Санкт-Петербургский государственный университет

Кафедра компьютерного моделирования и многопроцессорных систем

Попова Евгения Николаевна

Выпускная квалификационная работа бакалавра

Численный анализ структур в финансовой математике

Направление 010300

Фундаментальная информатика и информационные технологии

Научный руководитель,

доктор физ.-мат. наук,

профессор

Богданов А. В.

Санкт-Петербург

2016

Содержание

Введение

Ценообразование опционов является одной из важных задач в финансовой математике. В настоящее время объем вычислений в различных задачах становится все больше, и архитектура вычислительных систем постоянно меняется, поэтому эффективная реализация моделей ценообразования опционов на современных устройствах становится все более важной проблемой. Более того, огромное количество задач со сложными математическими моделями не могут быть решены аналитически. В таких случаях используются численные методы, которые требуют высокой вычислительной мощности. Чтобы получить достаточно точные результаты, временные затраты могут быть большими, поэтому ускорение на графических процессорах является эффективным способом решения задач с помощью численных методов.

Технология CUDA, разработанная NVIDIA, является архитектурой параллельных вычислений общего назначения, которая использует графические процессоры для решения сложных задач. Архитектура CUDA позволяет разработчикам использовать языки C и C++, которые являются наиболее широко используемыми языками программирования высокого уровня.

В финансовой сфере время играет огромную роль, так как любая задержка в обработке информации может привести к экономическим потерям. Поэтому использование параллельных вычислений при построении модели ценообразования опционов является оправданным.

Огромную роль в открытии методов вычисления цены европейских опционов сыграла работа [1]. Формула Блэка-Шоулза является одним из основных вычислительных инструментов, который измеряет и прибыль, и убытки, и риск опционной сделки для инвесторов. Для уравнения Блэка-

Шоулза можно найти точное аналитическое решение, однако для обобщенной модели в некоторых случаях (напр. в случае азиатского опциона) точное решение найти невозможно. Эта проблема приводит к использованию численных методов, которые в свою очередь приводят к приближенному значению цены опциона.

Часто используют методы Монте-Карло [2][3], которые основаны на получении большого числа стохастических процессов. В настоящее время финансовые расчеты часто основываются на этих методах из-за присущей им высокой степени параллелизма.

Таким образом, исследование оценки опционов с помощью методов Монте-Карло делают данную работу актуальной.

Постановка задачи

Для дальнейшего понимания работы стоит дать следующие определения [4]:

Опр. 1. Опцион – это контракт, согласно которому покупатель опциона получает право (не обязательство) купить или продать базовый актив по заранее оговоренной цене, которая называется ценой «страйк» или ценой исполнения, в определенный момент в будущем или на протяжении определенного отрезка времени. В то же время, продавец опциона обязан совершить сделку по продаже или купле базового актива в соответствии с условиями проданного опциона.

Опр. 2. «Колл»-опцион – опцион, предоставляющий покупателю право совершить покупку базового актива по цене «страйк».

Опр. 3. «Пут»-опцион – опцион, предоставляющий покупателю право продать базовый актив по цене «страйк».

Условия исполнения опциона (продажи или покупки актива) принимается продавцом опциона, а покупатель должен соблюдать эти условия. Следовательно, корректное значение опциона имеет решающее значение для совершения сделки для обеих сторон.

Введем следующие обозначения:

$C(S,t)$ – цена опциона, зависящая от времени и цены базового актива,

$S(S_0)$ – цена базового актива (начальная цена базового актива),

σ – волатильность базового актива, статистический финансовый показатель, который характеризует изменчивость цены базового актива,

r – безрисковая процентная ставка,

$A(S,t)$ – средняя цена опциона за определенный временной промежуток, зависящая от времени и цены базового актива,

K – цена-«страйк» опциона,

T – время экспирации (момент или временной отрезок, в который может быть выполнен опцион),

t – текущее время.

Существуют различные виды опционов [5]. Наиболее популярными являются американский, европейский и некоторые виды экзотических опционов (азиатский, бинарный, барьерный и др.). Рассмотрим некоторые из них подробнее:

1. Европейский опцион может быть исполнен только в момент экспирации. Тогда в случае колл-опциона, если цена базового актива S к моменту истечения контракта будет выше, чем цена исполнения K, тогда покупатель, при совершении сделки, получит прибыль, равную $S_T - K$. Если же цена базового актива в момент экспирации будет ниже цены исполнения, то покупатель не получит никакой прибыли, так как опцион будет не выгодно исполнять.

Тогда в этом случае справедливая цена опциона, также называемая премией, которая выплачивается продавцу, может быть вычислена следующим образом:

В случае европейского колл-опциона:

$$C(T) = \max(S_T - K, 0).$$

В случае европейского пут-опциона:

$$C(T) = \max(K - S_T, 0).$$

2. Азиатский опцион – это опцион, цена исполнения которого основывается на средней стоимости базового актива за определенный временной промежуток. Преимуществом этого опциона является то, что продавцу намного сложнее управлять условиями контракта, чтобы уменьшить прибыль покупателя, так как она будет зависеть только от цены базового актива в моменты мониторинга и не будет фиксированной и заранее известной в момент покупки опциона, как в случае с европейским типом.

Тогда справедливая цена азиатского опциона может быть вычислена следующим образом:

1) Если мониторинг цены базового актива является непрерывным:

- Для колл-опциона

$$C(T) = \max(\frac{1}{T}\int_0^T \omega(t)dt - K\ ,0).$$

- Для пут-опциона

$$C(T) = \max(K - \frac{1}{T}\int_0^T \omega(t)dt\ ,0).$$

2) Если мониторинг цены базового актива является дискретным:

- Для колл-опциона

$$C(T) = \max(\frac{1}{T}\sum_{i=0}^{n} \omega(t_i) - K\ ,0).$$

- Для пут-опциона

$$C(T) = \max\left(K - \frac{1}{T}\sum_{i=0}^{n} \omega(t_i)\ ,0\right).$$

Существуют различные модели ценообразования опционов:

- Модель Блэка-Шоулза
- Биномиальная модель
- Модель Хестона
- Модель Монте-Карло
- Модель Кокса-Рубинштейна
- Другие

В этой работе будет использован метод Монте-Карло, а алгоритмы нахождения цены будут реализованы с помощью:

- стохастического дифференциального уравнения,
- континуального интеграла.

Метод Монте-Карло основан на получении большого числа стохастических процессов. Этому методу присуща высокая степень параллелизма.

Поэтому целью этой работы является построить модель ценообразования опционов, используя гетерогенные вычислительные системы, и за счет этого снизить вычислительные издержки методов оценки опционов.

Чтобы достичь положительного результата, необходимо выполнить следующие задачи:

1. Исследовать алгоритмы методов оценки опционов;

2. Реализовать последовательный и параллельный алгоритмы, используя CUDA;

3. Провести численные эксперименты и проанализировать их результаты;

Обзор литературы

В источниках [1], [4], [5] описаны основы финансовой математики, в том числе связанные с опционами. Рассмотрены различные модели ценообразования и фондовый рынок в целом.

В работах [2], [3] представлены описания методов Монте-Карло, в [6] методы для стохастического дифференциального уравнения для азиатских и европейских опционов.

В статьях [7], [8] исследованы многомерные интегралы, используемые при ценообразовании опционов, которые находят применение в вычислении цены азиатских опционов.

Работы [11], [12], [14] помогают в понимании архитектуры вычислительных систем и параллельных вычислений и в освоении технологии CUDA.

Источники [13], [15], [16] являются официальными и предоставлены компанией NVIDIA.

Глава 1. Модель Блэка-Шоулза.

