On a GPU Acceleration of the Stochastic Grid Bundling Method

Álvaro Leitao Rodríguez and Cornelis W. Oosterlee

Technische Universiteit Delft (TU Delft) - Centrum Wiskunde & Informatica (CWI)

13/6/2014

Outline

Introduction and Motivation

- Multi-dimensional early-exercise option contracts.
- Increasing the dimensionality.
  - Counterparty Valuation Adjustment (CVA).
- SGBM becomes expensive.
- Solution: parallelization of the method.
- General-Purpose computing on Graphics Processing Units (GPGPU).

Bermudan Options

- Right to exercise at a set number of times: $t \in [t_0 = 0, \ldots, t_m, \ldots, t_M = T]$.
- d-dimensional underlying process: $\mathbf{S}_t = (S_t^1, \ldots, S_t^d) \in \mathbb{R}^d$.
- Intrinsic value of the option: $h_t := h(\mathbf{S}_t)$.
- The value of the option at the terminal time T:

$$V_T(\mathbf{S}_T) = \max(h(\mathbf{S}_T), 0).$$

- The conditional continuation value Q_{t_m}, i.e. the discounted expected payoff at time t_m:

$$Q_{t_m}(\mathbf{S}_{t_m}) = D_{t_m}\mathbb{E}\left[V_{t_{m+1}}(\mathbf{S}_{t_{m+1}})|\mathbf{S}_{t_m}\right].$$

- The Bermudan option value at time t_m and state $\mathbf{S}_{t_m}$:

$$V_{t_m}(\mathbf{S}_{t_m}) = \max(h(\mathbf{S}_{t_m}), Q_{t_m}(\mathbf{S}_{t_m})).$$

- Value of the option at the initial state $\mathbf{S}_{t_0}$, i.e. $V_{t_0}(\mathbf{S}_{t_0})$.

Bermudan options scheme

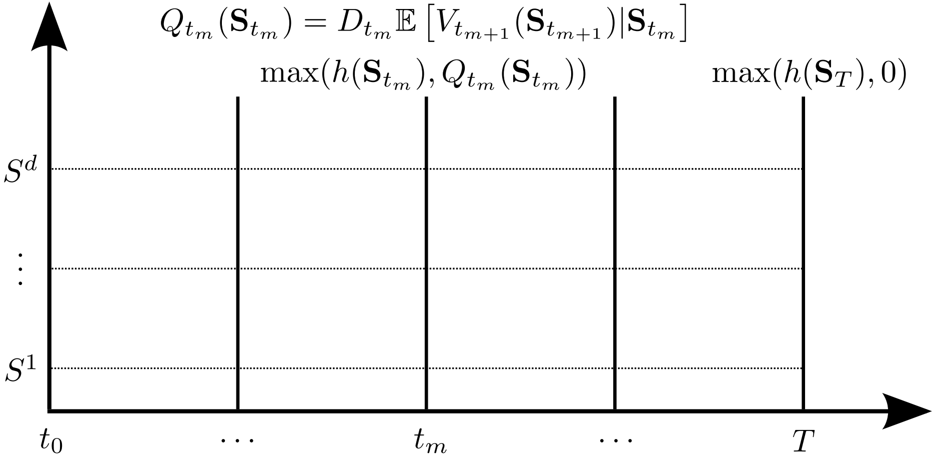

Figure: d-dimensional Bermudan option

Stochastic Grid Bundling Method

- Regression-based method.
- Forward in time: Monte Carlo simulation.
- Backward in time: Early-exercise policy by using dynamic programming.
- Step I: Generation of stochastic grid points

$$\{\mathbf{S}_{t_0}(n), \ldots, \mathbf{S}_{t_M}(n)\}, \ n = 1, \ldots, N.$$

- Step II: Option value at terminal time $t_M = T$

$$V_{t_M}(\mathbf{S}_{t_M}) = \max(h(\mathbf{S}_{t_M}), 0).$$

Stochastic Grid Bundling Method (II)

- Backward in time, t_m, $m \leq M$,:
- Step III: Bundling into ν non-overlapping sets or partitions

$$\mathcal{B}_{t_{m-1}}(1), \ldots, \mathcal{B}_{t_{m-1}}(\nu)$$

- Step IV: Mapping high-dimensional state space onto a low-dimensional space (least squares regression)

$$Z(\mathbf{S}_{t_m}, \alpha_{t_m}^{\beta}) \approx V_{t_m}(\mathbf{S}_{t_m}).$$

- Step V: Computing the continuation and option values at t_{m-1}

$$\widehat{Q}_{t_{m-1}}(\mathbf{S}_{t_{m-1}}(n)) = \mathbb{E}[Z(\mathbf{S}_{t_m}, \alpha_{t_m}^{\beta}) | \mathbf{S}_{t_{m-1}}(n)].$$

The option value is then given by:

$$\widehat{V}_{t_{m-1}}(\mathbf{S}_{t_{m-1}}(n)) = \max(h(\mathbf{S}_{t_{m-1}}(n)), \widehat{Q}_{t_{m-1}}(\mathbf{S}_{t_{m-1}}(n))).$$

Bundling

- Original: Iterative process (K-means clustering).
- Problems: Too expensive (time and memory) and distribution.
- New technique: Equal-partitioning.
- Two stages: sorting and splitting.

