Lab 4: Accelerating American-Style Option

By: Cormac Taylor

I pledge my honor that I have abided by the Stevens Honor System.

Overview

Pricing American-style options involves solving optimal stopping problems, where the holder can exercise the option at any time before expiration. Traditional methods like the Least-Squares Monte Carlo (LSMC/Longstaff-Schwartz) approach rely on backward dynamic programming, which is inherently sequential and computationally expensive. We explore GPU-accelerated quantization tree methods to overcome these limitations as seen in GPGPUs in computational finance: Massive parallel computing for American style options.

Key Challenge:

- Backward induction and regression steps in LSMC are sequential, limiting parallelization.

Solution:

- Quantization trees discretize the state space of the underlying asset using Voronoi partitions, enabling parallel computation of transition probabilities in a forward pass. Additionally, we exploit CUDA for massive parallelism in Monte Carlo simulations, nearest neighbor searches, and atomic counter updates.

Details

Algorithms

LSMC (Longstaff-Schwartz)

- CPU Implementation (`lsm_cpu`)
    - Simulates asset paths using Geometric Brownian Motion.
    - Applies backward induction at each timestep, regresses continuation values using in-the-money paths and updates cashflows.
    - Limitation is that regression and backward steps are sequential and dominate runtime.
- Partial GPU Implementation (`lsm_gpu`)
    - Parallelizes path simulation and regression steps.
    - Uses CUDA kernels for path generation, matrix operations (e.g., `compute_xty_lsm_gpu`), and cashflow updates.
    - Some bottlenecks are that the kernel launches are relatively small, meaning the launch/memory transfer overhead is significant. Additionally, atomic operations for regression matrix updates and limited parallelism in backward steps hold it back.

Quantization Tree

- Methodology:
    - Uses Voronoi grids to discretize the asset's state space at each timestep.
    - Estimates transition probabilities via Monte Carlo simulations by counting grid transitions (`compute_transitions_qt2` / `…_qt3`).

    - Computes option values via backward induction using precomputed transition matrices.
- GPU Optimizations:
    - `qt2`: Path-wise parallelization with brute-force nearest neighbor searches.
    - `qt3`: Time-layer parallelization using shared memory for faster grid access.
    - Atomic Operations: Used for accurate transition tracking without race conditions.

Code Structure

- `driver.cu`: Benchmarks all algorithms total cost, measuring runtime and GFLOPs.
- `lsm_cpu.cu` / `lsm_gpu.cu`: Implement LSMC with and without partial GPU acceleration.
- `qt2.cu` / `qt3.cu`: Implement quantization tree methods with different parallel strategies.

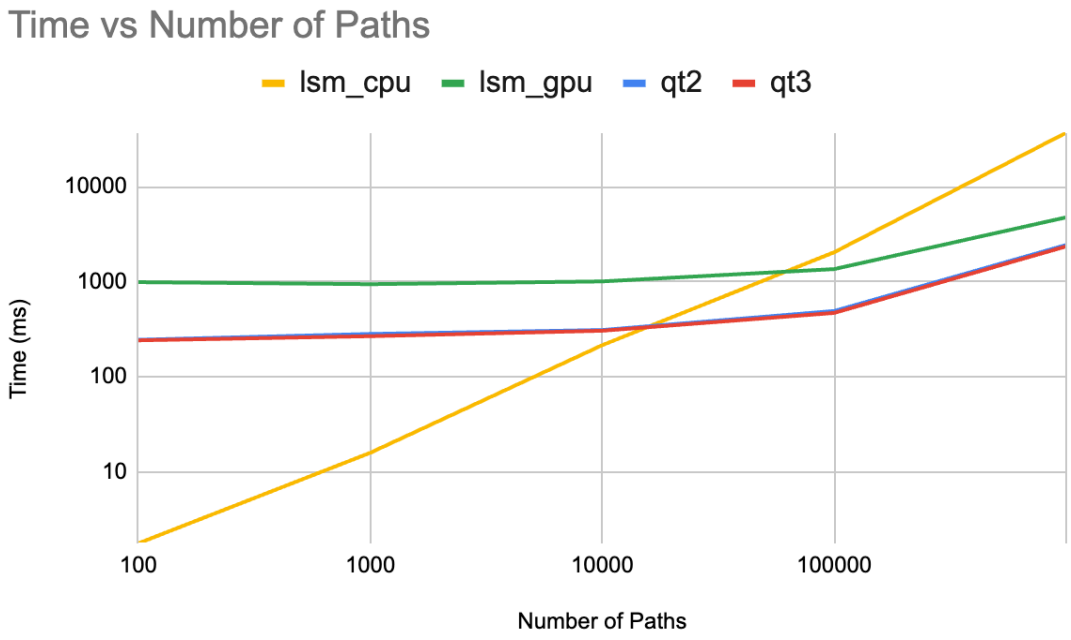
Results

Figure 1:

Algorithm	Runtime (ms)	Speed-Up vs. CPU
`lsm_cpu`	37,098	1
`lsm_gpu`	4,766	7.8
`qt2`	2,438	15.2
`qt3`	2,358	15.7