Перед тем как решить любое уравнение, стоит попытаться решить его аналитически. И только в том случае, если аналитического решения не существует или его решение является трудоемкой задачей, используются численные методы, которые сводятся к выполнению конечного числа действий над числами. В таком случае полученный результат имеет определенную погрешность, в то время как аналитическое решение является точным.

Уравнение Блэка-Шоулза позволяет получить аналитическое решение. Впервые модель Блэка-Шоулза для оценки опционов была опубликована в статье [1] в 1973 году. Эта формула была открыта экономистами Фишером Блэком, Мирном Шоулзом и Робертом Мертоном. Данную модель используют и в настоящее время.

Ключевым элементом при оценке опциона с помощью этой модели является ожидаемая волатильность базового актива. Так как возрастающая или убывающая цена на него влияет прямо пропорционально на цену опциона.

Однако для использования данной модели должны быть выполнены следующие условия:

- В течение всего срока действия опциона не выплачиваются дивиденды по базовому активу.
- Опцион являются европейским, то есть он может быть исполнен только в момент экспирации.
- Финансовые рынки являются эффективными.
- Комиссий и транзакционных издержек нет.
- Безрисковая процентная ставка и волатильность являются постоянными и заранее известными.

- Модель основана на логнормальном распределении базового актива.

Тогда формула Блэка – Шоулза выглядит следующим образом:

$$C(S,t) = SN(d_1) - Ke^{-r(T-t)}N(d_2)$$

$$d_1 = \frac{\ln\left(\frac{S}{K}\right) + \left(r + \frac{\sigma^2}{2}\right)(T-t)}{\sigma\sqrt{T-t}} \quad (1)$$

$$d_2 = d_1 - \sigma\sqrt{T-t}$$

$SN(d_1)$ характеризует ожидаемую прибыль от покупки базового актива. $Ke^{-r(T-t)}N(d_2)$ определяет размер цены исполнения в момент экспирации.

Необходимо заметить, что допущения, изложенные выше, соответствует идеальному рынку, а значит, их полное выполнение не является реальным. Однако именно эта модель послужила началом для дальнейшего исследования опционов на фондовом рынке.

Глава 2. Методы Монте-Карло

2. 1. Стохастическое дифференциальное уравнение

Опр. 4. Стохастическое дифференциальное уравнение (СДУ) — дифференциальное уравнение, в котором один член или более представляют собой стохастический процесс (случайный процесс).

Соответствующее модели Блэка-Шоулза (1) стохастическое дифференциальное уравнение выглядит следующим образом [5]:

$$dS(t) = rS(t)dt + \sigma dB(t),$$

где $B(t)$ – стандартный винеровский процесс, $B(t) \sim \mathcal{N}(0, 1)$.

Тогда цену базового актива в момент экспирации можно вычислить по следующей формуле:

$$S(T) = S_0 e^{\left(r - \frac{\sigma^2}{2}\right)T + \sigma\sqrt{T}\varepsilon} \qquad (1)$$

где ε – случайная величина, распределенная по нормальному закону.

2. 1. 1. Стохастическое дифференциальное уравнение для европейского опциона

Для европейских опционов метод Монте-Карло можно описать следующим алгоритмом [6]:

1. Сгенерировать n случайных чисел для ε;
2. Для каждого значения ε, вычислить цену базового актива $S(T)$, по формуле (2)
3. Найти среднее значение цены базового актива $S_{\text{ср}}$
4. Найти цену опциона по следующей формуле:

$$C = e^{-rT} S_{\text{ср}}$$

2. 1. 2. Стохастическое дифференциальное уравнение для азиатского опциона

Сложность применения метода Монте-Карло к ценообразованию азиатских опционов заключается в том, что необходимо в каждый момент наблюдения генерировать случайный временной ряд для цены базового актива, в отличие от европейских опционов, где было достаточно сгенерировать случайный ряд для цены базового актива только в момент экспирации.

Для начала, разобьем время мониторинга на m отрезков, то есть шаг $\Delta t = \frac{T}{m}$:

$$t_i = \frac{iT}{m}, \qquad i = 0, 1, \dots, m$$

1. генерируем m случайных значений ε, и каждому числу сопоставляем значение цены базового актива по следующей формуле:

$$S(t_{i+1}) = S(t_i) e^{\left(r - \frac{\sigma^2}{2}\right)\Delta t + \sigma\sqrt{\Delta t}\varepsilon}, \qquad i = 0, 1, \dots, m$$

2. Находим среднее арифметическое значение полученного ряда:

$$\bar{S} = \frac{1}{m}\sum_{i=1}^{m} S(t_i)$$

3. Повторяем 1 и 2 шаги n раз. Получаем n значений цен базового актива.

4. Вычисляем цену опциона:

$$C(S_0, T) = \frac{1}{n} e^{-rT} \sum_{j=1}^{n} S_j$$

2. 2. Континуальный интеграл

Опр. 4. Континуальный интеграл/функциональный интеграл – результат функционального интегрирования, то есть вычисление интеграла некоторого функционала Ф по пространству функций $x(t)$.

2. 2. 1. Континуальный интеграл для европейского опциона

Уравнение Блэка-Шоулза в частных производных имеет вид:

$$\frac{\partial C}{\partial t} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 C}{\partial S^2} + rS\frac{\partial C}{\partial S} - rV = 0$$

Сделаем следующую замену:

$$x = \ln S$$

Тогда оно примет следующий вид [7]:

$$\frac{\partial C}{\partial t} + \frac{1}{2}\sigma^2 \frac{\partial^2 C}{\partial x^2} + \mu\frac{\partial C}{\partial x} - rC = 0$$

$$\mu = r - \frac{\sigma^2}{2}$$

$$C(e^{X_T}, T) = F(S_T) = \max(e^{X_T} - K, 0)$$

Для решения этой задачи можно использовать формулу Фейнмана-Каца:

$$C(S,t) = e^{-rT}E[F(S_T)],$$

$E[F(S_T)]$ – мат. ожидание, которое можно представить также в виде:

$$E[F(S_T)] = \int_{-\infty}^{+\infty} \max\{e^{x_T} - K, 0\} p(x_T, T|x,t)dx_T, \qquad (3)$$

где $p(\cdot)$ – плотность вероятности перехода из (x,t) в (x_T, T).

$$p(x_T,\, T|x,t) =$$

$$= \int_{-\infty}^{+\infty} \cdots \int_{-\infty}^{+\infty} dx_1 \cdots dx_n \frac{1}{\sqrt{(2\pi\sigma^2\Delta t)^{n+1}}} e^{\left(-\frac{1}{2\sigma^2\Delta t}\sum_{k=1}^{n+1}(x_k-(x_{k-1}+\mu\Delta t))^2\right.}$$

Выражение (3) можно представить в виде [8]:

$$E[F(S_T)] = \int_{-\infty}^{+\infty} \cdots\cdots \int_{-\infty}^{+\infty} dx_1 \cdots dx_{n+1} \max\{e^{x_{n+1}} - K, 0\} \cdot$$

$$\frac{1}{\sqrt{(2\pi\sigma^2\Delta t)^{\frac{n+1}{2}}}} e^{\left(-\frac{1}{2\sigma^2\Delta t}\sum_{k=1}^{n+1}(x_k-(x_{k-1}+\mu\Delta t))^2\right.}$$

Тогда алгоритм вычисления цены европейского опциона методом Монте-Карло с помощью континуального интеграла будет выглядеть следующим образом:

1. Вычислить $x_0 = \ln S_0$.
2. Сгенерировать значения $x_1, x_2, \dots, x_{n+1}$ с помощью значения x_0 и генеральной совокупности с плотностью:

$$\frac{1}{\sqrt{2\pi\sigma^2\Delta t}} e^{\left(-\frac{1}{2\sigma^2\Delta t}(x_i-(x_{i-1}+\mu\Delta t))^2\right.}$$

3. Вычислить $\max\{e^{x_{n+1}} - K, 0\}$.
4. Повторить 1 – 3 шаги N раз.
5. Полученные на третьем шаге значения обозначить как $a_i, i = 1, \dots, N$.
6. Найти значение цены опциона:

$$C(S_0, T) = \frac{e^{-rT}}{N} \sum_{i=1}^{N} a_i$$

7. Повторять все шаги до тех пор, пока значение, полученное на текущем шаге, $C(S_0, T)$ не будет отличаться от значения, полученного на предыдущем шаге, на достаточно малую величину.

