Parallel Option Pricing with Fourier Space Time-stepping Method on Graphics Processing Units

Vladimir Surkov
University of Toronto

The First Workshop on Parallel and Distributed Computing in Finance
April 18, 2008

Joint work with Ken Jackson and Sebastian Jaimungal, University of Toronto

The Option Pricing Problem

- Option payoff is given by $\varphi(\mathbf{S})$
- Stock price follows an exponential Lévy model:

$$\mathbf{S}(t) = \mathbf{S}(0)e^{\mathbf{X}(t)}, \qquad \mathbf{X}(t) \text{ is a Lévy process}$$

The Option Pricing Problem

- Option payoff is given by $\varphi(\mathbf{S})$
- Stock price follows an exponential Lévy model:

$$\mathbf{S}(t) = \mathbf{S}(0)e^{\mathbf{X}(t)}, \qquad \mathbf{X}(t) \text{ is a Lévy process}$$

Generalizing PIDE for Lévy processes

$$\begin{cases} (\partial_t + \mathcal{L})\, v &= 0\,, \\ v(T, \mathbf{x}) &= \varphi(\mathbf{S}(0)\, e^{\mathbf{x}})\,, \end{cases}$$

where $\mathcal{L}$ is the infinitesimal generator of the Lévy process:

$$\mathcal{L}f(\mathbf{x}) = \left(\gamma \cdot \partial_{\mathbf{x}} + \tfrac{1}{2}\, \partial_{\mathbf{x}} \cdot \mathbf{C} \cdot \partial_{\mathbf{x}}\right) f(\mathbf{x}) + \int_{\mathbb{R}^n/\{\mathbf{0}\}} \left(f(\mathbf{x}+\mathbf{y}) - f(\mathbf{x}) - \mathbf{y} \cdot \partial_{\mathbf{x}} f(\mathbf{x})\, \mathbb{1}_{|\mathbf{y}|<1}\right) \nu(d\mathbf{y})$$

Numerical Methods for Option Pricing

Finite difference methods

- Alternating Direction Implicit-FFT - Andersen and Andreasen (2000)
- Implicit-Explicit (IMEX) - Cont and Tankov (2004)
- IMEX Runge-Kutta - Briani, Natalini, and Russo (2004)
- Fixed Point Iteration - d'Halluin, Forsyth, and Vetzal (2005)

Numerical Methods for Option Pricing

Finite difference methods

- Alternating Direction Implicit-FFT - Andersen and Andreasen (2000)
- Implicit-Explicit (IMEX) - Cont and Tankov (2004)
- IMEX Runge-Kutta - Briani, Natalini, and Russo (2004)
- Fixed Point Iteration - d'Halluin, Forsyth, and Vetzal (2005)

Quadrature methods

- Reiner (2001)
- QUAD - Andricopoulos, Widdicks, Duck, and Newton (2003)
- Q-FFT - O'Sullivan (2005)

Numerical Methods for Option Pricing

Finite difference methods

- Alternating Direction Implicit-FFT - Andersen and Andreasen (2000)
- Implicit-Explicit (IMEX) - Cont and Tankov (2004)
- IMEX Runge-Kutta - Briani, Natalini, and Russo (2004)
- Fixed Point Iteration - d'Halluin, Forsyth, and Vetzal (2005)

Quadrature methods

- Reiner (2001)
- QUAD - Andricopoulos, Widdicks, Duck, and Newton (2003)
- Q-FFT - O'Sullivan (2005)

Transform-based methods

- Carr and Madan (1999)
- Raible (2000)
- Lewis (2001)

