Parallel Option Pricing with Fourier Space Time-stepping Method on GPUs

Vladimir Surkov
University of Toronto

Second SHARCNET Symposium on GPU and Cell Computing
May 20, 2009

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

Option Pricing Problem

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

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

Option Price as Expectation

$$V(t, \mathbf{S}(t)) = \mathbb{E}_t^{\mathbb{Q}}\left[e^{-r(T-t)}\,\varphi(\mathbf{S}(T))\right]$$

The formulation leads to Monte Carlo and tree methods

$$\mathbb{E}_t^{\mathbb{Q}}\left[e^{-r(T-t)}\,\varphi(\mathbf{S}(T))\right] \approx \frac{1}{N}\sum_{n=1}^{N} e^{-r(T-t)}\,\varphi(\mathbf{S}^{\{n\}}(T))$$

Dual Option Pricing Problem

Option Price as Solution to PIDE

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

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

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

The advantages of this formulation are:

- Rapidly converging numerical methods
- Allows for pricing of highly path-dependent and/or exercisable options

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			0.001
4096	0.04263998	0.000026		0.002
8192	0.04264641	0.000006	2.0018	0.004
16384	0.04264801	0.000002	2.0010	0.007
32768	0.04264841	0.000000	2.0011	0.014

- *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			0.009
4096	256	9.22523484	0.0004495		0.035
8192	512	9.22538196	0.0001471	1.6114	0.151
16384	1024	9.22542478	0.0000428	1.7808	0.629
32768	2048	9.22543516	0.0000104	2.0444	2.733

- *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:

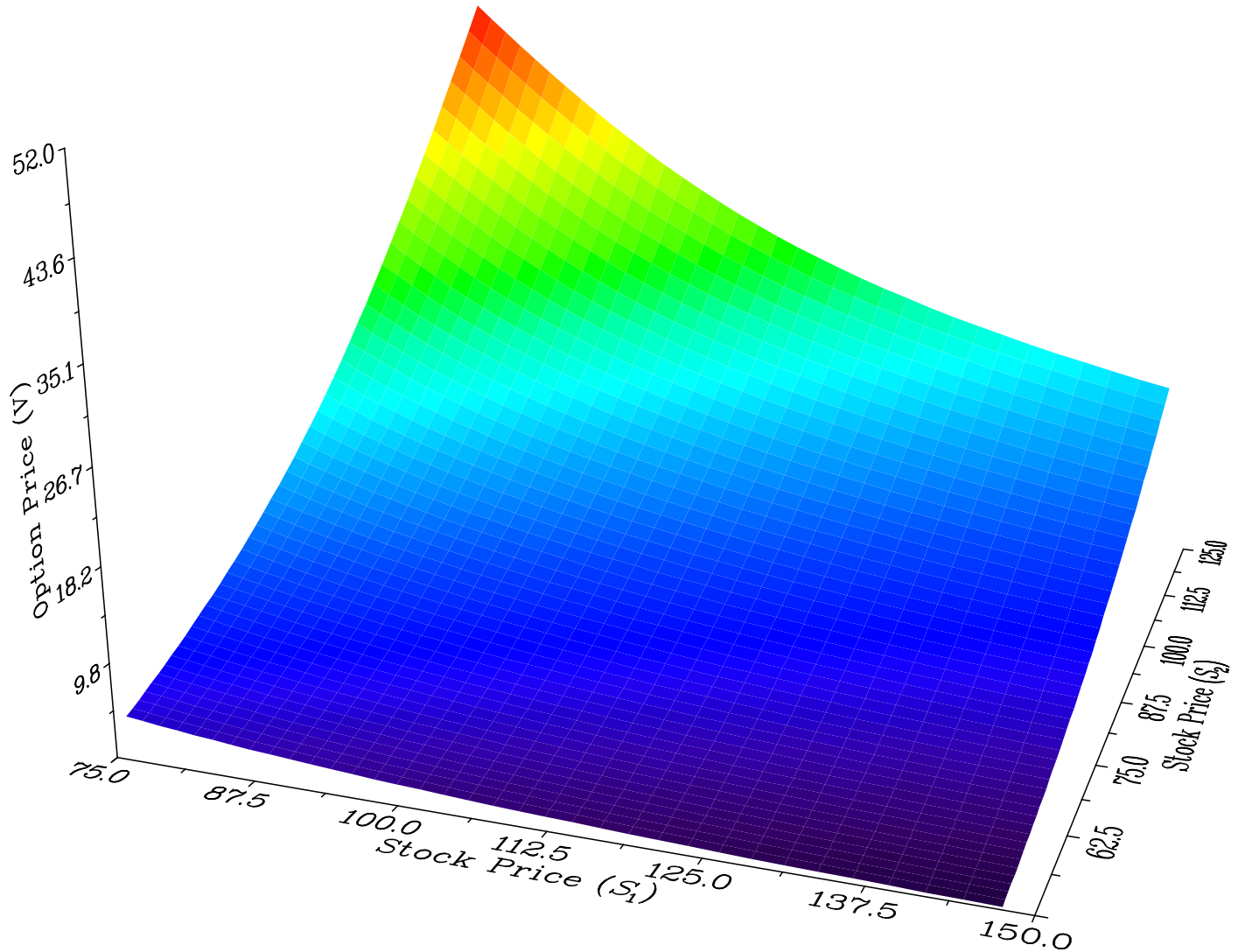
$$\begin{aligned}\Psi(\omega_1, \omega_2) &= i(\mu_1 - \frac{\sigma_1^2}{2})\omega_1 + i(\mu_2 - \frac{\sigma_2^2}{2})\omega_2 - \frac{\sigma_1^2\omega_1^2}{2} - \rho\sigma_1\sigma_2\omega_1\omega_2 - \frac{\sigma_2^2\omega_2^2}{2} \\ &\quad + \lambda_1 \left(e^{i\tilde{\mu}_1\omega_1 - \tilde{\sigma}_1^2\omega_2^2/2}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

