The anatomy of a GPU based counterparty credit risk system

Patrik Tennberg, TriOptima

July 10th 2014

Agenda

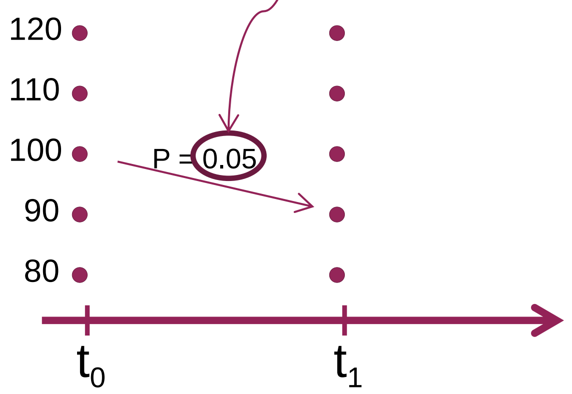
- A new methodology for counterparty credit risk calculations
- System Overview
- Architecture
- CUDA made easy

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:
  - Stepping forward in Monte-Carlo simulation for counterparty credit risk
  - Backward induction to price derivatives
- 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

S_{dt}

	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 which also increase accuracy

The method is developed by Claudio Albanese

www.albanese.co.uk

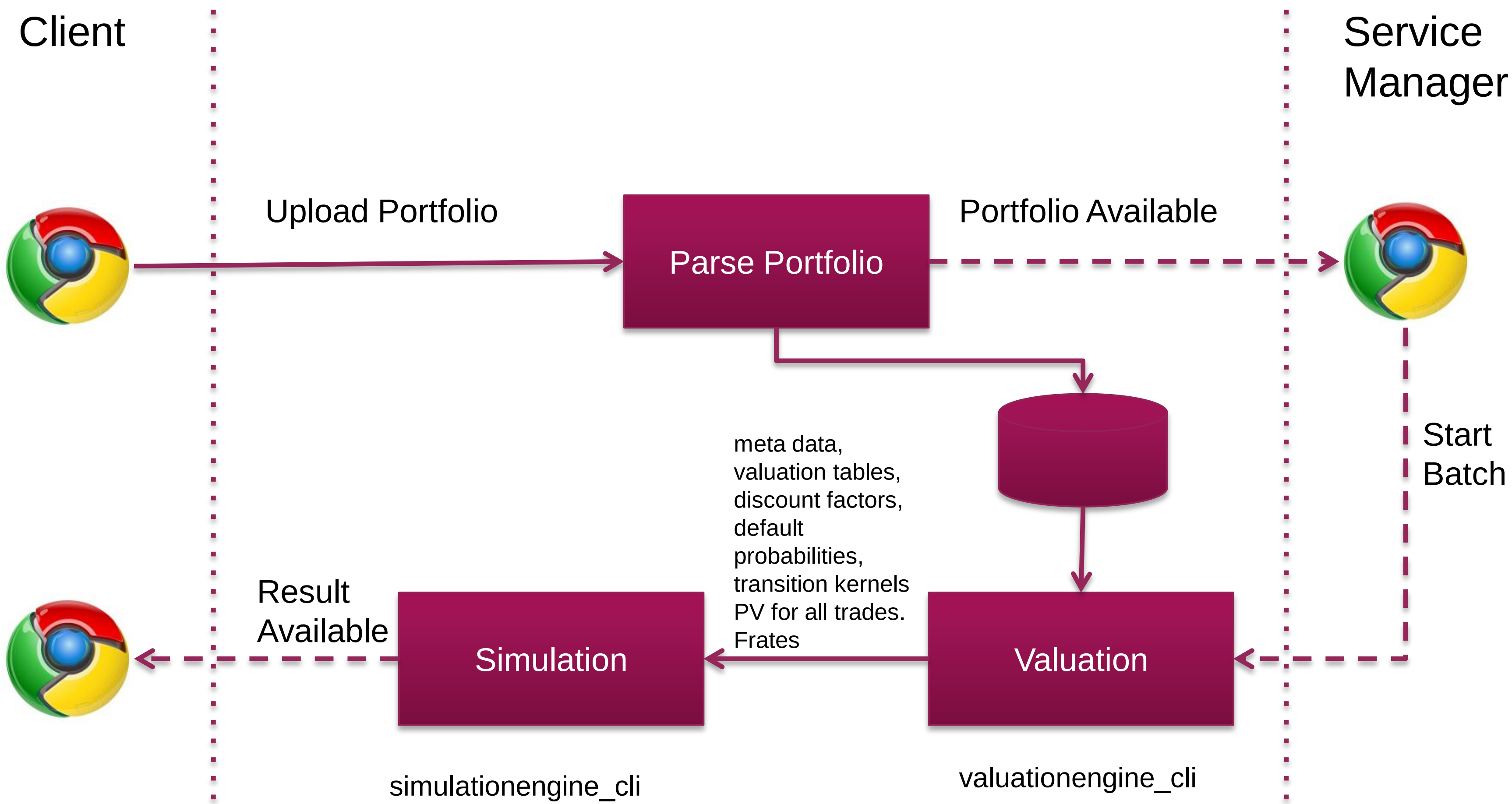
More information regarding the method can be found here:

