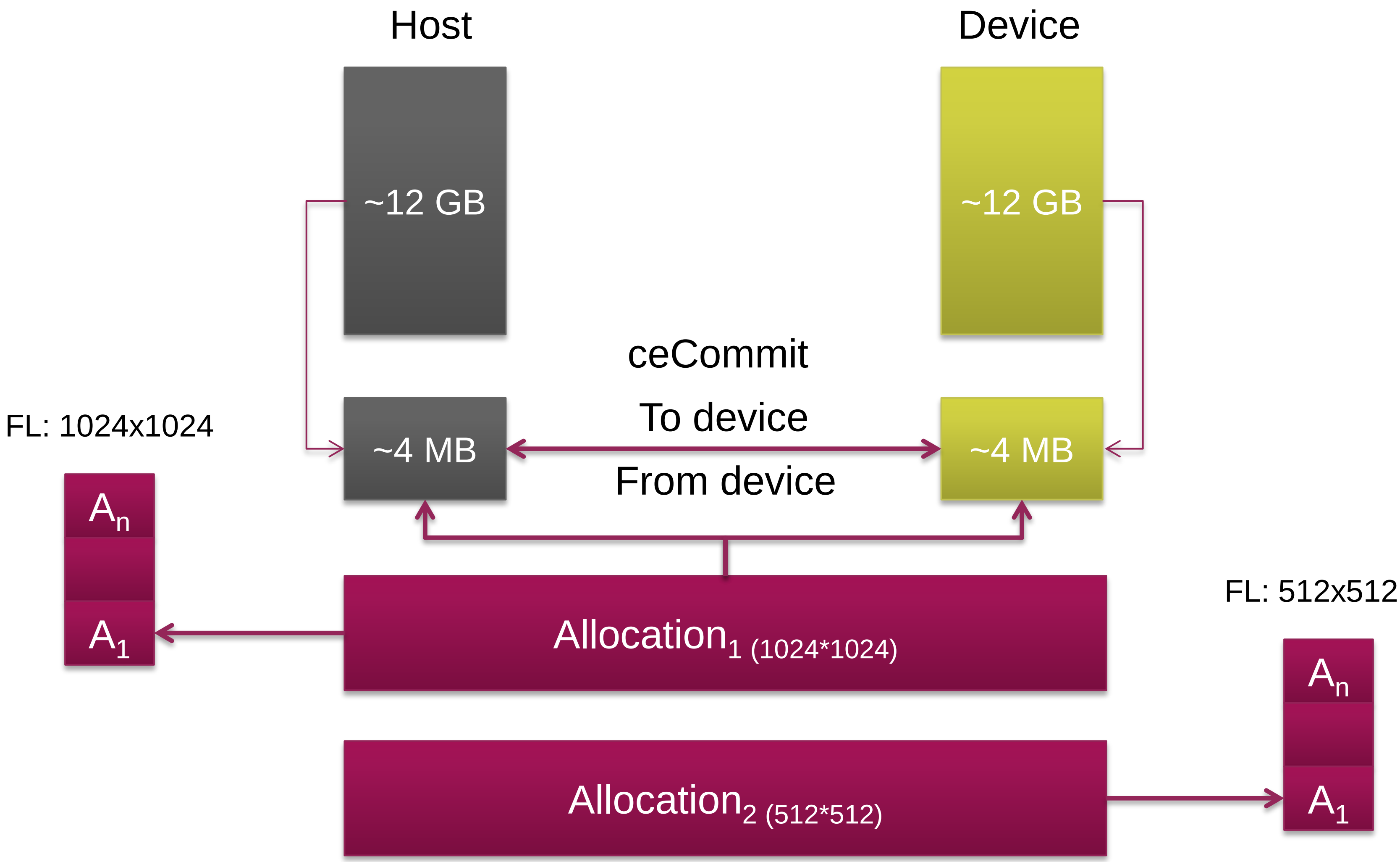
Memory Management

- Fixed size dynamic Memory allocation
  - Memory allocated from a heap – the heap can only grow
  - Memory is never returned to the heap
    - Free-list based on allocation size
  - The heap is released for each job
- Host and GPU memory pools
  - Semi automatic synchronization

Uniform Memory will simplify memory management but is still too slow to be used for performance critical applications

TriOptima

Memory Management



TriOptima

Memory Management

- Statistics and Memory Debugger Support
- cudaMalloc vs. CE Heap Manager
  - cudaMalloc: 258 seconds
  - CE: 47 seconds

