GPU-Accelerated Pricing of Credit Value Adjustment for an Interest Rate Swap under Gaussian Heath Jarrow Morton (HJM) model

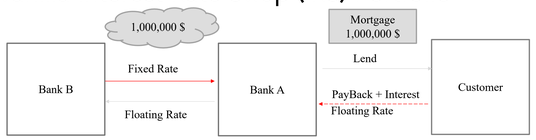
Michel Herrera Sanchez, michelherrerasanchez@gmail.com

Background

Portfolio Credit Value Adjustment (CVA) calculation is an intractable problem in Investment Banking

1,000,000 of trades
5000 pricing points (time)
400 scenarios
⟶ 2000, 000, 000,000 Simulations

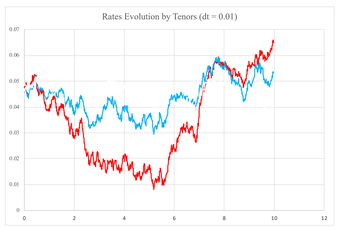
Vanilla Interest Rate Swap (IRS) Portfolio

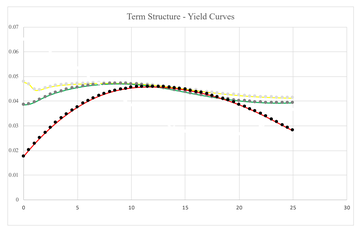


What is the value of the portfolio if BankB or Customer default ?

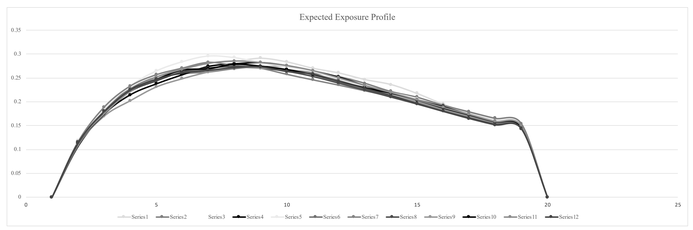
Numerical Simulations, Exposure and CVA

A set of stochastic differential equations are simulated with Monte Carlo (MC) methods to predict the futures values of the floating rates (HJM) to price the IRS portfolio





100_000 IRS portfolio exposure profiles (scenarios)



Credit Value Adjustment Calculation (CVA)

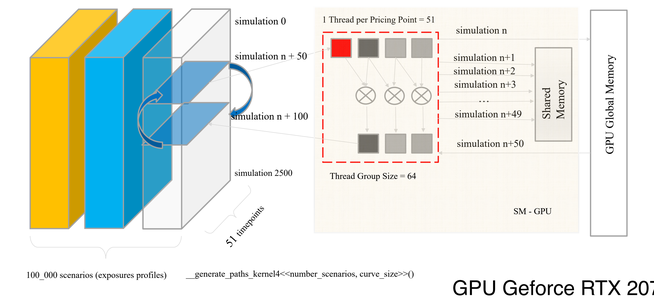
$$\mathrm{CVA} = (1-R)\sum_{k=0}^{n} DF\ (t_k, t)\ \mathrm{EE}(t_k)\ PD(t_k, t_k - 1)$$

Where:

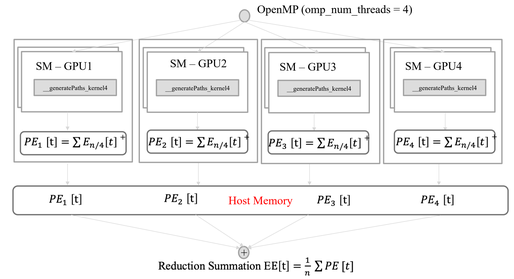
- R is the recovery rate
- DF (t, t) is the discount factor, discounting the value from time of default back to the valuation time.
- EE (t) is the Expected Exposure at time t
- PD (t_k , t_{k-1}) is the probability of default of the counterparty at time τ, $\tau \leq T$