Infinitesimal Generator and Characteristic Exponent

The characteristic exponent of a Lévy-Khinchin representation can be factored from a Fourier transform of the operator (Sato 1999)

$$\mathcal{F}[\mathcal{L}v](\tau, \boldsymbol{\omega}) = \psi(\boldsymbol{\omega})\mathcal{F}[v](\tau, \boldsymbol{\omega})$$

where

$$\Psi(\boldsymbol{\omega}) = i\boldsymbol{\gamma} \cdot \boldsymbol{\omega} + \tfrac{1}{2}\, \boldsymbol{\omega} \cdot \mathbf{C} \cdot \boldsymbol{\omega} + \int_{\mathbb{R}^n} \left(e^{i\boldsymbol{\omega}\cdot\mathbf{y}} - 1 - i\mathbf{y} \cdot \boldsymbol{\omega}\ \mathbb{1}_{|\mathbf{y}|<1}\right) \nu(d\mathbf{y})$$

Infinitesimal Generator and Characteristic Exponent

The characteristic exponent of a Lévy-Khinchin representation can be factored from a Fourier transform of the operator (Sato 1999)

$$\mathcal{F}[\mathcal{L}v](\tau,\boldsymbol{\omega}) = \psi(\boldsymbol{\omega})\mathcal{F}[v](\tau,\boldsymbol{\omega})$$

where

$$\Psi(\boldsymbol{\omega}) = i\gamma\cdot\boldsymbol{\omega} + \tfrac{1}{2}\,\boldsymbol{\omega}\cdot\mathbf{C}\cdot\boldsymbol{\omega} + \int_{\mathbb{R}^n}\left(e^{i\boldsymbol{\omega}\cdot\mathbf{y}} - 1 - i\mathbf{y}\cdot\boldsymbol{\omega}\;\mathbb{1}_{|\mathbf{y}|<1}\right)\nu(d\mathbf{y})$$

Model	**Characteristic Exponent** $\psi(\omega)$
Black-Scholes-Merton	$i\mu\omega - \frac{\sigma^2\omega^2}{2}$
Merton Jump-Diffusion	$i\mu\omega - \frac{\sigma^2\omega^2}{2} + \lambda(e^{i\tilde{\mu}\omega-\tilde{\sigma}^2\omega^2/2} - 1)$
Variance Gamma	$-\frac{1}{\kappa}\log(1 - i\mu\kappa\omega + \frac{\sigma^2\kappa\omega^2}{2})$
CGMY	$C\Gamma(-Y)\left[(M-i\omega)^Y - M^Y + (G+i\omega)^Y - G^Y\right]$

Numerical Method Derivation

- Apply the Fourier transform to the pricing PIDE

$$\begin{cases} \partial_t \mathcal{F}[v](t,\omega) + \Psi(\omega)\mathcal{F}[v](t,\omega) &= 0\ , \\ \mathcal{F}[v](T,\omega) &= \mathcal{F}[\varphi](\omega) \end{cases}$$

Numerical Method Derivation

- Apply the Fourier transform to the pricing PIDE

$$\begin{cases} \partial_t \mathcal{F}[v](t,\omega) + \Psi(\omega)\mathcal{F}[v](t,\omega) &= 0\ , \\ \mathcal{F}[v](T,\omega) &= \mathcal{F}[\varphi](\omega) \end{cases}$$

- Resulting ODE has explicit solution

$$\mathcal{F}[v](t_1,\omega) = \mathcal{F}[v](t_2,\omega)\cdot e^{(t_2-t_1)\Psi(\omega)}$$

Numerical Method Derivation

- Apply the Fourier transform to the pricing PIDE

$$\begin{cases} \partial_t \mathcal{F}[v](t,\boldsymbol{\omega}) + \Psi(\boldsymbol{\omega})\mathcal{F}[v](t,\boldsymbol{\omega}) &= 0\ , \\ \mathcal{F}[v](T,\boldsymbol{\omega}) &= \mathcal{F}[\varphi](\boldsymbol{\omega}) \end{cases}$$

- Resulting ODE has explicit solution

$$\mathcal{F}[v](t_1,\boldsymbol{\omega}) = \mathcal{F}[v](t_2,\boldsymbol{\omega}) \cdot e^{(t_2-t_1)\Psi(\boldsymbol{\omega})}$$

- Apply the inverse Fourier transform

$$v(t_1,\mathbf{x}) = \mathcal{F}^{-1}\left\{\mathcal{F}[v](t_2,\boldsymbol{\omega}) \cdot e^{(t_2-t_1)\Psi(\boldsymbol{\omega})}\right\}(\mathbf{x})$$