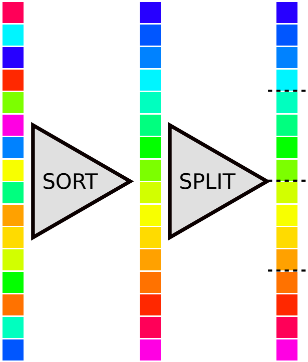

Figure: Equal partitioning scheme

Parallel SGBM on GPU

- First implementation: efficient C-version.
- NVIDIA CUDA platform.
- Two parallelization stages:
  - Forward: Monte Carlo simulation.
  - Backward: Bundles at each time step.
- Memory transfers to/from GPU: Unified Virtual Addressing (UVA).
  - Asynchronous transfers.
  - Page-locked memory accesses.

Parallel SGBM on GPU - Forward in time

- One GPU thread per Monte Carlo simulation.
- Random numbers “on the fly”: cuRAND library.
- Avoiding memory transfers and usage:
  - Compute the intrinsic value of the option.
  - Equal-partitioning: sorting criterion calculations.
- Original: memory transfers and usage cannot be avoided.

Monte Carlo implementation scheme

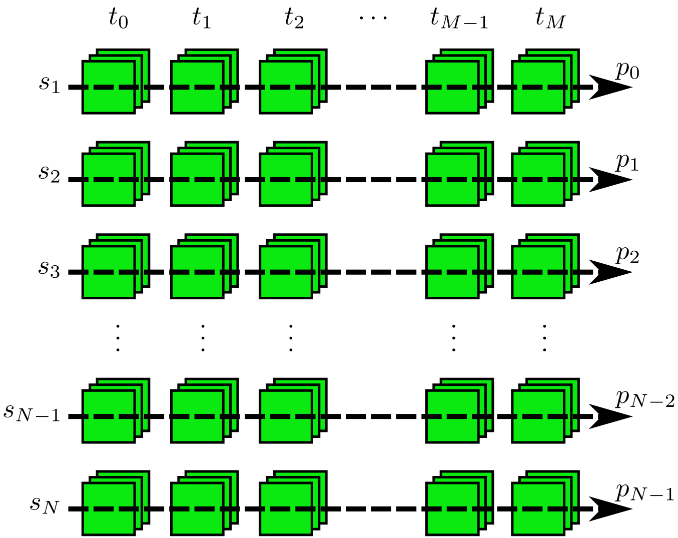

Figure: SGBM Monte Carlo

Parallel SGBM on GPU - Backward in time

- One GPU thread per bundle.
- Each bundle calculations (option value and regression) in parallel.
- All threads collaborate in order to compute the continuation value.
- Final reduction: Thrust library.

