Montecarlo Pricing Optimization

Parallel Programming Final Project

Group 23
David Lu T14902116
蔡琦皇 P13922006
楊翊廷 R14944018
詹淯翔 B13201026

Motivation: The Need for Speed

The Financial Problem: Flexibility vs. Cost

- Monte Carlo Methods are crucial for pricing complex options (e.g., Asian, European) due to their flexibility.
- The Bottleneck: High accuracy requires simulating millions of price paths with hundreds of time steps.
- CPU Reality: Simulations take seconds to minutes—too slow for real-world trading decisions.

Solution: Parallelization

Embarrassingly Parallel: Each simulation path is completely independent of others.

Perfect Fit for CUDA: GPUs are designed to handle massive numbers of parallel tasks simultaneously.

Project Goal: Maximize simulation throughput and explore CUDA's application in computational finance.

Problem description

Asset Price Dynamics: We model the underlying asset price St using the Geometric Brownian Motion (GBM)

$$S_{t+\Delta t} = S_t \cdot \exp\left((r - \tfrac{1}{2}\sigma^2)\Delta t + \sigma\sqrt{\Delta t}\, Z_t\right), \qquad Z_t \sim N(0,1)$$

Monte Carlo Method: Estimating the expected payoff of financial derivatives by simulating thousands of possible price paths and discounting them back to the present.

$$S^{(j)}_{t+\Delta t} = S^{(j)}_t \cdot \exp\left(\left(r - \frac{1}{2}\sigma_j^2\right)\Delta t + \sigma_j\sqrt{\Delta t}\, Z_t^{(j)}\right), \quad j = 1, \ldots, M$$

Computational Bottleneck: Real-time pricing of complex options (e.g., Asian or Multi-Asset) requires a massive number of simulations, leading to high latency on serial CPU implementations.

Objective

Goal: Identify the most efficient way to accelerate Monte Carlo simulations.

- **Performance Testing:** Benchmarking throughput and latency.
- **Scaling Strategy:** Transitioning from CPU to GPU, and finally to Multi-GPU.
- **Validation:** Ensuring accuracy remains consistent across all hardware platforms.

Implementation

We implemented three levels of parallelism:

- **OpenMP:** Multi-core CPU parallelization.
- **CUDA (Single GPU):** Massive threading for SIMT (Single Instruction, Multiple Threads).
- **Multi-GPU (4x CUDA):** Distributed workload across 4 GPUs to maximize throughput.

European Options

Definition: Can only exercise at maturity date T.

Path: Only the final price St matters.

Validation: Compare MC result with Black-Scholes formula, which is a closed-form solution of the result.

Asian Options

Definition: Payoff depends on the Average Price over time.

Complexity: The result has path dependency, must track price at every time step.

Cannot use Black-Scholes formula.

Memory Challenge: Requires frequent read/write during the path simulation.

Basket Options (Multi-Asset)

The Real-World Challenge

Definition: Option on a portfolio (e.g., Apple + Google + Tesla).

Characteristics:

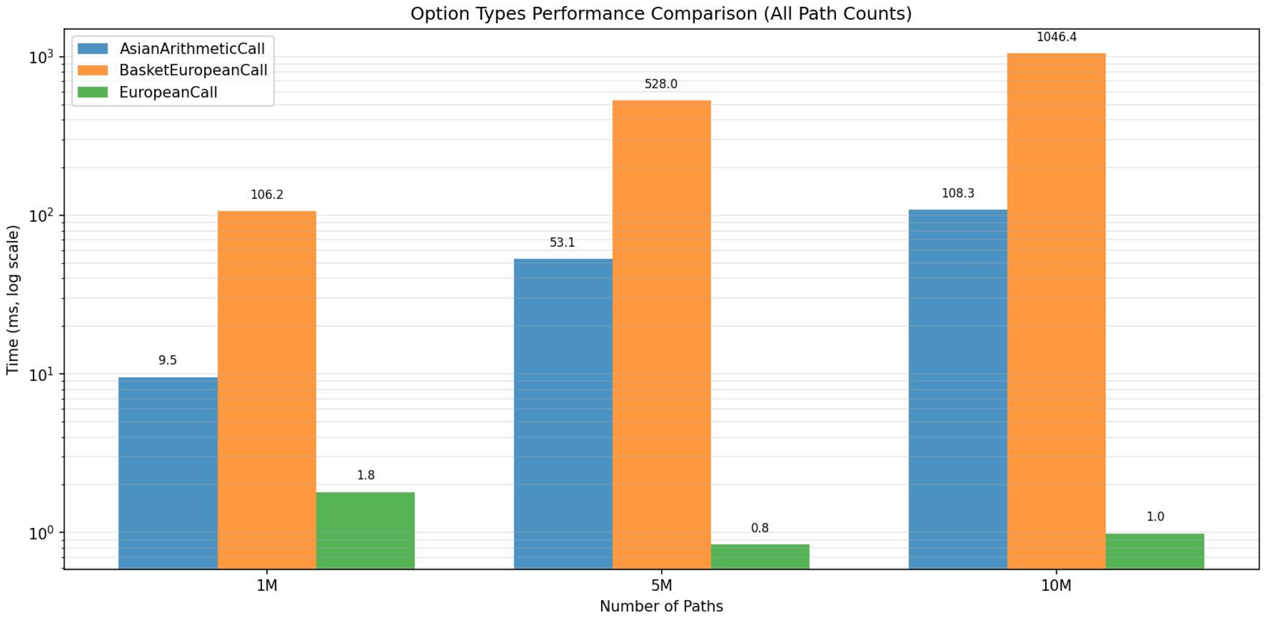
- Assets move together and has correlation
- Math Heavy:
    - Cholesky Decomposition (Matrix operations).
    - Computational cost explodes as assets increases