Numerical Method Derivation

- Apply the Fourier transform to the pricing PIDE

$$\begin{cases} \partial_t \mathcal{F}[v](t,\omega) + \Psi(\omega)\mathcal{F}[v](t,\omega) &= 0\,, \\ \mathcal{F}[v](T,\omega) &= \mathcal{F}[\varphi](\omega) \end{cases}$$

- Resulting ODE has explicit solution

$$\mathcal{F}[v](t_1,\omega) = \mathcal{F}[v](t_2,\omega)\cdot e^{(t_2-t_1)\Psi(\omega)}$$

- Apply the inverse Fourier transform

$$v(t_1,\mathbf{x}) = \mathcal{F}^{-1}\left\{\mathcal{F}[v](t_2,\omega)\cdot e^{(t_2-t_1)\Psi(\omega)}\right\}(\mathbf{x})$$

Fourier space time-stepping (FST) method

$$v^{n-1} = \mathrm{FFT}^{-1}[\mathrm{FFT}[v^n]\cdot e^{\Psi\,\Delta t}]$$

European Call Convergence Results

N	Value	Change	$\log_2$Ratio	CPU-Time
2048	0.04261423			0.002
4096	0.04263998	0.000026		0.005
8192	0.04264641	0.000006	2.0018	0.010
16384	0.04264801	0.000002	2.0010	0.019
32768	0.04264841	0.000000	2.0011	0.038

- *Option:* European call $S = 1.0, K = 1.0, T = 0.2$
- *Model:* Kou jump-diffusion
  $\sigma = 0.2, \lambda = 0.2, p = 0.5, \eta_- = 3, \eta_+ = 2, r = 0.0$
- *Quoted Price:* 0.0426761 Almendral and Oosterlee (2005)

American Put Convergence Results

N	M	Value	Change	$\log_2$Ratio	CPU-Time
2048	128	9.22478538			0.027
4096	256	9.22523484	0.0004495		0.109
8192	512	9.22538196	0.0001471	1.6114	0.451
16384	1024	9.22542478	0.0000428	1.7808	1.869
32768	2048	9.22543516	0.0000104	2.0444	8.195

- *Option:* American put $S = 90.0, K = 98.0, T = 0.25$
- *Model:* CGMY
  $C = 0.42, G = 4.37, M = 191.2, Y = 1.0102, r = 0.06$
- *Quoted Price:* 9.2254803 Forsyth, Wan, and Wang (2007)

Spread Option

- Payoff depends on difference of two stock prices

$$\varphi(S_1(T), S_2(T)) = \max(\alpha S_2(T) - \beta S_1(T) - K, 0)$$

Spread Option

- Payoff depends on difference of two stock prices

$$\varphi(S_1(T), S_2(T)) = \max(\alpha S_2(T) - \beta S_1(T) - K, 0)$$

- Stock price process is a 2D Merton jump-diffusion process

$$\frac{dS_i(t)}{S_i(t-)} = \mu_i dt + \sigma_i dW_i(t) + (J_i - 1)dN_i(t)$$

Spread Option

- Payoff depends on difference of two stock prices

$$\varphi(S_1(T), S_2(T)) = \max(\alpha S_2(T) - \beta S_1(T) - K, 0)$$

- Stock price process is a 2D Merton jump-diffusion process

$$\frac{dS_i(t)}{S_i(t-)} = \mu_i dt + \sigma_i dW_i(t) + (J_i - 1)dN_i(t)$$

- Characteristic exponent is given by:

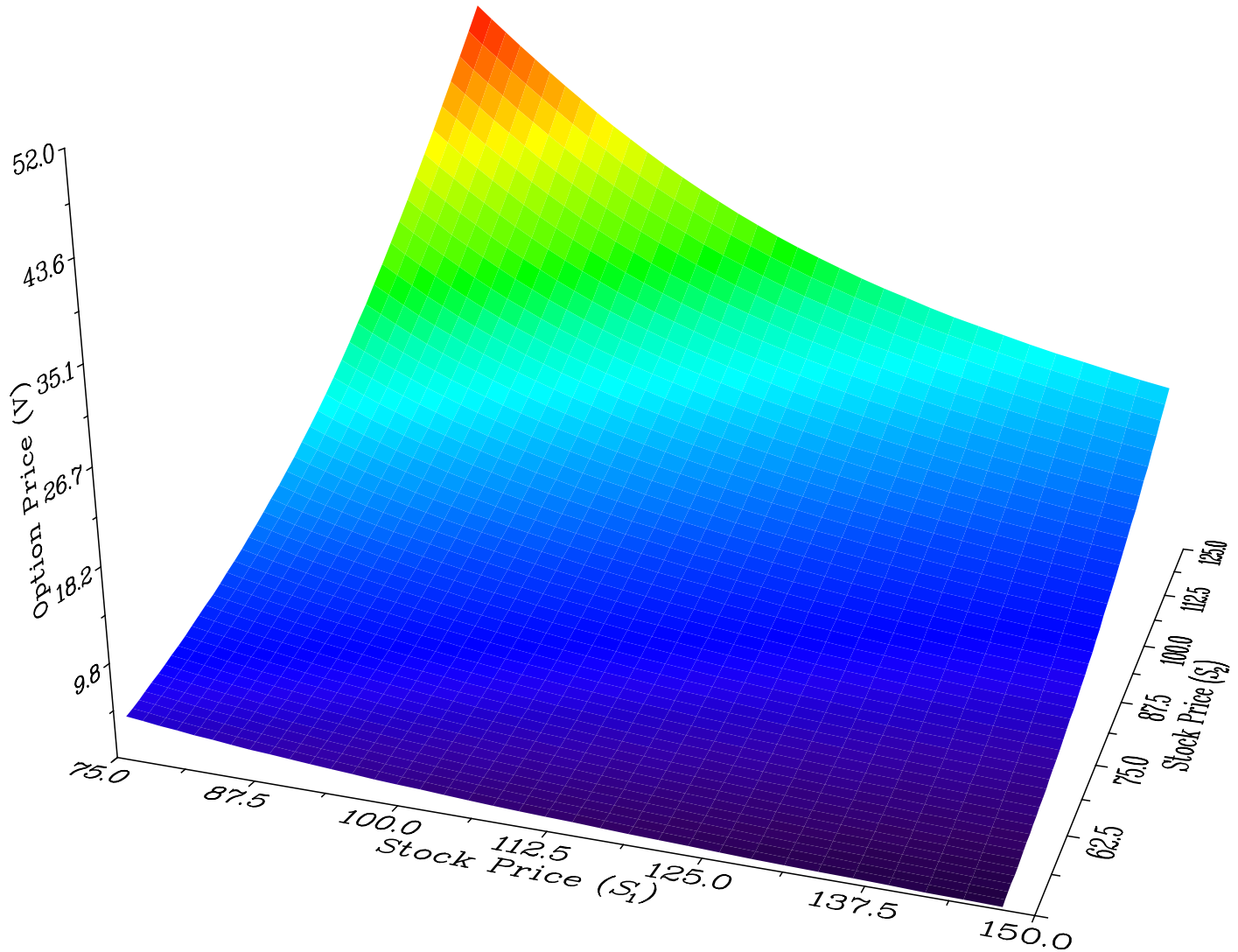
$$\begin{aligned}\Psi(\omega_1, \omega_2) &= i(\mu_1 - \frac{\sigma_1^2}{2})\omega_1 + i(\mu_2 - \frac{\sigma_2^2}{2})\omega_2 - \frac{\sigma_1^2\omega_1^2}{2} - \rho\sigma_1\sigma_2\omega_1\omega_2 - \frac{\sigma_2^2\omega_2^2}{2} \\ &\quad + \lambda_1 \left(e^{i\tilde{\mu}_1\omega_1 - \tilde{\sigma}_1^2\omega_2^2/2} - 1\right) + \lambda_2 \left(e^{i\tilde{\mu}_2\omega_2 - \tilde{\sigma}_2^2\omega_2^2/2} - 1\right)\end{aligned}$$