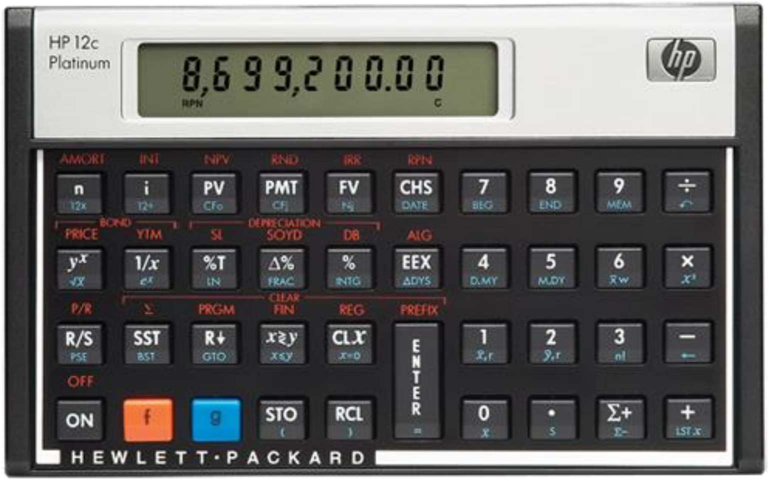
It is important to be able to seamlessly switch between **CUDAs heap manager** and your **own heap manager** since the later can hide memory related bugs and prevent you from using cuda-memcheck

Pros and Cons of the ComputeEngine

- Pros
  - Device Agnostic
  - Relatively Easy and Intuitive to use – mathematical notation
  - Sandbox development

- Cons
  - We are not using the full potential of the GPUs
    - Usage of logical GPUs mitigates part of this problem