Experiments

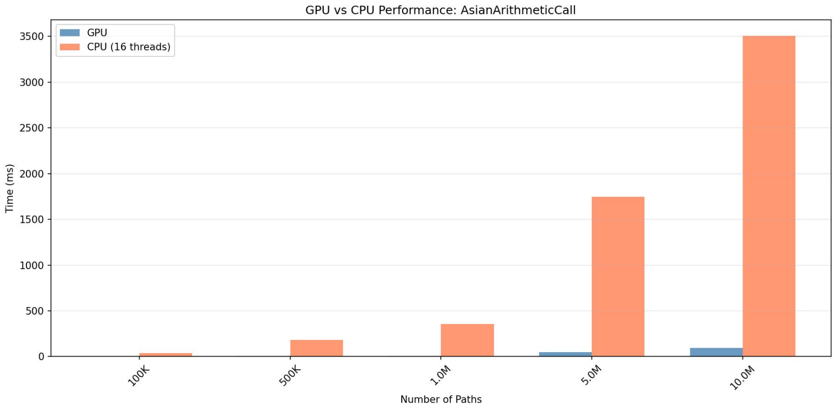
Testing 4 option types with increasing complexity:

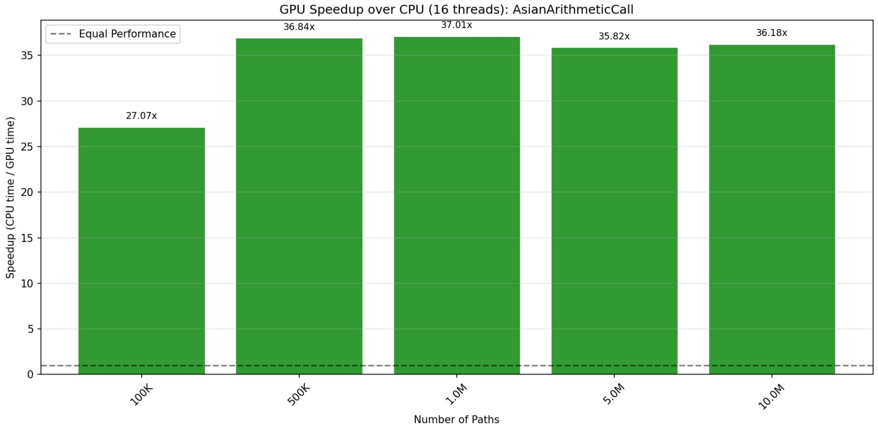
- European: Standard benchmark.
- Asian: Path-dependent (Arithmetic Average)
- Multi-Asset(Basket): High-dimensional simulation using Cholesky decomposition.

