AMD **Accelerated** Parallel Processing TECHNOLOGY

SAMPLE

Binomial Option Pricing - MultiGPU

1 Overview

1.1 Location

`$<APPSDKSamplesInstallPath>\samples\opencl\cl\`

1.2 How to Run

See the *Getting Started* guide for how to build samples. You first must compile the sample.

Use the command line to change to the directory where the executable is located. The pre-compiled sample executable is at `$<APPSDKSamplesInstallPath>\samples\opencl\bin\x86\` for 32-bit builds, and `$<APPSDKSamplesInstallPath>\samples\opencl\bin\x86_64\` for 64-bit builds.

Type the following command(s).

1. `BinomialOptionMultiGPU`
   Runs with default options, where x = 128.
2. `BinomialOptionMultiGPU -h`
   This prints the help file.

1.3 Command Line Options

Table 1 lists, and briefly describes, the command line options.

Table 1 Command Line Options

Short Form	Long Form	Description
-h	--help	Shows all command options and their respective meaning.
	--device	Devices on which the program is to be run. Acceptable values are cpu or gpu.
-q	--quiet	Quiet mode. Suppresses all text output.
-e	--verify	Verify results against reference implementation.
-t	--timing	Print timing.
	--dump	Dump binary image for all devices.
	--load	Load binary image and execute on device.
	--flags	Specify compiler flags to build the kernel.
-p	--platformId	Select the platformId to be used (0 to N-1, where N is the number of available platforms)
-d	--deviceId	Select deviceId to be used (0 to N-1, where N is the number of available devices).
-v	--version	AMD APP SDK version string.
-x	--samples	Number of samples to be calculated.
-i	--iterations	Number of iterations for kernel execution.

2 Introduction

Option pricing is a very important problem encountered in financial engineering. This sample shows an implementation of the Binomial Option pricing for European Options.

The most common definition of an *option* (see reference [1]) 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*; the future date is called the *expiration date*.

There are two basic options types:

- 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: they can be exercised any time up to, and including, the expiration date; 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 is 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; thus, there is 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 these investments. In other words, if an asset is worth P dollars T years from now, it is worth $P \cdot e^{-rT}$ today.

3 Binomial Option Pricing

The binomial option pricing model (see reference [2]) is an iterative solution that models the price evolution over the whole option validity period. For some types of options, such as the American options, using an iterative model is the only choice, since there is no known closed-form solution that predicts price over time.

The binomial model represents the price evolution of the option's underlying asset; this takes into account the binomial tree of all possible prices at equally-spaced time steps from today under the assumption that at each step the price can only move up and down at fixed rates and with respective pseudo-probabilities, P_u and P_d. The root node is the current price; each column of the tree represents all the possible prices at a given time; each node of value *S* has two child nodes of values *u * S* and *d * S*, where *u* and *d* are the factors of upward and downward movements for a single time-step, *dT*. Note that *u* and *d* are derived from volatility, *v*: $u = e^{(v * sqrt(dT))}$, $d = e^{(-v * sqrt(dT))}$; $(u * d = 1)$. P_d is simply equal to $(1 - P_u)$, and P_u is derived from the assumption that, over a period of *dT*, the underlying asset yields the same profit as a riskless investment on average. So, if it is worth *S* at time *t*, then it is worth $S * e^{(r * dT)}$ at time *(t + dT)*. This leads to the following equation:

Equation 1 $S * e^{(r * dT)} = (P_u * u * S + (1 - P_d) * d * S)$

From this we deduce P_u:

Equation 2 $P_u = (e^{(r * dT)} - d) / (u - d)$

From the binomial tree representation, we then iteratively derive the option price for each node of the tree, starting at the leaves. At each leaf of the tree (for example, at option expiry) deriving the call and put option price is simple:

> $V_{call} = max(S - X ,0)$: If the market price *S* at expiry date is greater than strike price *X*, a call option returns its holder *(S - X)* dollars of profit for a same-day sale transaction; otherwise, it returns zero profit.
>
> $V_{put} = max(X - S,0)$: If the market price *S* at expiry date is lower than strike price *X*, a put option gives its holder *(X - S)* dollars of profit; otherwise, it returns zero profit.

Having calculated all possible option prices at expiry date, we start moving back to the root, using the following formula: $V_t = ((P_u * Vu_{t+1}) + (Pd * Vd_{t+1})) * e^{(-r * dT)}$, where V_t is the option price for one of the nodes at time *t*, and Vu_{t+1} and Vd_{t+1} are the prices of its two child nodes. This formula is derived from the observation that an option that is worth V_t at time *t*, is worth at time *t + dT*, $V_t * e^{(r * dT)}$ on one hand, and $((Pu * Vu_{t+1}) + (Pd * Vd_{t+1}))$ on the other.

4 Implementation Details

Each block of work-items calculates call price from given stock price, strike price, *dT*, probabilities P_u and P_d. At first, all leaf node values are calculated by block of work-items and stored in shared memory. We begin at the leaf nodes of the tree, with indices in the range `[0, NUM_STEPS]`, corresponding to time step `NUM_STEPS`, where `NUM_STEPS` is the depth of tree. We reduce the price array step-by-step, moving back in time to the root of the tree (time step 0). We use two blocks of shared memory with the size of the `NUM_STEMPS` floats. In each step, one shared blocks acts as input, the other as output. In the next step, the input shared block becomes the output, and the output shared block becomes the input.

MultiGPU Implementation

A given stock price is independent of the other stock prices. Thus, we divide the samples (strike price values) into a number of blocks and distribute these samples (strike price values) to all the GPUs available, depending on the device compute capability.

The workload is distributed among the devices by calculating peak GFlops of each device.

Peak GFlops of a device = (Number of Compute Units *
Number of Stream Processors per Compute Unit *
Number of Processing Elements per stream processor *
Max Clock Frequency *
Number of floating Point operations per cycle per processing element) / 1000.

We distribute the samples (strike price values) among the devices by using the following formula.

Number of samples to be executed on a specific device = Ratio of the device *
Total number of samples.

Where Ratio of the device = Peak GFlops of the device / (Total GFlops of all the devices).

5 Recommended Input Option Settings

For best performance, enter the following on the command line: `-x 1048576 -i 10 -q -t`

6 References

1. http://en.wikipedia.org/wiki/Option_(finance)
2. Simon Benninga, Zvi Wiener, *The Binomial Option Pricing Model*.

Contact

Advanced Micro Devices, Inc.
One AMD Place
P.O. Box 3453
Sunnyvale, CA, 94088-3453
Phone: +1.408.749.4000

For AMD Accelerated Parallel Processing:
URL: developer.amd.com/appsdk
Developing: developer.amd.com/
Support: developer.amd.com/appsdksupport
Forum: developer.amd.com/openclforum