Coherent global market simulation and securitization measures for counterparty credit risk

TriOptima

System Overview

triCalculate Overview



TriOptima

Architecture

Architectural Goals

- Device Agnostic
  - MKL and CUDA
- Portable
  - Linux (Prod), OSX and Windows (Development)
- Simple and natural programming model
  - Universal language of mathematics
  - Application code has no knowledge about devices, threads and other complicated stuff
- Testable
  - 700+ tests, executed at every code commit
- Fast Enough!
  - Simplicity over performance as long as it fast enough

TriOptima

High Level Architecture

ComputeEngine

- MKL
- CUDA
- ?
- Financial Services
- Linear Algebra
- API

Common

- Export / Import
- System
- Logging
- Math
- Date / Time
- API

Valuation

- Models
- Pricers
- Curves
- M. Factors
- Trades
- Calibration
- API

Simulation

- Scenarios
- Evaluator
- Simulation
- API

TriOptima

NVIDIA CUDA

made easy

TriOptima

ComputeEngine – Low Level API

- Device Management
  - *ceGetDeviceCount, ceEnumDevices, ceCreate*DC
- Memory management
  - *DataHandle, ceAllocateData, ceFreeData*
  - *Supported Types: 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

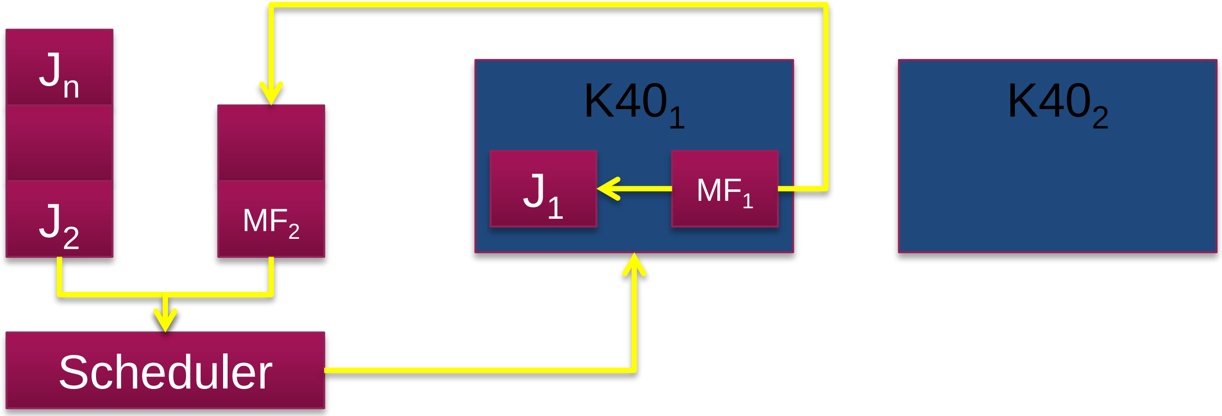
```
typedef Matrix<float> FloatMatrix;

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 (e.g. CUDA or MKL (CPU)).
- 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

TriOptima

Parallel Execution Model

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



TriOptima

Performance

- Portfolio
  - 5727 trades, IR Swaps, cap-floor, swaptions and FX forwards in several currencies
  - 505 counterparties
  - 81 time steps
- Valuation (2 K40)
  - Generated data 15 GB
  - Took 84 seconds
- Simulation (2 CPU, 12 cores each)
  - 100,000 scenarios
  - Took ~2 minutes 24 seconds

TriOptima

Pros and Cons

- Pros
  - Device Agnostic
  - Easy and Intuitive to use – mathematical notation
  - Sandbox development
- Cons
  - Performance is not optimal

Thank You

patrik.tennberg@trioptima.com
www.trioptima.com