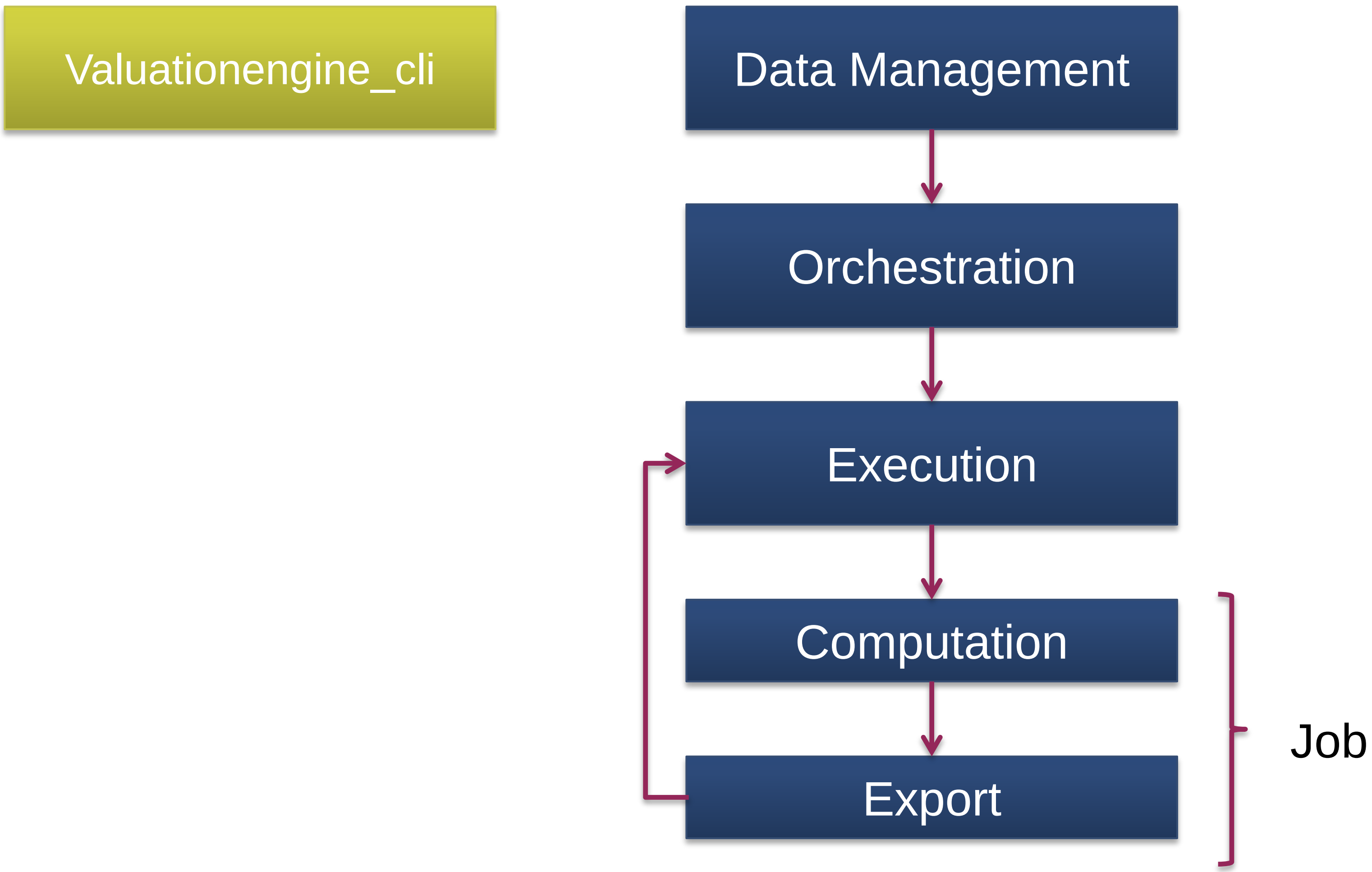
ValuationEngine



Modules - ValuationEngine

- Models:
  - Stochastic Drift (IR), Local Correlation (FX), Distance To Default (Credit)
- Pricers:
  - FX Forward, FX Option, Inflation Swap, etc.
- Pricer Data (Curves)
  - Credit, FX, IR
- Calibrations
- Trades / Silos
- Cache

Valuation Process



Orchestration

- Input: Calibration, Configuration, Portfolio, Curves
- Portfolio -> Silos:
  - One Silo per Market Factor (e.g. IR_EUR) and Pricer (e.g. IR_SWAP)
  - Silos can have the same market factor (model) but different pricers
- Job Generation
  - Valuation Tables, PV, Discount Factors, Transition Matrices, Default Probabilities (Computations)
  - Meta Data (e.g. FX rates, sensitivity specification, etc.)
    - Used by the simulation phase
  - A job is a Computation and associated cache and Export of the computed data (e.g. write it to disk)

Orchestration

- Caching is done on calibration level (e.g. market factor)
  - Valuation table
  - Present Values
  - Discount Factors
  - Transition Probability Matrices
  - Default Probabilities

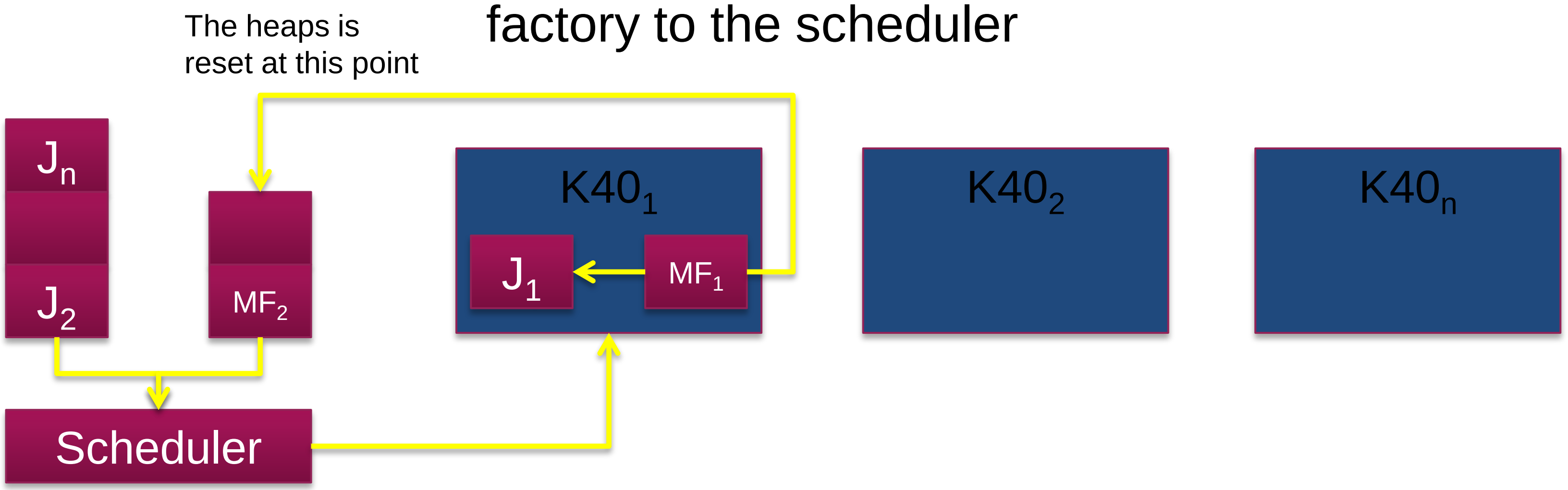
Computation / Export

- Computations
  - Valuation tables
  - Transition Probability Kernels
  - Discount factors
  - Present Values, FX Rates, etc.
- Export
  - When the computation is done the result is exported down to disk
  - Meta data describes for the simulation engine how data is to be interpreted and is part of the export

17GB of data for a midsize portfolio (~6000 trades)

