GPU チャレンジ 2010 自由課題部門・特別賞応募レポート

二田　晴彦†

1. はじめに

このレポートでは，GPU チャレンジ 2010 自由課題部門・グリッド協議会金融分科会特別賞に応募するプログラムについて述べ，またプログラムの計算速度を評価する．

2. 計算の概要[1]

GPU チャレンジ 2010 特別賞の課題は，モンテカルロ法を用いてコールオプションの価値を計算することである．計算に使われる確率過程(リスク中立世界における)は原資産を S として

$$dS = rSds + vSdz$$

である．ここでは，r はリスク中立世界での期待収益率，v はボラティリティ，dz はウィナー過程である．S のパスを発生されるために満期までの時刻 T を長さ dt の区間に N 分割する．シミュレーションでは，1 回の試行で正規分布に従う N 個の乱数を用いて S の完全なパスを作成する．さらに S に対してシミュレーションを行うよりも，$\ln S$ に対してシミュレーションを行った方が正確な値が得られる．その場合パス生成の式は，伊藤の補題より，

$$\ln S(t+dt) - \ln S(t) = (r - v^2/2)dt + v\,\mathrm{sqrt}(dt)\,e$$

となる．ここで e は平均 0，分散 1 の正規乱数である．満期 T のコールオプションの現在価値 c は，

$$c = \exp(-rT)\max(0, S(T) - k)$$

となる．ここで k は行使価格である．これを mc 回だけ繰り返して平均を取ることで，コールオプションの現在価値の期待値が計算できる．

3. プログラムの使用方法

コンパイルするためには，ソースコードを展開したディレクトリに移動し，

```
$ make
```

を実行する．コンパイルが成功すると，同じディレクトリに mc_main と言うファイルが生成される．計算を行うには

```
$ ./mc_main
```

を実行する．これでサンプルプログラムと同じパラメータで 10 億試行のモンテカルロ法が計算される．パラメータを変更して実行したい場合，

```
$ ./mc_main 1.0 100.0 0.05 0.1 95.0 100000 10
```

のように，T，$S(0)$，r，v，k，mc，n をコマンドラインから順に入力して実行する．

4. 最適化手法

サンプルプルグラムでは，1 つのメルセンヌツイスター乱数生成器を使い計算しているが，GPU での計算では，GPU 上の各スレッドがそれぞれ個別にメルセンヌツイスター乱数生成器を持ち，独立に試行を繰り返し，計算を行う．最後に各スレッドがグローバルメモリに結果を転送する．GPU での処理が終わった後，CPU はグローバルメモリに置かれた各スレッドの結果を取り出し，それを総和し最終的な結果を得る．

次に各部の最適化手法について述べる．一様乱数の生成と正規乱数への変換が計算時間の大部分を占めるため，今回はこの 2 つについて特に最適化を行う．

一様乱数の生成部(メルセンヌツイスター)

各スレッドが独立にかつ相関の低い乱数列を生成するために，Dynamic Creator[2]を用いる．各スレッドには，異なる乱数生成のパラメータが割り当てられ，独立に乱数が生成される．また，状態変数は通常の方法では，ローカルメモリに配置され低速なアクセスになるため，スレッド間で共有する必要はないが，高速なアクセスを達成するためシェアードメモリに置く．

一様乱数から正規乱数への変換部(Moro の方法)

一様乱数から正規乱数への変換に用いられる Moro の方法は，次の 3 つの点で GPU での計算に不向きである．

1. 大きな条件分岐があり，ワープダイバージェント(ワープが分岐の両方を実行し，性能が低下すること)が発生する．

† 新潟大学大学院自然科学研究科
Graduate School of Science and Technology, Niigata University

2. 対数計算が必要である．
3. 倍精度浮動小数点数の除算が必要である．

これらの欠点を抑えるために，1.に対しては，多項式計算の部分の分岐を結合させ，ワープダイバージェントを低減している．

2.に関しては，仮にこれが単精度浮動小数点数であれば，プログラム中で intrinsic 関数を呼ぶことで，SFU(Super Function Unit)を使い対数の計算を高速に行うことができる．しかし，倍精度浮動小数点数においては，SFU は利用できず，倍精度の標準数学関数はそれほど速くないという問題がある．そのため今回は，21 次の多項式近似を用いる．対数関数の多項式近似は，

$$\log(1+x) = c_0x + c_1x^2 + ... + c_{20}x^{21}$$

と表される．ここで，c_0 から c_{20} は定数である．多項式の計算は，Horner 法を使い乗算回数を削減する．数値実験の結果から，この 21 次の多項式近似による対数計算は，標準の数学関数と同等の結果が得られることを確認した．

最後に 3.に関しても単精度浮動小数点数であれば，SFU を使い高速に計算できるが，倍精度浮動小数点数には，SFU は利用できない．倍精度浮動小数点数で高速に除算を計算するために，今回は除算 a / b の処理を，b の逆数を計算し，それを a に掛ける処理 $a * (b^{-1})$に変更する．逆数の計算には，まず SFU を使い単精度浮動小数点数で荒く逆数を計算し，それを 3 次の Newton 法を一回適用し，倍精度浮動小数点数の精度を得る手法を用いる．数値実験の結果から，この手法は除算演算子を使う方法を同等の結果が得られることを確認した．

5. 実行結果

CPU で計算させた場合と，GPU で計算させた場合の速度と計算結果を比較する．CPU の計算環境は，

・OS：Linux x86_64 2.6.28-18-generic
・CPU：Core2 Duo E7300 2.66GHz
・コンパイラ：gcc version 4.3.3

である．また GPU の計算環境は，

・OS：Linux x86_64 2.6.18-128.el5
・CPU：Core i7 920 2.67GHz
・GPU：Tesla C1060
・コンパイラ：nvcc release 2.3, V0.2.1221

である．計算に用いたパラメータは，$T = 1$，$S(0) = 100$，$r = 0.05$，$v = 0.1$，$k = 95$，$mc = 1{,}000{,}000{,}$，$n = 10$ である．また，Dynamic Creator で生成したメルセンヌツイスターの乱数周期は 2^{521}-1 であり，GPU の総スレッド数は，ブロックあたりのスレッド数 64，ブロック数 512 で，64×512=32,768 である．表 1 に実行結果を示す．計算結果(コールオプションの現在価値)は，CPU による計算で 10.40545，GPU で 10.40523 となっている．使用した乱数生成手法は，どちらもメルセンヌツイスターであるが，乱数のパラメータが異なり，従い乱数系列も異なるために結果は完全には一致していない．しかし，標準誤差を考慮すると，どちらも同じ結果に収束していると言える．計算時間は，CPU に対し GPU が 23 倍程度高速化できている．

表 1 実行結果

計算環境	計算結果	結果の標準誤差	計算時間［秒］
CPU	10.40545	0.00028	549.6
GPU	10.40523	0.00028	23.3

6. まとめ

GPU チャレンジ 2010 自由課題部門・グリッド協議会金融分科会特別賞に応募するプログラムについて述べた．最適化は，乱数生成部分を中心に行い，CPU と比較した場合，23 倍の高速化を達成した．

参 考 文 献

[1] ジョン ハル，フィナンシャルエンジニアリング―デリバティブ取引とリスク管理の総体系 第 7 版，金融財政事情研究会，2009

[2] Makoto Matsumoto and Takuji Nishimura, “Dynamic Creation of Pseudorandom Number Generators”, Monte Carlo and Quasi-Monte Carlo Methods 1998, Springer, pp. 56-69, 2000