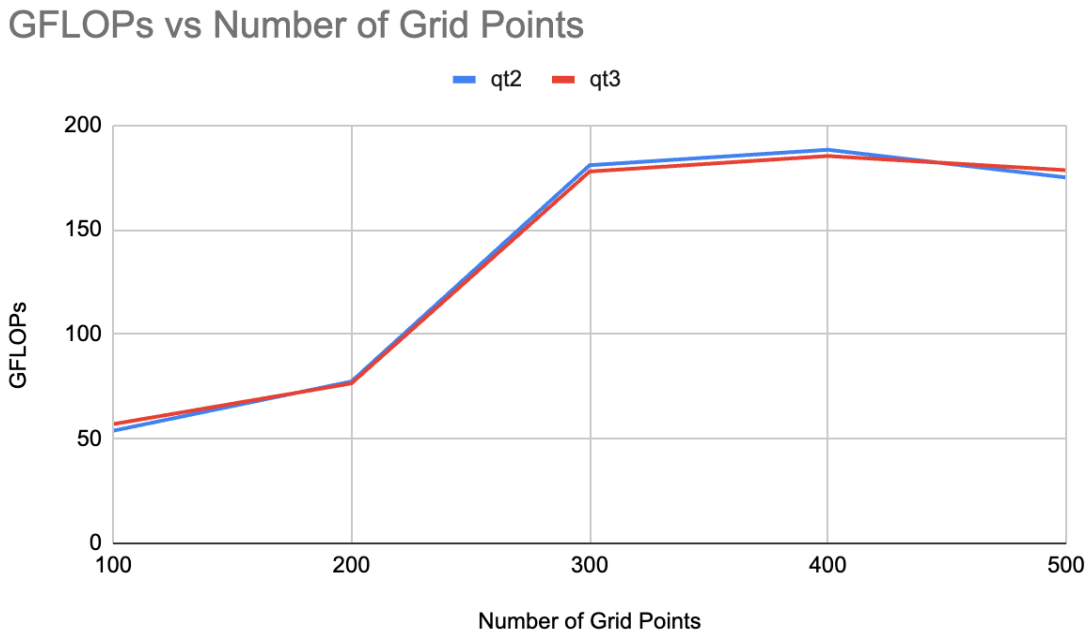
At scale with 1,000,000 Monte Carlo paths we see large benefits for parallelization. Quantization trees have 500 grid points per time step.

Figure 2:



Here we see how total execution time scales with larger Monte Carlo path quantities. Quantization trees have 500 grid points per time step.

Figure 3:



Here we see how GFLOP/s performance scales with larger grid point quantities. 100,000 Monte Carlo paths used throughout.

Key Observations

- Quantization tree methods (`qt2`, `qt3`) achieve 15× speed-up over the CPU-based LSMC.
- Brute-force nearest neighbor searches outperform kd-trees due to thread divergence issues on CUDA.

Discussion

We will now discuss the result of our testing, but will first reveal the hardware that was used in the testing. Testing was run on a server with 5 Tesla V100S-PCIE-32GB, each with the following specifications:

Figure 4:

Tesla V100S-PCIE-32GB	
(80) Multiprocessors, (64) CUDA Cores/MP	5120 CUDA Cores
GPU Max Clock rate	1597 MHz (1.60 GHz)
Total amount of global memory	32501 MBytes
Total amount of constant memory	65536 bytes
Total amount of shared memory per block	49152 bytes
Total number of registers available per block	65536
Warp size	32
Maximum number of threads per block	1024

On this GPU peak performance is:

$$CUDA\ Cores\ *\ Clock\ Rate(GHz)\ *\ FLOP\ per\ Core\ per\ Cycle$$

$$=\ 5120\ *\ 1.6\ *\ 2\ =\ 16384\ GFLOPS\ =\ 16.38\ TFLOPS$$

Quantization trees offer several advantages that make them well-suited for option pricing tasks. They support a parallel forward pass where transition probabilities are computed in a forward sweep, effectively decoupling dependencies between time layers. This design enhances memory efficiency by utilizing shared memory and minimizing expensive global memory accesses. Additionally, quantization tree methods scale well, with linear time complexity in the number of timesteps, in contrast to the quadratic cost associated with Least Squares Monte Carlo

(LSMC) regression steps. However, these benefits come with trade-offs. For instance, increasing grid resolution can improve pricing accuracy but at the cost of significantly more memory consumption—for example, using 500 grid points requires around 365MB of transition data. This forces use of global memory, opening opportunities for intelligent use of faster memory structures. A potentially promising next step is to develop a hybrid method that combines quantization trees with machine learning to generate adaptive grids dynamically.

This lab demonstrates that GPU-accelerated quantization tree methods significantly outperform traditional LSMC for pricing American-style options at scale. By enabling a parallel forward pass and optimizing memory access, `qt2` and `qt3` achieve real-time performance, making them well-suited for applications like high-frequency trading and real-time risk evaluation. Further hardware improvements and algorithmic innovations are expected to close the gap between theoretical accuracy and practical deployment.

Sources

G. Pagès and B. Wilbertz, "GPGPUs in computational finance: Massive parallel computing for American style options," arXiv preprint arXiv:1101.3228, 2011. [Online]. Available: https://arxiv.org/abs/1101.3228