Backward implementation scheme

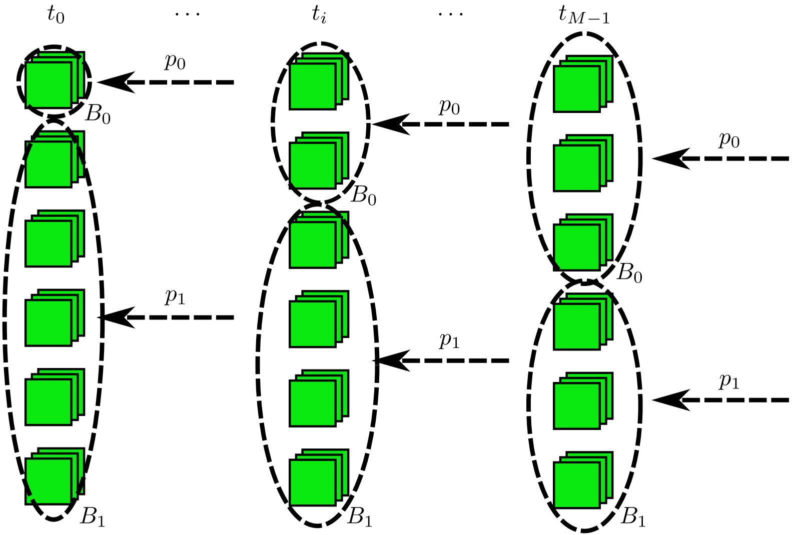

Figure: SGBM backward stage

Results

- Accelerator Island system of Cartesius Supercomputer.
  - Intel Xeon E5-2420 (Sandy Bridge).
  - NVIDIA Tesla K20m.
  - C-compiler: GCC 4.4.6.
  - CUDA version: 5.5.
- Geometric basket Bermudan put option: $\mathbf{S}_{t_0} = (40, \ldots, 40) \in \mathbb{R}^d$, $K = 40$, $r_t = 0.06$, $\sigma = (0.2, \ldots, 0.2) \in \mathbb{R}^d$, $\rho_{ij} = 0.25$, $T = 1$ and $M = 10$.
- Multi-dimensional Geometric Brownian Motion.
- Euler discretization, $\delta t = T/M$.

Results

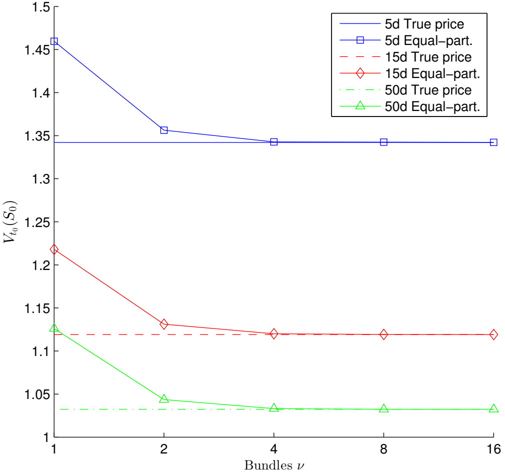

Figure: Convergence of SGBM with Equal-partitioning technique.

Results

Table: Time (s) for the C and CUDA versions. Test configuration: $N = 2^{22}$, $\delta t = T/M$ and $\nu = 2^{11}$.

	k-means			Equal-partitioning		
	$5d$	$10d$	$15d$	$5d$	$10d$	$15d$
C	676.25	1347.07	2008.16	157.83	234.60	320.59
CUDA	38.77	145.28	307.64	13.85	14.78	15.42
Speedup	17.44	9.27	6.52	11.40	15.87	20.79

Results

Table: Time (s) for a high-dimensional problem with equal-partitioning. Test configuration: $N = 2^{22}$ and $\delta t = T/M$.

	$\nu = 2^{12}$			$\nu = 2^{14}$		
	$30d$	$40d$	$50d$	$30d$	$40d$	$50d$
C	570.83	787.36	989.15	571.97	787.22	984.51
CUDA	18.06	21.44	25.09	19.25	22.60	26.06
Speedup	31.61	36.72	39.42	29.71	34.83	37.78

Conclusions

- Efficient parallel GPU implementation.
- Extend the SGBM's applicability: Increasing dimensionality and amount of bundles.
- New bundling technique.
- Future work:
  - American options.
  - CVA calculations.

References

CUDA webpage.
http://www.nvidia.com/object/cuda_home_new.html.

cuRAND webpage.
https://developer.nvidia.com/curand.

Shashi Jain and Cornelis W. Oosterlee.
The stochastic grid bundling method: Efficient pricing of bermudan options and their greeks, 2013.

Thrust webpage.
http://thrust.github.io/.

Questions and/or suggestions

Acknowledgements

This research was supported by the European Union in the FP7-PEOPLE-2012-ITN Program under Grant Agreement Number 304617 (FP7 Marie Curie Action, Project Multi-ITN STRIKE – Novel Methods in Computational Finance).

Thanks

Appendix

- Basis functions:

$$\phi_k(\mathbf{S}_{t_m}) = \left((\prod_{\delta=1}^{d} S_{t_m}^{\delta})^{\frac{1}{d}} \right)^{k-1}, \; k = 1, \ldots, 5,$$

- The expectation can directly be computed as:

$$\mathbb{E}\left[\phi_k(\mathbf{S}_{t_m})|\mathbf{S}_{t_{m-1}}(n)\right] = \left(P_{t_{m-1}}(n) e^{\left(\bar{\mu} + \frac{(k-1)\bar{\sigma}^2}{2}\right)\Delta t} \right)^{k-1},$$

where,

$$P_{t_{m-1}}(n) = \left(\prod_{\delta=1}^{d} S_{t_{m-1}}^{\delta}(n) \right)^{\frac{1}{d}}, \; \bar{\mu} = \frac{1}{d} \sum_{\delta=1}^{d} \left(r - q_\delta - \frac{\sigma_\delta^2}{2} \right), \; \bar{\sigma}^2 = \frac{1}{d^2} \sum_{p=1}^{d} \left(\sum_{q=1}^{d} C_{pq}^2 \right)^2$$