Spread Option Convergence Results

N	Value	Change	$\log_2$Ratio	CPU-Time
512	15.03639950			0.880245
1024	15.02776432	0.008635		2.821585
2048	15.02919574	0.001431	2.5928	11.919293
4096	15.02924971	0.000054	4.7293	48.371978
8192	15.02924214	0.000008	2.8345	209.806361

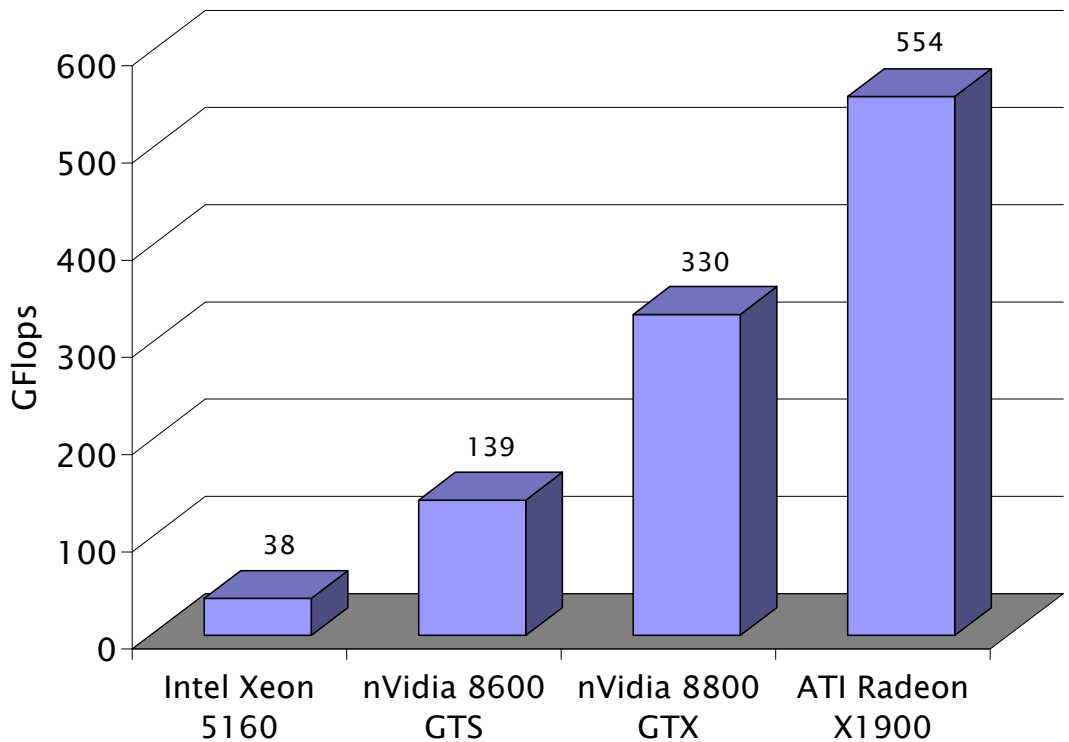
- *Option:* Spread call $S_1 = 96.0, S_2 = 100.0, K = 2.0, T = 1.0$
- *Model:* Merton jump-diffusion
  $\sigma_1 = 0.1, q_1 = 0.05, \lambda_1 = 0.25, \tilde{\mu_1} = -0.13, \tilde{\sigma_1} = 0.37, \sigma_2 = 0.2, q_2 = 0.05, \lambda_2 = 0.5, \tilde{\mu_2} = 0.11, \tilde{\sigma_2} = 0.41, \rho = 0.5, r = 0.1$
- *Kirk's Formula Price:* 15.03001533

Spread Option Price Surface



Graphics Processing Units (GPUs) Overview

- Effective in highly parallel, compute-intensive applications
- Rich functionality, not restricted to graphics rendering
- FST utilizes efficient FFT evaluation through NVIDIA CUDA library