GPU Acceleration

Loop Parallelization, Tiling, cuRAND and Stencil Parallel Patterns



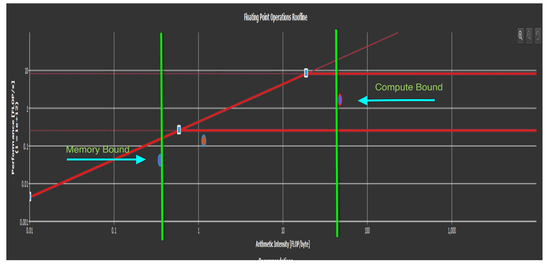
Multi-GPU Scenario Simulation



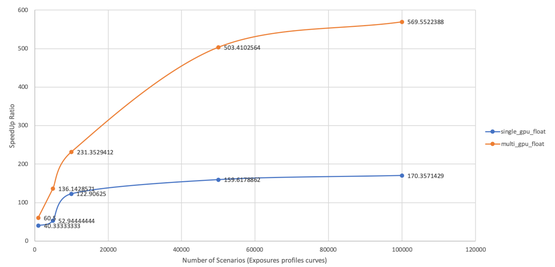
GPU Geforce RTX 2070 vs Intel Xeon W-2123

Results

The optimization improved the **arithmetic intensity** for kernel __generate_path_kernel4 from 1,16 flops/byte up to **47,5 flops/byte**. (1)



(1) Roof-line Analysis __generate_path_kernel



(2) Speed Up Analysis

Using single-floating point accuracy the **GPU** acceleration reached an **speed up** of **170x** for a simulation profile containing 100000 exposures curves with 51 timepoints. The Multi-GPUs acceleration increased the **speed up** close to **600x.** Comparison based on the reference serial cpu implementation. (2)

Platform	MC Simulation optimization	MC simulation execution time (ms)	MC Simulation speed up
cpu	__generate_kernel_path_cpu_double	21114	
single-gpu	__generate_kernel_path_double	1470	14.36326531
multi-gpu	__generate_kernel_path_double	509	41.48133595
single-gpu	__generate_kernel_path4_double	540	39.1
multi-gpu	__generate_kernel_path4_double	164.59	128.2823987
cpu	__generate_kernel_path_cpu_float	19335	
single-gpu	__generate_kernel_path_float	1030	18.77184466
multi-gpu	__generate_kernel_path_float	397	48.70277078
single-gpu	__generate_kernel_path4_float	111.863	172.8453555
multi-gpu	__generate_kernel_path4_float	30.5459	632.9818404

Optimization	MC Simulation kernel name	Mean-square errors	Accuracy
tiling_gpu_double	__generate_kernel_path_double	0.01377044	10^-2
tiling_multigpu_double	__generate_kernel_path_double	0.01397286	10^-2
tiling_shared_gpu_double	__generate_kernel_path4_double	0	10^-6
tiling_shared_multigpu_double	__generate_kernel_path4_double	0.00059256	10^-4
tiling_gpu_float	__generate_kernel_path_float	0.01380047	10^-2
tiling_multigpu_float	__generate_kernel_path_float	0.0143285	10^-2
tiling_shared_gpu_float	__generate_kernel_path4_float	0.000006	10^-6
tiling_shared_multigpu_float	__generate_kernel_path4_float	0.003549	10^-3

The execution on multi-gpu solution was faster and scales linearly but reduced floating point accuracy due to floating point stability of the algorithm.

This work won the best hackathon awards at the PUMPS 2021 Hackhathon organised as part of the Programming and Tuning Massively Parallel Systems + Artificial Intelligence summer school (PUMPS+AI) taught by prof. Wen-mei Hwu at the Barcelona Supercomputing Center (BSC) and Universitat Politècnica de Catalunya (UPC). For more information and detailed results please refer to the presentation document https://github.com/mahsanchez/cva_hjm_cuda/blob/master/doc and the open source implementation https://github.com/mahsanchez/cva_hjm_cuda

Conclusions

The MC simulations methods scale linearly as the number of scenarios increased with multiple-GPU

A single GPU-Acceleration achieved better accuracy results.

CVA calculations become a tractable problem with GPUs.

Further Work

Use multiple CUDA Streams to improve GPU utilization.

Improve the floating-point accuracy for the multi-gpu implementation.

Implement a sensitivity analysis using using AAD.

References

Ruiz, I (2015) XVA Desks A New Era for Risk Management Understanding, Building and Managing Counterparty Risk

Savigne, A (2018) Modern Computational Finance, AAD and Parallel Simulations

Wen-Mei Hwu and Kirk, D (2018) Programming massively Parallel Processors. 3rd Edition.

Acknowledgments