Spread Option Convergence Results

N	Value	Change	$\log_2$Ratio	CPU-Time
512	15.03639950			0.187
1024	15.02776432	0.008635		0.749
2048	15.02919574	0.001431	2.5928	2.972
4096	15.02924971	0.000054	4.7293	12.123
8192	15.02924214	0.000008	2.8345	50.010

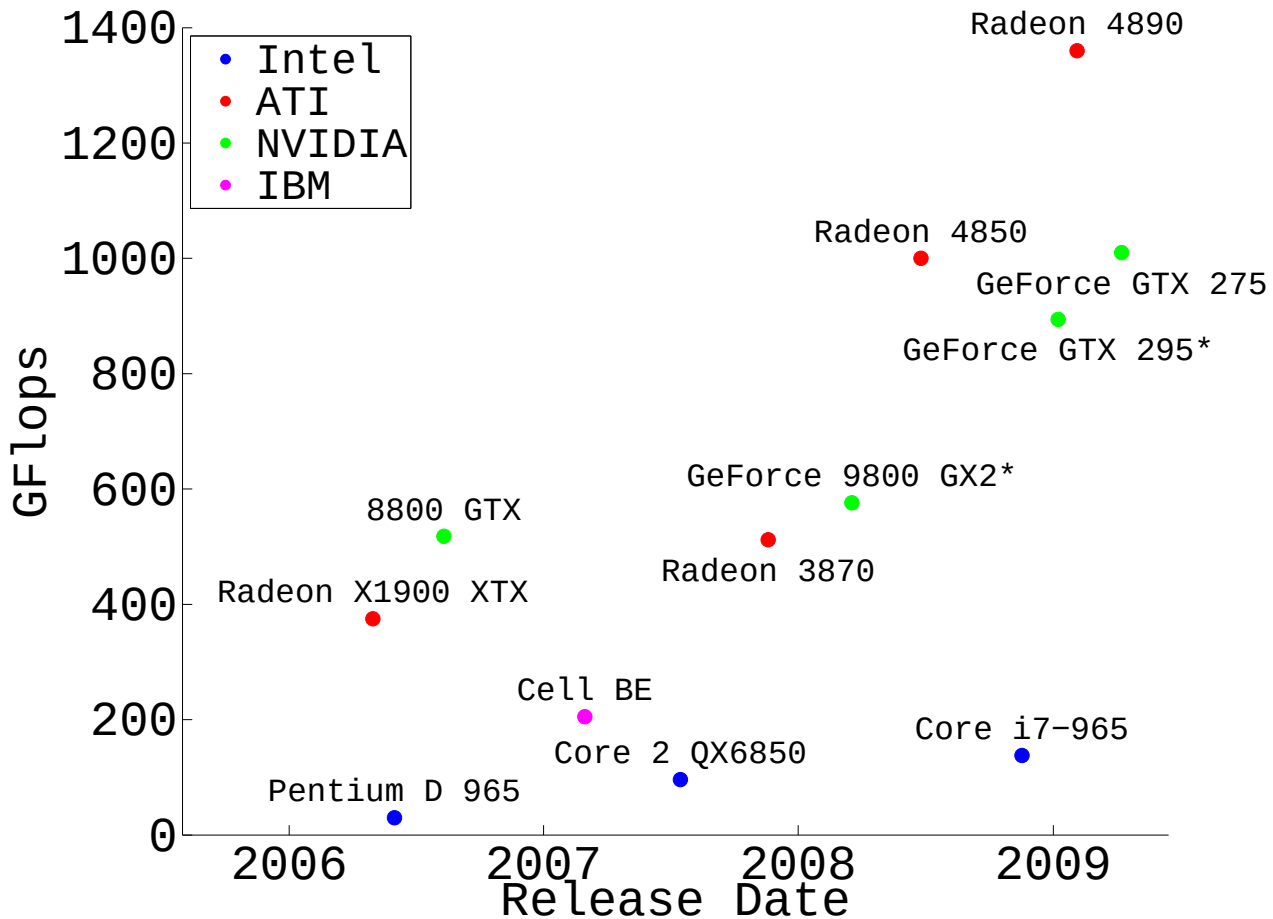
- *Option:* Spread call $S_1 = 96.0, S_2 = 100.0, K = 2.0, T = 1.0$
- *Model:* Merton jump-diffusion
  $\sigma_1 = 0.1, q_1 = 0.05, \lambda_1 = 0.25, \tilde{\mu_1} = -0.13, \tilde{\sigma_1} = 0.37, \sigma_2 = 0.2, q_2 = 0.05, \lambda_2 = 0.5, \tilde{\mu_2} = 0.11, \tilde{\sigma_2} = 0.41, \rho = 0.5, r = 0.1$
- *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.)
2048	7.28155746	1.167	1.178
4096	7.27979297	1.734	1.932
8192	7.28005799	3.353	3.501
16384	7.28011385	6.601	6.519
32768	7.28012262	13.234	12.642