FST-GPU Method for European options

Input: Option payoff v^T, characteristic exponent Ψ
Output: Option values v^0

Upload v^T, $e^{\Psi \Delta t}$ to GPU

$v^0 \leftarrow \text{FFT}^{-1}[\text{FFT}[v^T] \cdot e^{\Psi \Delta t}]$

Download v^0 from GPU

return v^0

FST-GPU Method for American options

Input: Option payoff v^T, characteristic exponent Ψ
Output: Option values v^0

Upload v^T, $e^{\Psi \Delta t}$ to GPU

$v^N \leftarrow v^T$

for $n \leftarrow M$ **to** 1 **do**

$\tilde{v}^n \leftarrow \text{FFT}^{-1}[\text{FFT}[v^n] \cdot e^{\Psi \Delta t}]$

$v^{n-1} = \max\{\tilde{v}^n, v^T\}$

end

Download v^0 from GPU

return v^0

Performance Results for Single-asset Options

- Pricing single European option

Grid points	Price	CPU Time (msec.)	GPU Time (msec.)
4096	10.6783	4.09	7.09
8192	10.6781	8.38	10.89
16384	10.6782	16.54	18.83
32768	10.6782	33.24	35.21

Performance Results for Single-asset Options

- Pricing single European option

Grid points	Price	CPU Time (msec.)	GPU Time (msec.)
4096	10.6783	4.09	7.09
8192	10.6781	8.38	10.89
16384	10.6782	16.54	18.83
32768	10.6782	33.24	35.21

- Pricing single American option

Grid points	Time points	Price	CPU time (sec.)	GPU time (sec.)
4096	512	2.3577	0.17	0.07
8192	1024	2.3580	0.60	0.25
16384	2048	2.3581	3.27	1.11
32768	4096	2.3582	14.02	4.10

Performance Results for Multi-asset Options

- Pricing single European spread option

Grid points	Price	CPU time (sec.)	GPU time (sec.)
256^2	15.5678	0.10	0.11
512^2	15.5723	0.47	0.45
1024^2	15.5729	1.98	1.87
2048^2	15.5729	8.32	7.90