Result - Option Type Comparison

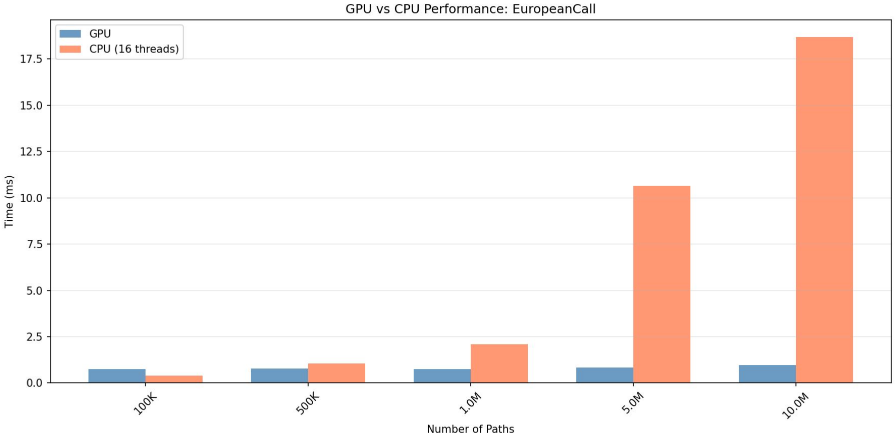


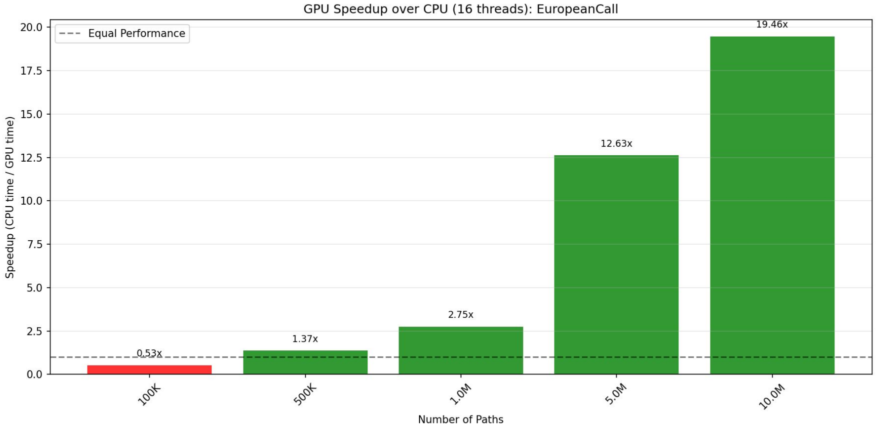
Result - GPU vs CPU - Asian Option



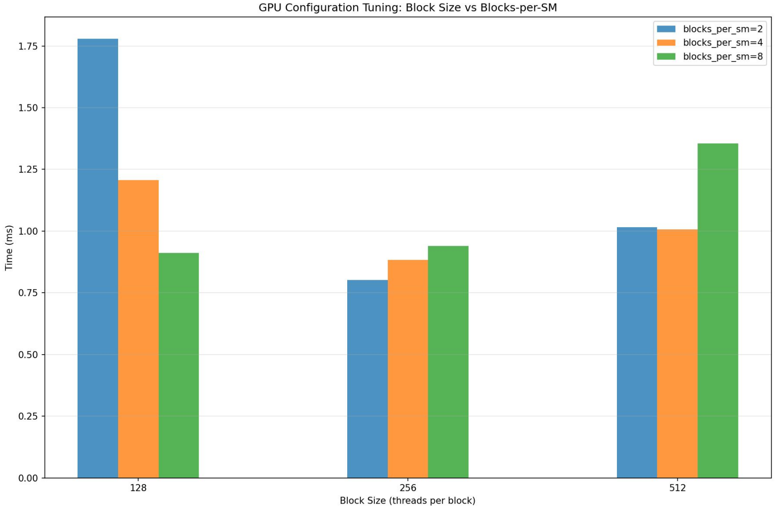


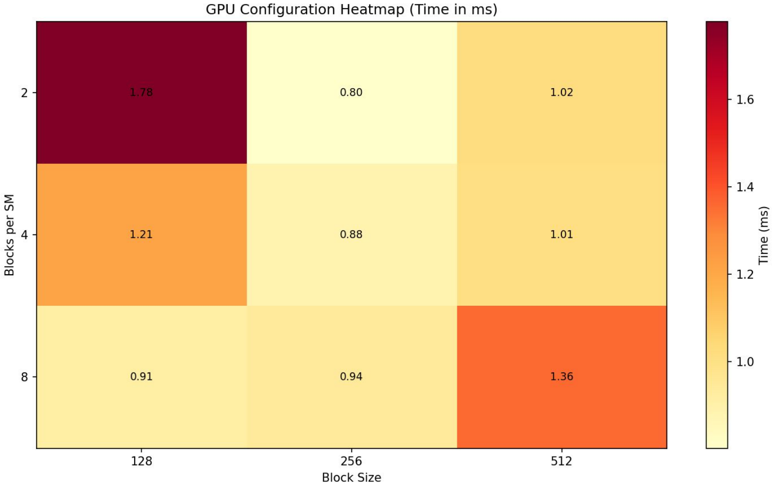
Result - GPU vs CPU - European Option





Result - Block Size Tuning





Conclusion

1. Runtime Comparison (Time Cost): Multi-Asset > European >> Asian(Multi-Asset requires the most computation time, while Asian is the fastest in this specific implementation.)
2. Speedup Characteristics
   - Asian: Consistent 35x speedup (Independent of path count).
   - European: Highly dependent on path count; GPU is faster while the paths are large.
3. Best Configuration:
   - Block Size: 256
   - Occupancy: 2 Blocks per SM
4. Using GPUs can significantly accelerate GBM-based Monte Carlo pricing.

Thanks for your attention