2. 2. 2. Континуальный интеграл для азиатского опциона

Цена азиатского колл-опциона равна следующему выражению:

$$F(S, A) = \max(A(0, T) - K\ ,0).$$

Согласно материалам из [] это значение можно также вычислить с помощью следующей формулы:

$$C(S_T, T) = e^{-rT} E[F(e^{x_T}, A(0, T))],$$

$$E[F(e^{x_T}, I)] = \int_{-\infty}^{+\infty} \cdots \cdots \int_{-\infty}^{+\infty} dx_1 \cdots dx_N \max\{A(0, T) - K, 0\} \cdot$$

$$\frac{1}{\sqrt{(2\pi\sigma^2\Delta t)^N}} e^{(-\frac{1}{2\sigma^2\Delta t}\sum_{k=1}^{N}(x_k - (x_{k-1} + \mu\Delta t))^2}$$

$$A(0, T) = \frac{\Delta t}{T} \int_0^1 \omega(\tau) e^{(x_n - x_{n-1})\tau + x_{n-1}} (1 + \frac{\sigma^2\Delta t(1-\tau)\tau}{2}) d\tau$$

Тогда алгоритм вычисления цены азиатского опциона методом Монте-Карло с помощью континуального интеграла будет выглядеть следующим образом:

1. Вычислить $x_0 = \ln S_0\ , \mu = r - \frac{\sigma^2}{2}$.
2. Сгенерировать значения $x_1, x_2, \ldots, x_{n+1}$ со следующими плотностями для x_i:

$$\frac{1}{\sqrt{2\pi\sigma^2\Delta t)}}e^{(-\frac{1}{2\sigma^2\Delta t}(x_i-(x_{i-1}+\mu\Delta t))^2}$$

3. Имея значения $x_0, x_1, \dots, x_{n+1}$, вычислить

$$A(0,T) = \frac{\Delta t}{T}\sum_{k=1}^{n+1}\int_0^1 \omega(\tau)e^{(x_k-x_{k-1})\tau+x_{k-1}}(1+\frac{\sigma^2\Delta t(1-\tau)\tau}{2})d\tau$$

4. Повторить 1 – 3 шаги N раз.

5. С помощью полученных на 4 шаге значений вычислить:

$$\bar{C_j} = \max(A(0,T)-K,0)\,, j = 1,\dots,N$$

6. Найти значение цены опциона:

$$C(S_0,T) = \frac{e^{-rT}}{N}\sum_{i=1}^{N}\bar{C_j}$$

7. Повторять все шаги до тех пор, пока значение, полученное на текущем шаге, $C(S_0,T)$ не будет отличаться от значения, полученного на предыдущем шаге, на достаточно малую величину.

Глава 3. Обзор вычислительных систем

Под вычислительной системой обычно понимают совокупность аппаратного и программного обеспечения, которая предназначена для решения задач пользователя. При этом система имеет вычислители, с помощью которых можно осуществлять параллельную обработку.

В сфере суперкомпьютерных технологий задача повышения производительности систем всегда оставалась важной. В процессе развития этих систем, принимались различные решения для достижения этой цели.

В 60-е годы появились интегральные микросхемы, что привело к скачку производительности за счет принципов конвейеризации и суперскалярности. На их основе были созданы микропроцессоры (CPU), а позже были созданы графические ускорители (GPU). [9]

Большое количество параллельных вычислений и сложность масштабирования многопроцессорных и многоядерных систем привели к использованию GPU не только для ускорения трехмерной графики, но и для решения задач, которые обладают высокой степенью параллелизма. Позже это привело к появлению вычислительной техники GPGPU. [14]

GPU состоит из однородных вычислительных элементов с общей памятью, каждый из которых может исполнять тысячи потоков. Эти потоки могут быть сгруппированы в блоки, которые имеют общий кэш и быструю разделяемую память.

Так как современные GPU имеют высокую скорость доступа к модулям памяти, обработка больших массивов данных может происходить параллельно, при этом производительность достигает высоких значений.

Если изначально CPU и GPU были созданы для определенного класса задач, а системы являлись гомогенными, то есть состояли из одного или

нескольких вычислителей одинаковой архитектуры, то сейчас часто используют гетерогенные системы. Гетерогенные системы могут использовать универсальный процессор CPU и графический GPU совместно.

Стандартной гетерогенной системой является совокупность одного CPU и одного или более GPU. Однако GPU используется как сопроцессор к центральному процессору, который является «хостом», и называется «устройством».

GPGPU (General Purpose computing for GPU) – это техника использования GPU для расчетов, которые обычно выполняются на CPU. [10]

Существуют различные платформы для GPGPU. Наиболее популярными являются:

- OpenCL
- Nvidia CUDA
- DirectCompute
- AMD FireStream
- И др.

3. 1. Обзор Nvidia CUDA

Компания Nvidia разработала программно-аппаратную архитектуру параллельных вычислений CUDA (Compute Unified Device Architecture), которая позволяет реализовать неграфические вычисления на графических процессорах. [11][12][13][15]

CUDA-программа использует и CPU, и GPU. На CPU выполняется последовательная часть кода. На GPU - параллельные участки кода, выполняемые одновременно несколькими потоками.

Платформа CUDA предоставляет набор расширений для языков C/C++.

CUDA позволяет по собственному усмотрению разработчика управлять доступом к набору инструкций и памяти.

С помощью CUDA можно определить так называемые ядра (kernel) – функции, выполняющиеся на графическом процессоре. Ядро выполняется такое же количество раз, сколько запускается потоков на GPU. То есть поток выполняет ядро.

При этом ядро может быть вызвано из кода, выполняющегося на CPU, но даже после вызова, код продолжит выполняться. Поэтому нужно производить синхронизацию.

В соответствии с CUDA ядро выполняется на так называемом гриде, который состоит из блоков, которые в свою очередь состоят из нитей (потоков). Разделение на блоки необходимо для эффективного взаимодействия нитей между собой в одном блоке, что дает некоторые преимущества для параллельных алгоритмов.

CUDA имеет несколько видов памяти:

1. Регистровая память. Регистры распределяются по нитям блока на этапе компиляции и являются доступными им для чтения и записи на все время исполнения ядра.
2. Разделяемая память. Размещена на потоковом мультипроцессоре. Распределяется по блокам, предоставляя каждому одно и то же количество разделяемой памяти. Является самой быстрой, но в то же время ограниченной.
3. Локальная память. Размещена в DRAM (динамической памяти с произвольным доступом). Используется нитями для хранения локальных данных.
4. Глобальная память. Обычная память DRAM. Расположена на GPU. Является самой медленной из всех типов памяти CUDA. Выделяется на грид, то есть любая нить может пользоваться ей для

чтения и записи. Обычно используется для хранения данных большого объема.

5. Константная и текстурная память. Размещены в DRAM. Обладают независимым кэшем. Имеют высокую скорость доступа. Доступна всем нитям GPU для чтения. CPU может проводить как запись, так и чтение.