Performance Results for Single-asset Options

- Pricing single European option

Grid points	Price	CPU Time (msec.)	GPU Time (msec.)
2048	7.28155746	1.167	1.178
4096	7.27979297	1.734	1.932
8192	7.28005799	3.353	3.501
16384	7.28011385	6.601	6.519
32768	7.28012262	13.234	12.642

- Pricing single American option

Grid points	Time points	Price	CPU time (sec.)	GPU time (sec.)
2048	128	8.01846275	0.009	0.017
4096	512	8.01147970	0.077	0.075
8192	2048	8.01337394	0.656	0.343
16384	8192	8.01402855	4.711	1.720
32768	32768	8.01391362	47.216	10.601

Performance Results for Multi-asset Options

- Pricing single European spread option

Grid points	Price	CPU time (sec.)	GPU time (sec.)
512^2	14.46794939	0.157	0.134
1024^2	14.46916508	0.571	0.466
2048^2	14.46924603	2.155	1.732
4096^2	14.46924541	8.585	N/A
8192^2	14.46924541	35.576	N/A

Performance Results for Multi-asset Options

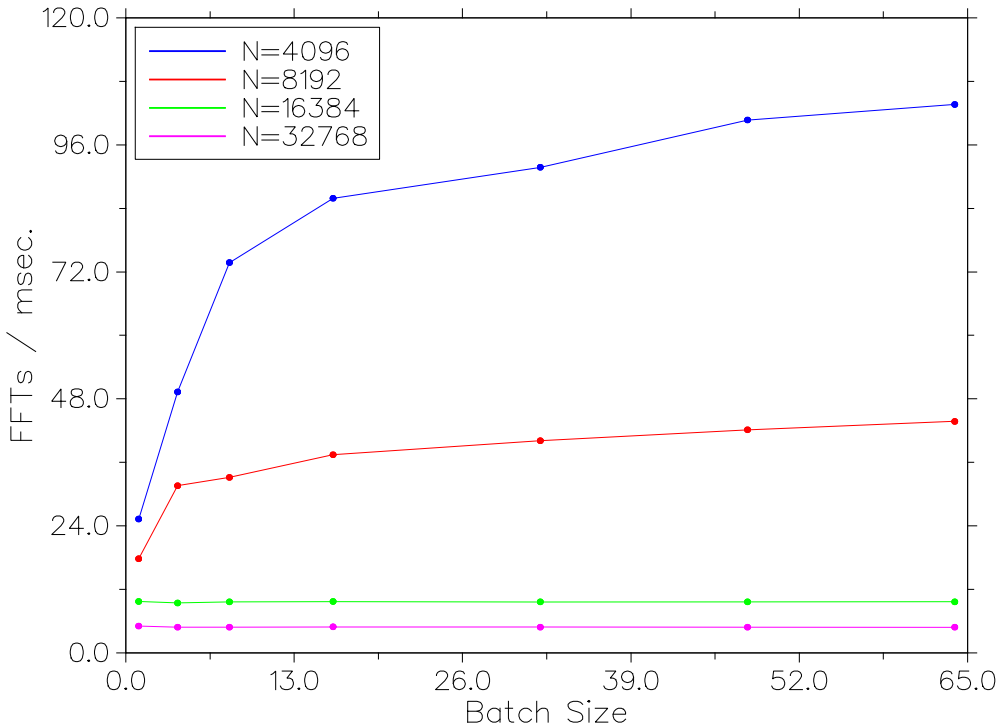
- Pricing single European spread option

Grid points	Price	CPU time (sec.)	GPU time (sec.)
512^2	14.46794939	0.157	0.134
1024^2	14.46916508	0.571	0.466
2048^2	14.46924603	2.155	1.732
4096^2	14.46924541	8.585	N/A
8192^2	14.46924541	35.576	N/A

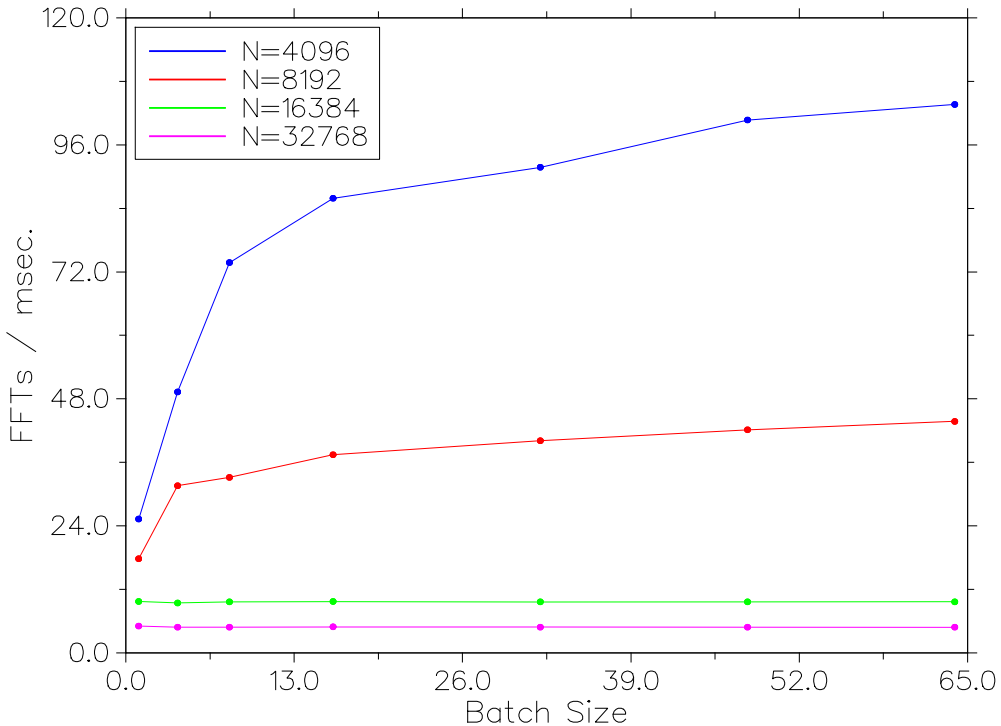
- Pricing single American "double-trigger stop-loss" option

Grid points	Time points	Price	CPU time (sec.)	GPU time (sec.)
512^2	64	2.53730898	1.097	0.258
1024^2	256	2.67880465	20.087	1.746
2048^2	1024	2.74424933	326.903	31.246
4096^2	4096	2.77366599	6539.073	N/A

Performance Results for Batched Option Pricing



Performance Results for Batched Option Pricing



Batch size	GPU time (msec.)	Options/sec.
1	3.51	284
4	6.12	653
16	18.39	870
64	66.93	956
256	261.52	979

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