Black-Scholes option pricing

Victor Podlozhnyuk
vpodlozhnyuk@nvidia.com

April 2009

Document Change History

Version	Date	Responsible	Reason for Change
0.9	2007/03/19	Victor Podlozhnyuk	Initial release
1.0	2007/04/06	Mark Harris	Minor clarity / grammar edits for initial release
2.3	2009/04/12	Victor Podlozhnyuk	Adapted to OpenCL implementation

Abstract

The pricing of options is a very important problem encountered in financial engineering since the creation of organized option trading in 1973. This sample shows an implementation of the Black-Scholes model in OpenCL for European options.

NVIDIA Corporation
2701 San Tomas Expressway
Santa Clara, CA 95050
www.nvidia.com

Introduction

The most common definition of an *option* is an agreement between two parties, the *option seller* and the *option buyer*, whereby the option buyer is granted a right (but not an obligation), secured by the option seller, to carry out some operation (or *exercise* the option) at some moment in the future. The predetermined price is referred to as the *strike price*, and the future date is called the *expiration date*. (See Kolb & Pharr. [1])

Options come in several varieties:

A *call option* grants its holder the right to *buy* the *underlying asset* at a *strike price* at some moment in the future.

A *put option* gives its holder the right to *sell* the *underlying asset* at a *strike price* at some moment in the future.

There are several types of options, mostly depending on when the option can be exercised. European options can be exercised only on the expiration date. American-style options are more flexible as they may be exercised at any time up to and including expiration date and as such, they are generally priced at least as high as corresponding European options. Other types of options are path-dependent or have multiple exercise dates (Asian, Bermudian).

For a call option, the profit made at the exercise date is the difference between the price of the asset on that date and the strike price, minus the option price paid. For a put option, the profit made at the exercise date is the difference between the strike price and the price of the asset on that date, minus the option price paid.

The price of the asset at expiration date and the strike price therefore strongly influence how much one would be willing to pay for an option.

Other important factors in the price of an option are:

- **The time to the expiration date, *T*:** Longer periods imply a wider range of possible values for the underlying asset on the expiration date, and thus more uncertainty about the value of the option.
- **The riskless rate of return, *r*,** which is the annual interest rate of bonds or other “risk-free” investments: Any amount P of dollars is guaranteed to be worth $P \cdot e^{rT}$ dollars T years from now if placed today in one of theses investments or in other words, if an asset is worth P dollars T years from now, it is worth $P \cdot e^{-rT}$ today.

This example demonstrates an OpenCL implementation of the Black-Scholes model for European options.

Black-Scholes model.

The Black-Scholes model provides a partial differential equation (PDE) for the evolution of an option price under certain assumptions. For European options, a closed-form solution exists for this PDE. (See Black & Scholes, [2])

$$V_{call} = S \cdot CND(d_1) - X \cdot e^{-rT} \cdot CND(d_2)$$

$$V_{put} = X \cdot e^{-rT} \cdot CND(-d_2) - S \cdot CND(-d_1)$$

$$d_1 = \frac{\log(\frac{S}{X}) + (r + \frac{v^2}{2})T}{v\sqrt{T}}$$

$$d_2 = \frac{\log(\frac{S}{X}) + (r - \frac{v^2}{2})T}{v\sqrt{T}}$$

$$CND(-d) = 1 - CND(d)$$

where

V_{call} is the price for an option call,

V_{put} is the price for an option put,

$CND(d)$ is the Cumulative Normal Distribution function,

S is the current option price,

X is the strike price,

T is the time to expiration.

r is the continuously compounded risk free interest rate,

v is the implied volatility for the underlying stock,

The cumulative normal distribution function is computed with a polynomial approximation that provides six-decimal-place accuracy. The expansion uses a fifth-order polynomial. (See Hull, [3])

Implementation details

Choosing a data storage layout

The existence of a closed-form expression makes calculating option prices an easy task. The main problem is choosing the best data storage layout for the particular OpenCL device.

Here are some important features of OpenCL implemention on NVIDIA GPUs:

- Work-groups are executed as subgroups of logically coherent work-items, called *warps*. However unlike work-items in a warp, warps in a work-group are dynamically scheduled. Generally no assumption may be made on the exact order of warp execution within a work-group and synchronism across a work-group may only be guaranteed at *barriers* or *memory fences*. Warp size is 32 work-items on G8x / G9x / G10x NVIDIA GPUs
- No memory caching for global memory operations. In case global memory bandwidth is the bottleneck, the task of global memory bandwidth saving should be handled explicitly by the programmer. This is frequently achieved by utilizing fast local storage, which has about an order of magnitude higher bandwidth (loading/storing from/to local memory is generally as fast as reading/writing private register memory). Maximum possible local storage size for G8x / G9x / G10x NVIDIA GPUs is 16KB.
- Global memory accesses should be *coalesced* for best performance. For coalescing on G8x / G9x NVIDIA GPUs, loads and stores from each work-item of a *half-warp* (e.g. a subgroup of higher or lower 16 work-items of a warp) must be sequentially arranged and form a contiguous aligned block of memory of size 16 * <request size>, and request size should be 4, 8 or 16 bytes. It is important to understand that memory requests are independently formed for each half-warp, and coalescing happens or not happens across half-warp threads issuing the same load or store instruction (not different memory operations in the kernel). G10x NVIDIA GPUs relax the coalescing rules, significantly improving global memory bandwidth on many access patterns that are not coalescable on G8x / G9x GPUs. For more detailed description please see section 5.1.2.1 of CUDA Programming Guide
- As many RISC processors, NVIDIA GPUs are capable of loading and storing only aligned data elements of fixed sizes of 1, 2, 4, 8 or 16 bytes at instruction (work-item) level, so in the general case using arrays of structures requires either padding user structures to one of these "elementary" sizes (when structures are small enough), or issuing more than one load / store instructions per structure access (and possibly padding to a multiple of elementary size). The first case can be coalesced, but since padding bytes do not normally participate in any computations, it results in effective memory bandwidth loss (in addition to simply increased global memory consumption). The second case will never be coalesced, as load/store requests within a half-warp will never fall into adjacent global memory locations.
- Due to these complications, data is typically arranged as a set arrays of elementary types, which is known as the "structure of arrays"(SoA) strategy, since it makes global memory coalescing possible for any number of elementary-type fields (arrays)

Mapping data to work-items

A simple implementation of a kernel evaluating Black-Scholes formula would assign each work-item a single option with index equal *get_global_id(0)*, which implies a 1D global ID NDRange with exactly as many work-items as there are options to process. But there are some hardware constraints to be taken into account:

- Work-group ID NDRange can be either one- or two-dimensional and is limited by 65535 work-group IDs across each dimension.
- Work-item local ID NDRange can be one-, two- or three-dimensional and is limited by 512, 512, 64 local work-item IDs across dimension indices 0, 1, 2 respectively, with the total work-group size limit being 512 work-items.
- Depending on the utilization of local memory and private register memory, optimal work-group size typically varies in the range of 64..256 work-items.

Therefore, one-to-one correspondence between work-items and 1D addressing restricts the maximum input data size by around 33 millions options. So, in order to allow for arbitrary numbers of options and stick with convenient 1D indexing, each thread should process more than one index if required; which is implemented with the code in Listing 1.

```
for(
    unsigned int opt = get_global_id(0);
    opt < optN;
    opt += get_global_size(0)
)
    BlackScholesBody(
        &d_Call[opt],
        &d_Put[opt],
        d_S[opt],
        d_X[opt],
        d_T[opt],
        R,
        V
    );
```

Listing 1. Processing multiple options per work-item.

Bibliography

1. Craig Kolb and Matt Pharr (2005). "Option pricing on the GPU". *GPU Gems 2*. Chapter 45.
2. Fischer Black and Myron Scholes (1973). "The Pricing of Options and Corporate Liabilities". *Journal of Political Economy* **81** (3): 637-654.
3. John C. Hull (1997) “Options, Futures, and Other Derivatives”

Notice

ALL NVIDIA DESIGN SPECIFICATIONS, REFERENCE BOARDS, FILES, DRAWINGS, DIAGNOSTICS, LISTS, AND OTHER DOCUMENTS (TOGETHER AND SEPARATELY, "MATERIALS") ARE BEING PROVIDED "AS IS." NVIDIA MAKES NO WARRANTIES, EXPRESSED, IMPLIED, STATUTORY, OR OTHERWISE WITH RESPECT TO THE MATERIALS, AND EXPRESSLY DISCLAIMS ALL IMPLIED WARRANTIES OF NONINFRINGEMENT, MERCHANTABILITY, AND FITNESS FOR A PARTICULAR PURPOSE.

Information furnished is believed to be accurate and reliable. However, NVIDIA Corporation assumes no responsibility for the consequences of use of such information or for any infringement of patents or other rights of third parties that may result from its use. No license is granted by implication or otherwise under any patent or patent rights of NVIDIA Corporation. Specifications mentioned in this publication are subject to change without notice. This publication supersedes and replaces all information previously supplied. NVIDIA Corporation products are not authorized for use as critical components in life support devices or systems without express written approval of NVIDIA Corporation.

Trademarks

NVIDIA, the NVIDIA logo, GeForce, NVIDIA Quadro, and NVIDIA CUDA are trademarks or registered trademarks of NVIDIA Corporation in the United States and other countries. Other company and product names may be trademarks of the respective companies with which they are associated.

Copyright