Глава 4. Реализация с помощью CUDA

Для того чтобы использование GPU было эффективным, необходимо использовать некоторые оптимизированные под него параллельные алгоритмы. Рассмотрим два алгоритма обработки массивов.

4. 1. Алгоритм параллельной редукции

Опр. 4. Редукцией массива $x_0, x_1, x_2, \dots, x_{n-1}$ называется следующее выражение:

$$A = (((x_0 + x_1) + x_2 + \cdots + x_{n-1})$$

Реализация последовательной редукции тривиальна.

Алгоритм параллельной редукции:

1. Используется N потоков.
2. Каждым потоком вычисляются суммы двух элементов, которые записываются в разделяемую память.
3. Второй шаг повторяется $logN$ раз, при этом уменьшая число активных потоков в 2 раза на каждой итерации, также необходима синхронизация после каждого шага.

На рисунке 1 на примере последовательности из 16 чисел наглядно показан данный алгоритм. Подробное описание алгоритма можно найти в данной книге [11].

4. 2. Алгоритм префиксных сумм

Рассмотрим алгоритм префиксных сумм.

Опр. 5. Префиксная сумма (prefix sum, scan) последовательности чисел $x_0, x_1, x_2, \ldots$ - это последовательность чисел $y_0, y_1, y_2, \ldots$, которые вычисляются следующим образом:

$$y_0 = x_0$$

$$y_1 = x_0 + x_1$$

$$y_2 = x_0 + x_1 + x_2$$

…

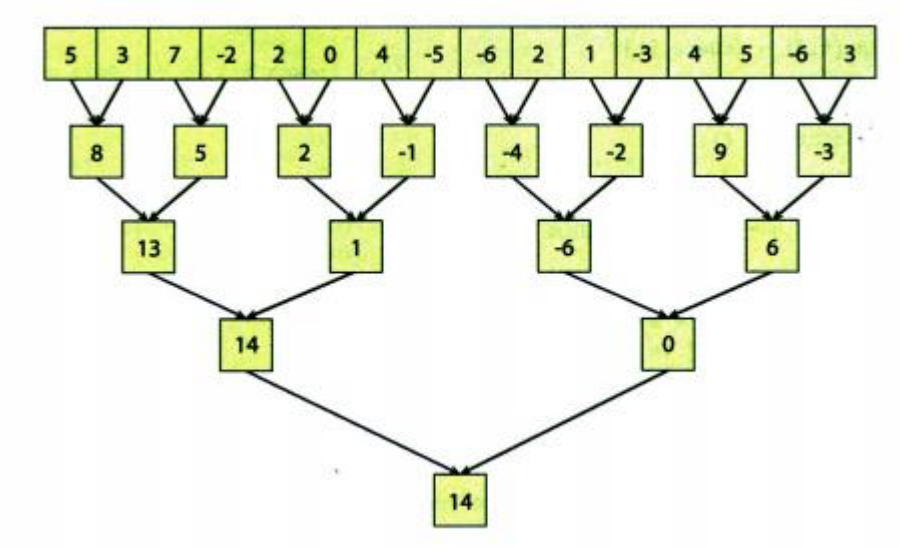

Рис. 1.

Реализация последовательного алгоритма префиксных сумм (который может запускаться на одном потоке на CPU, например) тривиальна.

Рассмотрим параллельную реализацию. Она стоит из двух частей, в каждой их которых повторяется $log2N$ шагов. На первом шаге строится дерево, состоящее из сумм (так же, как в случае с параллельной редукцией), на втором – результирующий массив по данному дереву сумм.

На рисунке 2 можно увидеть, как работает алгоритм для входной последовательности из 16 чисел. Подробное описание алгоритма можно увидеть в данной книге [11].

4. 3. Генерация случайных чисел.

Библиотека генерации случайных чисел cuRAND обеспечивает высокопроизводительное GPU-ускоренное создание случайных чисел. [16]

Существуют четыре алгоритма генерации случайных чисел:

- MRG32k3a
- Генератор «Вихрь Мерсенна»
- Генератор псевдослучайных чисел XORWOW
- Генераторы квазислучайных чисел Соболя, включая поддержку зашифрованных и 64-битных генераторов случайных чисел

Первые три являются псевдослучайными генераторами, последний – квазислучайным.

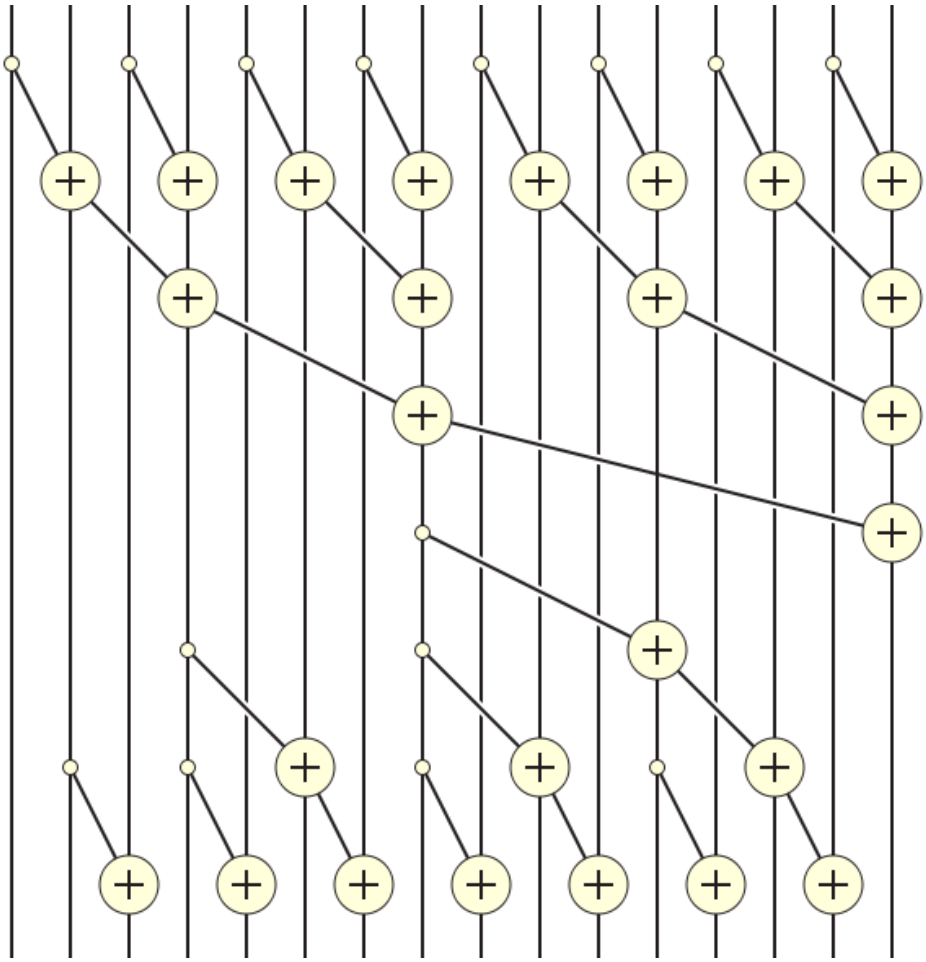

Рис. 2.

При использовании методов Монте-Карло важную роль играют качество и производительность генераторов случайных чисел.

Генератор псевдослучайных чисел создает все числа из диапазона с равной вероятностью, поэтому генерация случайной величины не влияет на возможность повторения ее генерации на следующем шаге. Генератор квазислучайных чисел, в свою очередь, уменьшает вероятность ее повторного появления, поэтому этот генератор покрывает пространство равномернее.

