HestonCUDA

William Chan, Irina Brinster, Erik Reed,
Jike Chong, and Matthew Dixon

Carnegie Mellon University
NASA AMES Research Park
Moffett Field, California 94305
williamchan@cmu.edu

December 4, 2011

Introduction - Problem Background

We want to price financial options/derivatives:

- Traditionally we use Black-Scholes Model
- Assumes Constant Volatility $\Rightarrow$ not accurate $\Rightarrow$ bad pricing $\Rightarrow$ bad profits

We want to use the Heston Model:

- Stochastic Volatility $\Rightarrow$ more accurate pricing $\Rightarrow$ profits
- Complex calculations $\Rightarrow$ slow $\Rightarrow$ bad profits

Goal: Speedup the Heston pricing model calculations with GPUs!

FFT or Quadrature

3 Methods of computation available:

1. Quadrature
2. Fast Fourier Transform
3. Monte Carlo

We will focus on Quadrature and FFT. It turns out these two methods are *embarassingly* parallelizable!

CUDA but not CUDA – Thrust!

CUDA is a pain to deal with... (block size, threads, warps, shared memory, etc...)

- Solution: NVIDIA Thrust!
  `http://developer.nvidia.com/Thrust`
- Write C++ style kernel functions w/o a need to deal with the CUDA messiness with most of the goodies of CUDA speedups!
- Tradeoff some performance for fast prototyping

Experimental Setup

- CPU: Intel Core 2 Quad Q8300 2.5 GHz
- GPU: NVIDIA GTX 460 1 GiB
- RAM: 4 GiB

Numerical Stability

Fast software is meaningless if it is not accurate!

Software	**Result**
Matlab Quadrature Reference	0.202871648173905
Matlab FFT Reference	0.203756595588233
CPU Quadrature	0.20287164817390**45**
GPU Quadrature	0.20287164817390**50**
CPU FFT	0.20375659558823**34**
GPU FFT	0.20375659558823**33**

GPUs are numerically stable relative to CPUs– bigger problem between Quadrature and FFT computation methods.

Quadrature Speedup

FFT Speedup

FFT is actually limited by workload– not enough work to feed all the CUDA cores!

Conclusion