Execution - Parallel Execution Model

- Calculations are partitioned into jobs
- When a job is scheduled for execution that job is assigned a matrix factory (a logical device)
- Jobs are scheduled over all available matrix factories
- As soon as a job is done it returns its matrix factory to the scheduler



Execution - Parallel Execution Model

- All memory shared between threads are read-only
- All communication between threads are done through queues

It is important to be able to seamlessly switch between **single threaded** execution and **multi-threaded** execution

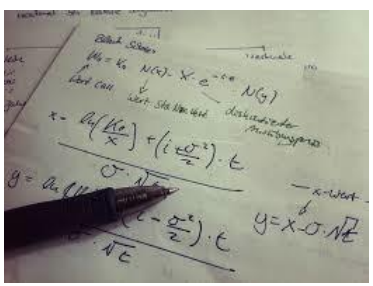
TriOptima

Sensitivities

- IR delta, FX delta, Credit Delta
- Bump and Revalue (numerical differentiation)
  - Calibration is shifted (e.g. yield curve for IR_DELTA)
  - New data sets:
    - Valuation Table, Discount Factors (IR delta, one set for every shift)
    - Valuation Table, FX Rates (FX delta)
    - Credit Probabilities (Credit delta)

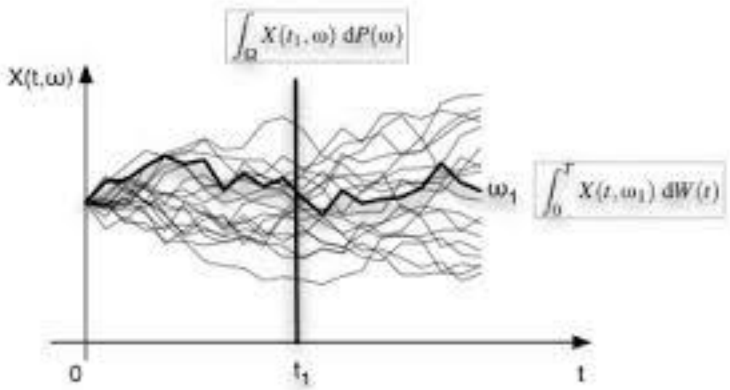
Performance

- Portfolio
  - 5900 trades, IR Swaps, cap-floor, swaptions and FX forwards in several currencies
  - 505 counterparties
  - 81 time steps



- Valuation (6 logical K40, 2 Physical)
  - Generated data ~17 GB
  - Took 46,5 seconds (81.5 s on 2 physical)
- Valuation with 35 sensitivities
  - Generated Data ~30 GB
  - Took 70 seconds

Simulation


$\int_\Omega X(t_1,\omega)\,dP(\omega)$
$X(t,\omega)$
ω_1
$\int_0^T X(t,\omega_1)\,dW(t)$
0
t_1
t

Simulation Process

- Read all Input Data, Collateral Agreements and Path Dependency Data (if applicable)
- Generate Scenarios
- Distribute scenarios over all available CPUs
  - Calculate CVA, DVA, FVA
  - Path Dependency requires lookup of path dependent payoffs
  - Collateral requires lookup of collateral agreement
  - Sensitivities requires additional full simulations
- Collect the result from all CPUs
- Export the result (Json)

Performance

- Portfolio
  - 5900 trades, IR Swaps, cap-floor, swaptions and FX forwards in several currencies
  - 505 counterparties
  - 81 time steps

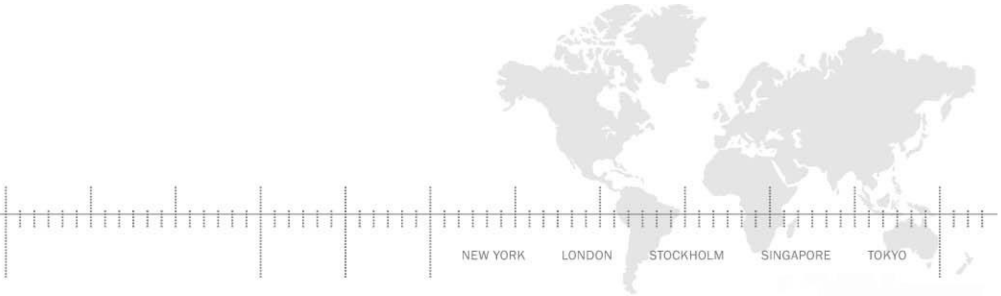
- Simulation (2 CPU, Intel 2699 v3, 18 cores)
  - 100,000 scenarios
  - Took 55 seconds
  - Took 15 minutes with 35 sensitivities

TriOptima

Thank You

patrik.tennberg@trioptima.com

www.trioptima.com



TriOptima