В силу особенностей двух типов генераторов, представленных выше, для симуляции Монте-Карло выгоднее использовать генераторы квазислучайных чисел Соболя.

Глава 5. Численные эксперименты.

Для запуска параллельной реализации алгоритма использованы три GPGPU NVidia Tesla M2050. Этот графический процессор имеет следующие характеристики:

Производительность операций с плавающей запятой	515 ГФлоп
Производительность операций с плавающей запятой одинарной точности (пиковая)	1.03 Тфлоп
Полный объем специальной памяти	3 ГБ GDDR5
Макс. потребление энергии	225 Вт
Системный интерфейс	PCIe x16 Gen2
Количество ядер CUDA	448

В данной главе будет представлено сравнение точности и скорости вычислений цены опциона на CPU и GPGPU. А также будет рассмотрена возможность масштабирования.

Вычисления произведены для опциона типа «колл» со следующими начальными данными:

1. $S_0 = 100$
2. $K = 110$
3. $T = 1$ год
4. $\sigma = 0.3$
5. $r = 0.1$

Если подставить данные значения в формулу Блэка-Шоулза (1) получим следующее:

$$d_1 = \frac{\ln\left(\frac{100}{110}\right) + (0.1 + \frac{0.3^2}{2})}{0.3} = 0.1656 \approx 0.17$$

$$d_2 = \frac{\ln\left(\frac{100}{110}\right) + (0.1 - \frac{0.3^2}{2})}{0.3} = -0.1344 \approx -0.13$$

$$N(d_1) = N(0.17) = 0.5675$$

$$N(d_2) = N(-0.13) = 1 - N(0.13) = 1 - 0.5517 = 0.4483$$

$$C(S_0, 0) = 100 * 0.5675 - 110 * e^{-0.1} * 0.4483 = 56.75 - 44.62 = 12.13$$

То есть цена такого европейского опциона будет равна 12,13. Стоит отметить, что формула Блэка-Шоулза следует определенным предположениям, которые практически не могут соответствовать реальным условиям.

5.1. Вычисление цены европейского опциона с помощью стохастического дифференциального уравнения

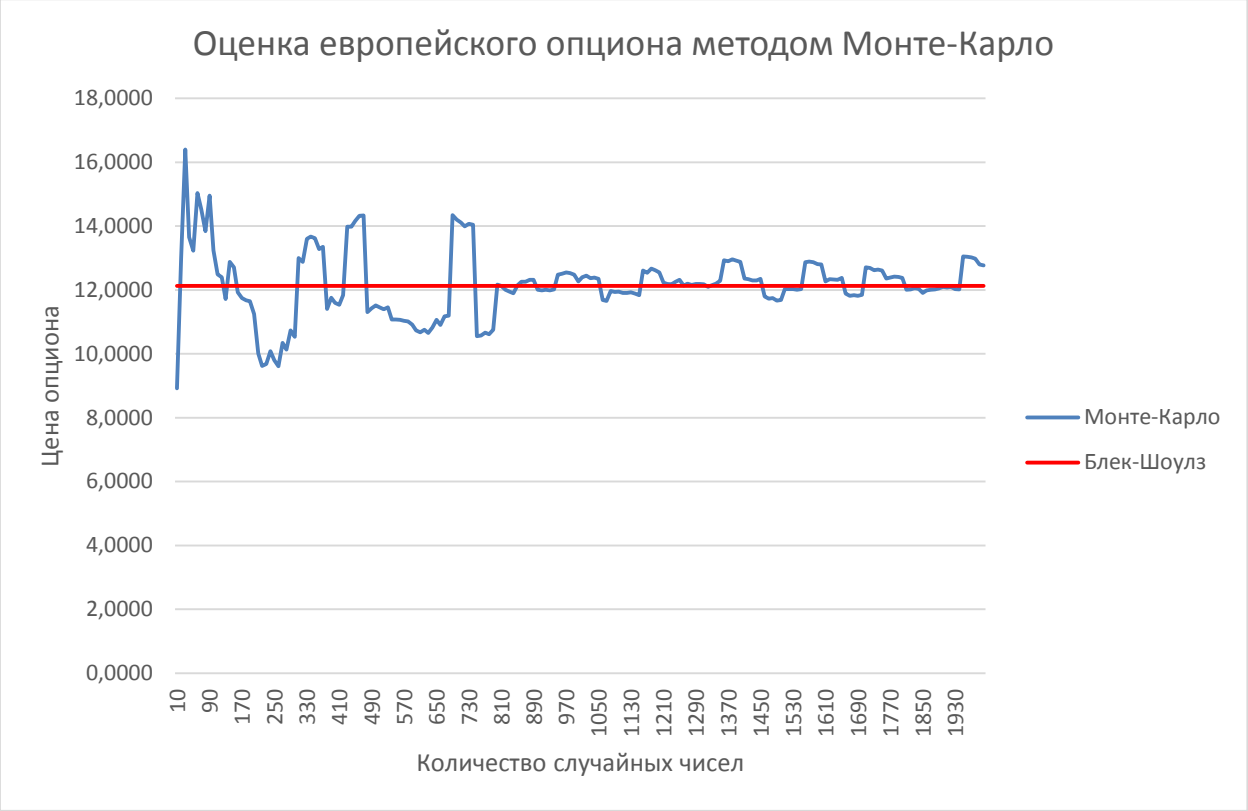

Рис. 3.

На графике рисунка 3 красной линией обозначено значение, которое вычислено с помощью формулы Блэка-Шоулза. Было сгенерировано малое количество случайных чисел, поэтому сходимость к значению, полученному по формуле Блэка-Шоулза, недостаточная. Попробуем увеличить количество случайных чисел.

На рисунке 4 видно, что при 20 млн. случайных величин сходимость намного выше, чем в предыдущем эксперименте с точностью до двух знаков после запятой.

