CUDA Implementation of Barrier Option Valuation using Jump-Diffusion Process and Brownian Bridge

GPU Technology Conference 21-23 September 2010, San Jose, California

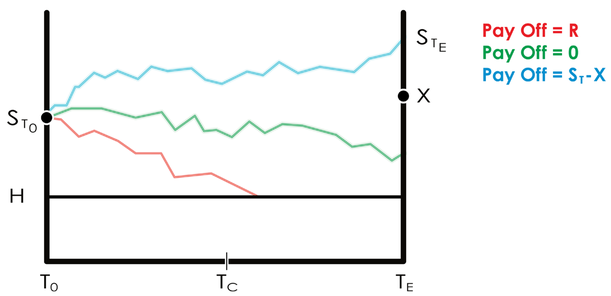
Dariusz Murakowski William Brouwer William Lynch Vincent Natoli

Introduction

The pace and volume of market trading have both steadily increased in recent years. The need for fast accurate approaches to valuation of complex financial instruments has never been greater. GPUs present an attractive option for accelerating these calculations. Monte Carlo valuation in particular maps well to the GPU architecture as independent trajectories can be associated with independent threads.

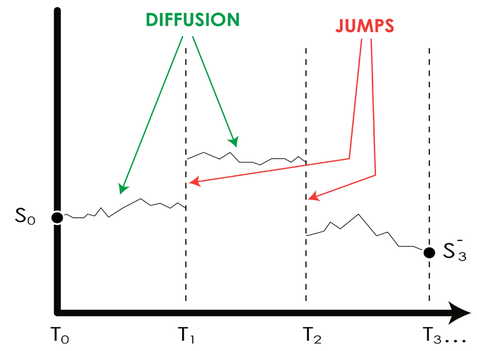
The purpose of this work is to evaluate the performance of GPU computing on a valuation model with elevated complexity. We follow the methodology first proposed by Metwally and Atiya, which addresses barrier options using an underlying jump-diffusion process and a Brownian bridge. We have developed optimized CPU and GPU implementations of the algorithm described by Metwally and have compared the performance.

What is a Barrier Option?



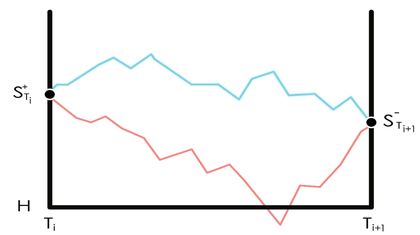
A barrier option's value depends on the price trajectory of the underlying asset. If the price falls below the barrier, H, at any time then the option pays out a rebate value R. If the price of the underlying option does not cross the barrier at any time up to the expiration time T_E then its value is $\max(S_{T_E} - X)$, where X is the strike price of the option.

Jump-Diffusion Processes



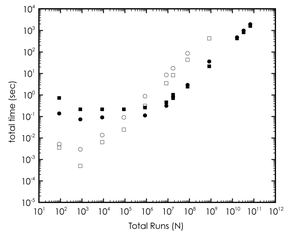
We include a Poisson distributed jump model in addition to the diffusion model of price movement. The jump model takes account of rare market events. Jumps occur in time intervals that are Poisson distributed. The jump value itself is normally distributed.

The Brownian Bridge



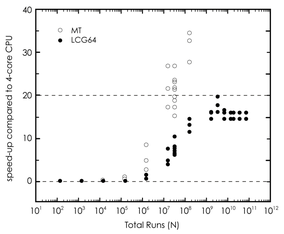
In between jumps prices move according to the Brownian diffusion model. If the asset price at the end of the diffusion interval is above the barrier, there is still a chance that it diffused through the barrier somewhere in the interval. This is accounted for using the Brownian bridge model which considers the first passage time of a diffusion process as well as the distribution of crossing times which may be calculated analytically.

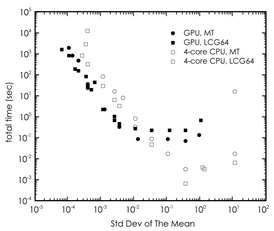
Results



Total Time vs Number of Monte Carlo runs. For large numbers of runs best results are found using the Mersenne Twister Random Number Generator on the GPU. GPU performance is more than 20x faster than CPU for a wide range of N.

Total Speedup of GPU implementation over CPU for Mersenne Twister and LCG64 RNG. We observe GPU performance exceeding CPU performance by up to 35x using the Mersenne Twister RNG. For very large N performance was about 15x greater than CPU.





Total run time vs standard deviation of the mean for GPU and CPU using Mersenne Twister and LCG64 RNGs. For cases where it is desired to reach a certain std dev., we show here that the GPU is significantly faster.

Conclusions

- ✓ Tesla C1060 GPU performance 15x-35x over quad core Nehalem
- ✓ Same Statistical Accuracy

STONE RIDGE TECHNOLOGY