Performance Results for Multi-asset Options

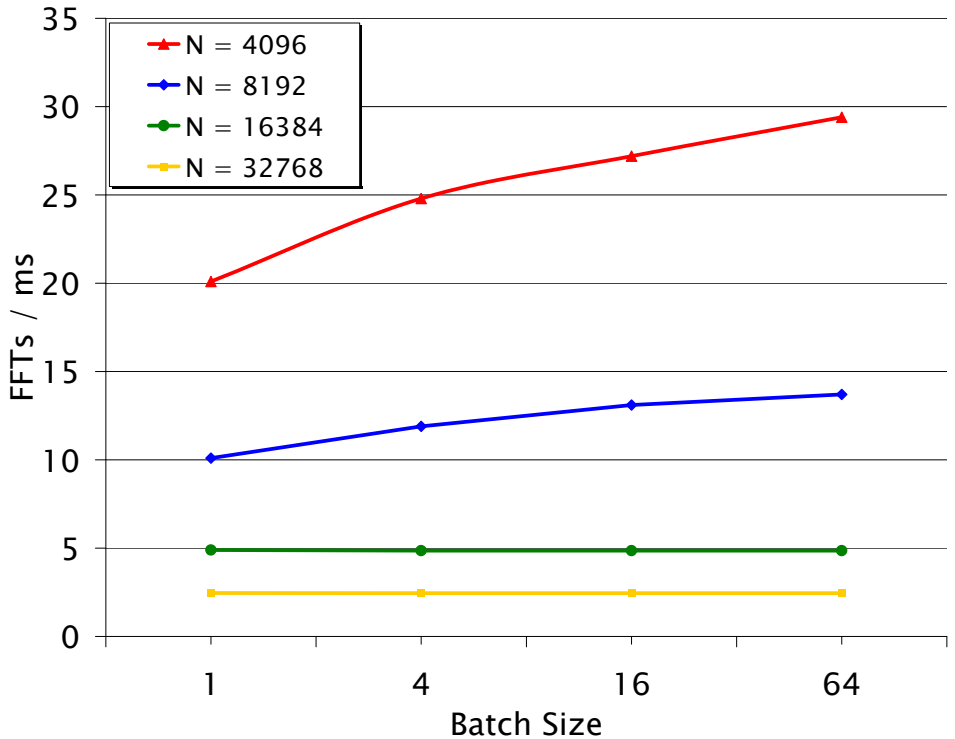
- Pricing single European spread option

Grid points	Price	CPU time (sec.)	GPU time (sec.)
256^2	15.5678	0.10	0.11
512^2	15.5723	0.47	0.45
1024^2	15.5729	1.98	1.87
2048^2	15.5729	8.32	7.90

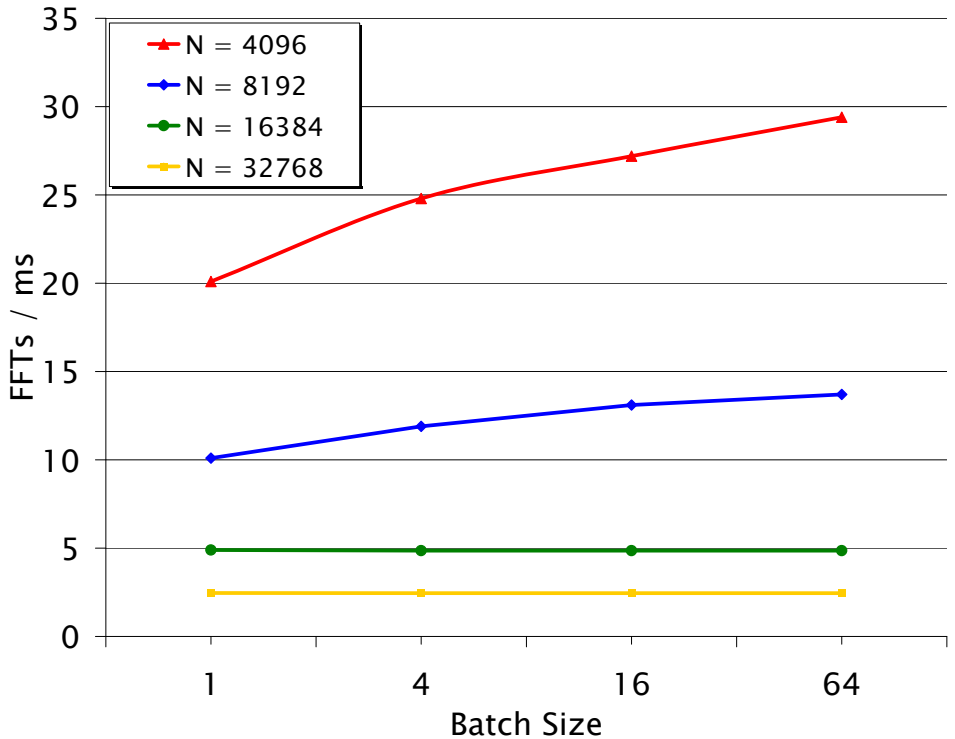
- Pricing single American "double-trigger stop-loss" option

Grid points	Time points	Price	CPU time (sec.)	GPU time (sec.)
256^2	128	3.5256	1.0	0.2
512^2	256	3.7416	9.9	1.4
1024^2	512	3.7785	82.5	10.9
2048^2	1024	3.7935	713.8	142.4

Performance Results for Batched Option Pricing



Performance Results for Batched Option Pricing



Batch size	GPU time (msec.)	Options/sec.
4	20.21	197
8	29.35	272
16	50.78	315
32	94.20	339
64	177.78	360

FST-GPU Method Summary

- Stable and robust, even for options with discontinuous payoffs
- Can be extended to exotic, multi-dimensional and regime-switching problems in a natural manner
- Easily applied to various stochastic processes
- GPUs can be efficiently leveraged to attain high computational throughput