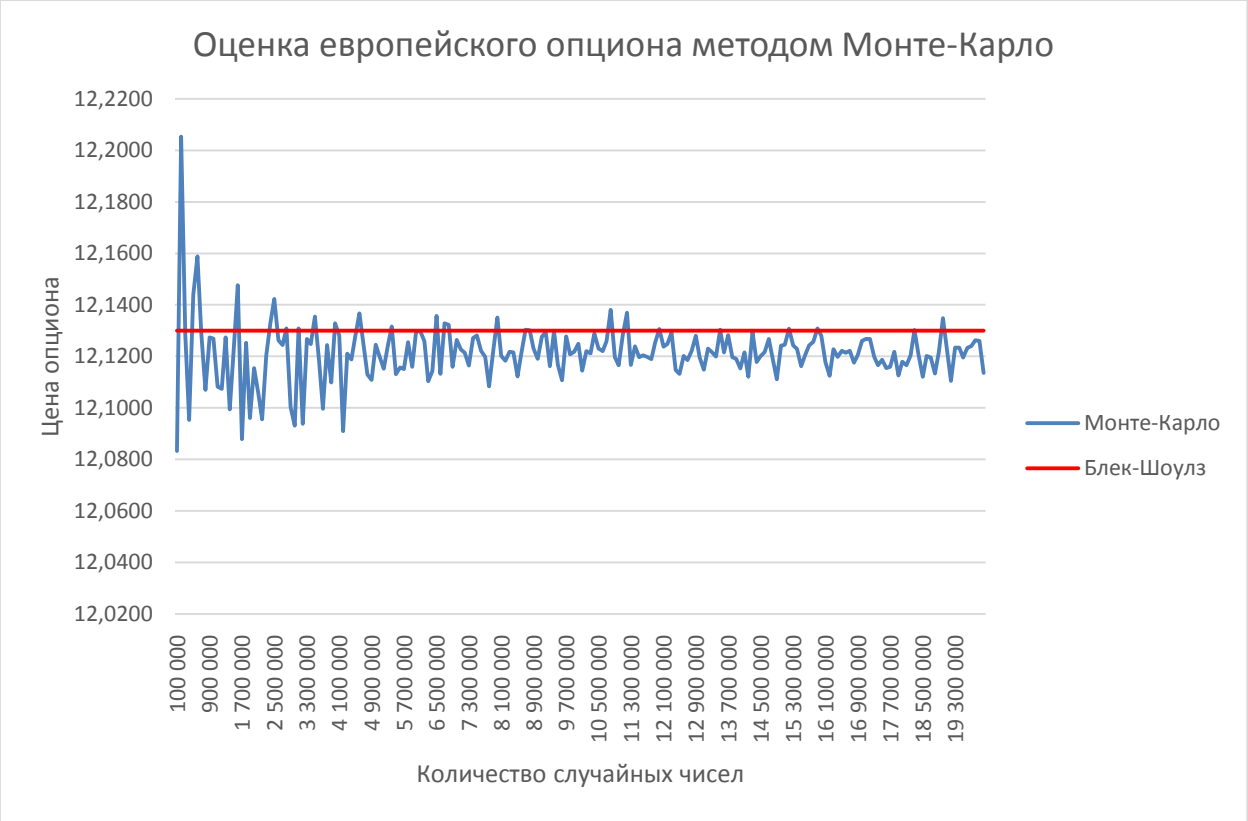

Рис. 4.

Важно определить ускорение на GPU, и сравнить скорость вычислений со скоростью на CPU.

На рисунке 5 показано ускорение на GPU, при различных наборах случайных чисел. При увеличении числа случайных величин ускорение растет, так как CPU исполняет поток инструкций последовательно, хоть и с

максимальной производительностью, а GPU исполняет большое число потоков инструкций параллельно.

В таблице 1 представлен сравнительный анализ скорости вычислений на CPU и GPU.

Кол-во случ. чисел	**Время на CPU, мс**	**Время на GPU, мс**
358 400	59	12
3 942 400	653	16
7 526 400	1242	20
11 110 400	1839	22
14 694 400	2442	26
20 428 800	3376	32

Табл. 1.

Таким образом, при 300 тыс. чисел ускорение равно 4.7, а при 2 млн. – 106.

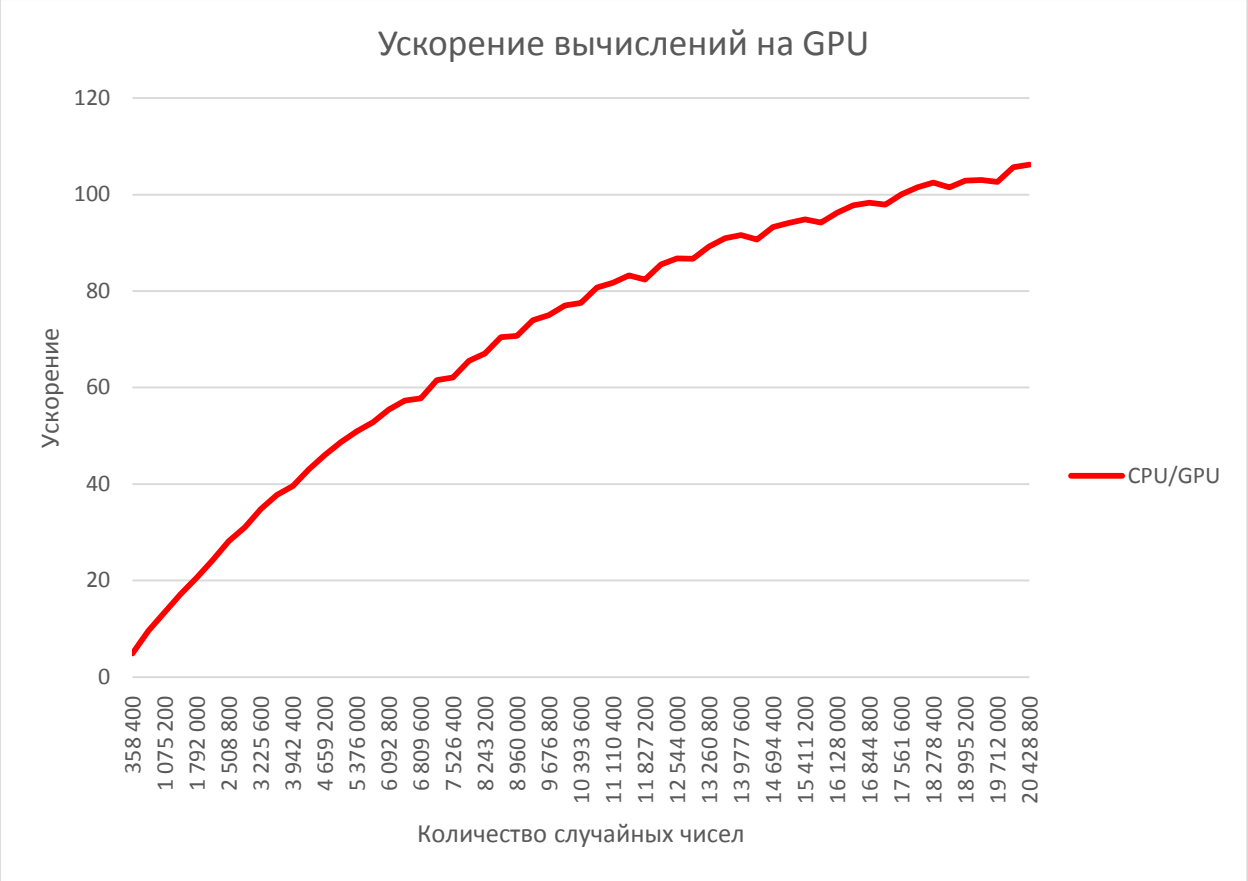

Рис. 5.

5. 2. Вычисление цены азиатского опциона с помощью стохастического дифференциального уравнения

Сложность азиатского опциона состоит в том, что он может быть исполнен в любой момент до определенного момента, указанного в контракте. Поэтому необходимо разделить время на отрезки и для каждой точки считать цену базового актива и среднюю.

Используем те же начальные данные, которые указаны в начале главы. На рисунке 6 видно среднее значение опциона, оно равно 3,92. Сходимость цены к этому значению представлена на графике.

Рисунок 7 изображает то, как изменяется скорость вычислений цены азиатского опциона в зависимости от использования CPU или GPU.

Можно заметить, что по сравнению с европейским опционом ускорение намного ниже. Это связано с особенностями опционов азиатского типа, так как на каждой итерации было необходимо вычислить промежуточное значение цены базового актива и использовать алгоритм префиксных сумм.

Так же, как и в предыдущем пункте, проведем сравнительный анализ скорости вычислений на CPU и GPU.

Кол-во случ. чисел	**Время на CPU, мс**	**Время на GPU, мс**
358 400	56	142
3 942 400	616	327
7 526 400	1182	635
11 110 400	1749	923
14 694 400	2299	1204
20 428 800	3180	1696

Табл. 2.

В соответствии с таблицей 2, ускорение в среднем равно 2. Причины такого малой разницы в скорости вычислений указаны выше.

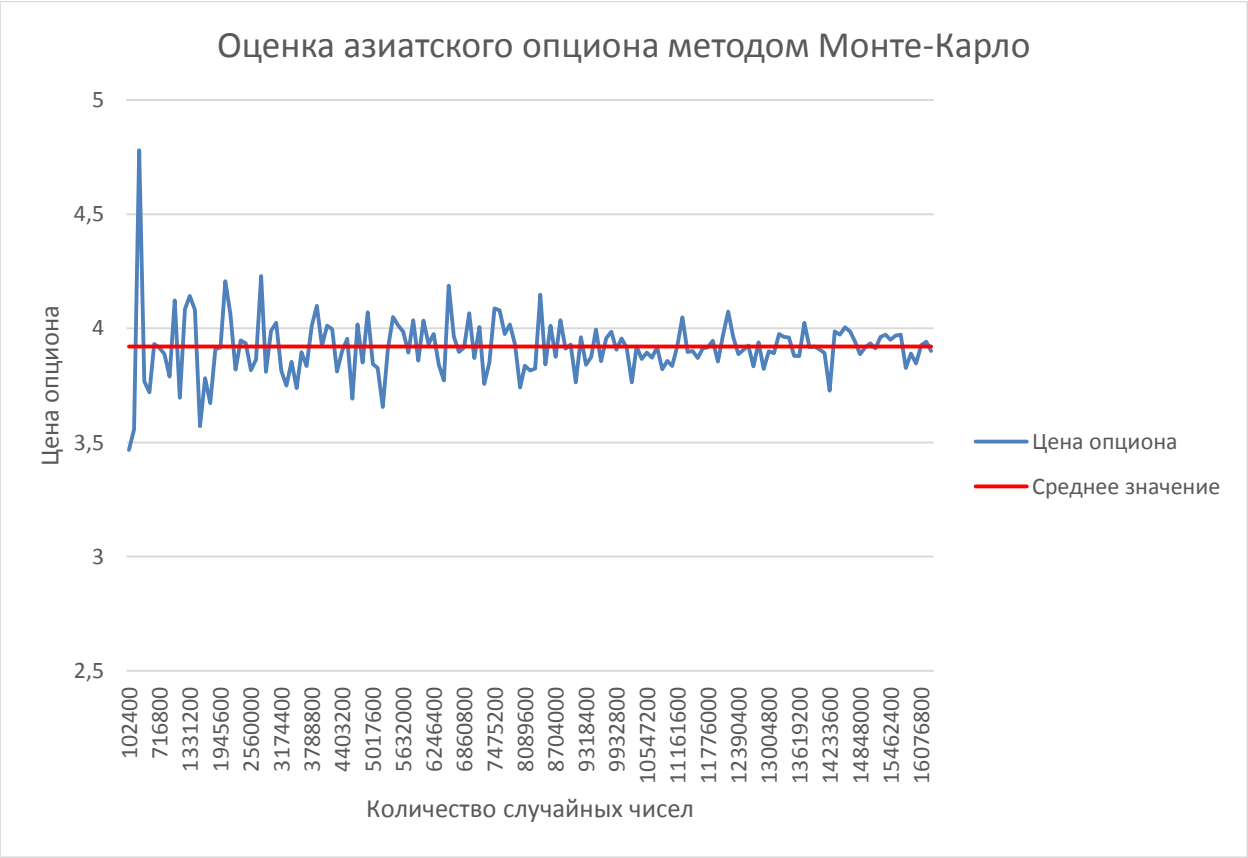
Оценка азиатского опциона методом Монте-Карло
Цена опциона
5
4,5
4
3,5
3
2,5
102400
716800
1331200
1945600
2560000
3174400
3788800
4403200
5017600
5632000
6246400
6860800
7475200
8089600
8704000
9318400
9932800
10547200
11161600
11776000
12390400
13004800
13619200
14233600
14848000
15462400
16076800
Количество случайных чисел
Цена опциона
Среднее значение

Рис. 6.

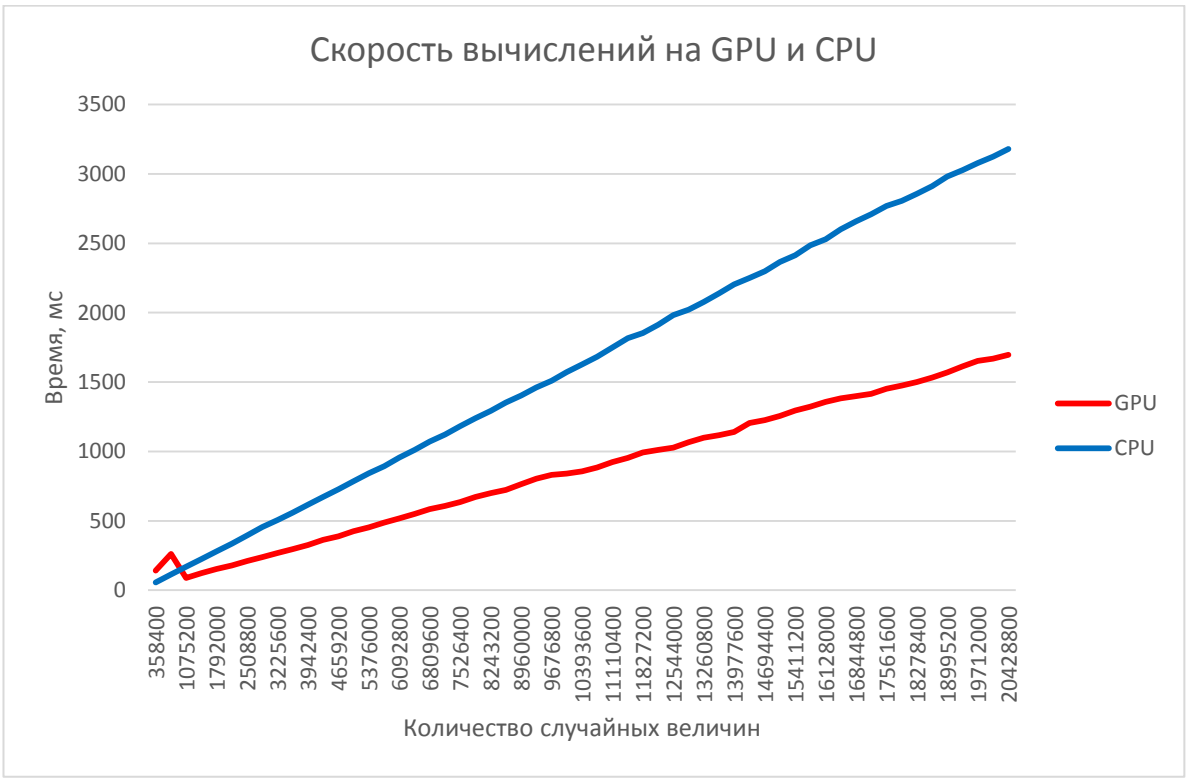
Скорость вычислений на GPU и CPU
Время, мс
3500
3000
2500
2000
1500
1000
500
0
358400
1075200
1792000
2508800
3225600
3942400
4659200
5376000
6092800
6809600
7526400
8243200
8960000
9676800
10393600
11110400
11827200
12544000
13260800
13977600
14694400
15411200
16128000
16844800
17561600
18278400
18995200
19712000
20428800
Количество случайных величин
GPU
CPU

Рис. 7.

5. 3. Вычисление цены европейского опциона с помощью континуального интеграла

Используем те же начальные данные и цену, полученную аналитически, равную 12,13.

Рис. 8.

На рисунке 8 видно, что вычисляемая цена опциона сходится к значению, вычисленному аналитически. Однако для достижения точности до второго знака после запятой требуется большее количество случайных чисел, чем в случае со стохастическим дифференциальным уравнением.

Сравним вычисления на CPU и GPU.

На рисунке 9 видно, что максимальное ускорение равно 74. Немного ниже, чем в случае со стохастическим дифференциальным уравнением, но это связано с некоторыми сложностями в данном алгоритме.

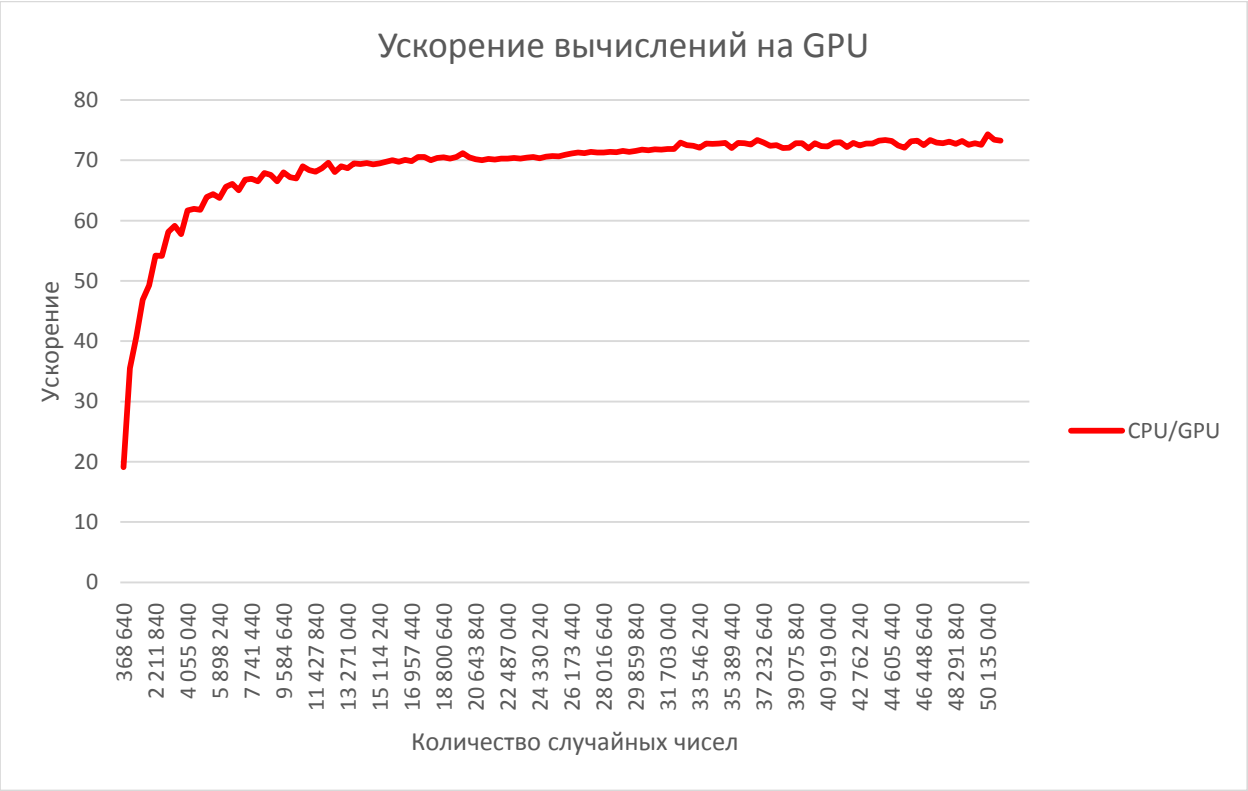

Рис. 9.

5. 4. Вычисление цены азиатского опциона с помощью континуального интеграла

Используем те же начальные данные. Вычислим цену азиатского опциона через континуальный интеграл.

На рисунке 10 видно, что вычисляемое значение опциона сходится к значению 3,61. Разница с результатом работы алгоритма со стохастическим уравнением (3,92) связана с основами подходов, однако является не существенной.

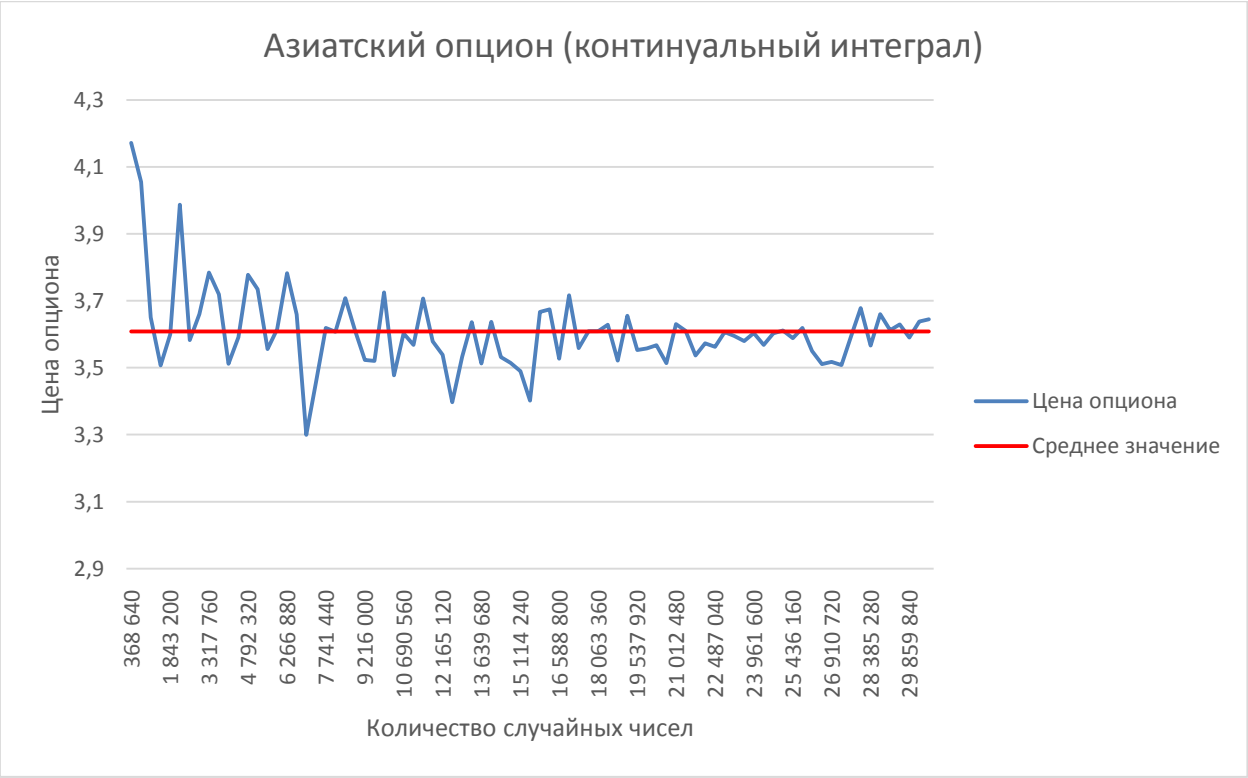

Рис. 10.

На рисунке 11 можно увидеть, что максимальное ускорение равно 83. Разница с ускорением вычислений с помощью стохастического уравнения связана с тем, что не используется алгоритм префиксных сумм.

5. 5. Масштабируемость

Масштабируемость в параллельных вычислениях играет важную роль. Так как при добавлении ресурсов важно чтобы производительность действительно увеличивалась.

Проверим алгоритм с использованием континуального интеграла на масштабируемость. Запустим его на одном, двух, трех GPU.


Ускорение вычислений на GPU
Ускорение
90
80
70
60
50
40
30
20
10
0
368 640
2 211 840
4 055 040
5 898 240
7 741 440
9 584 640
11 427 840
13 271 040
15 114 240
16 957 440
18 800 640
20 643 840
22 487 040
24 330 240
26 173 440
28 016 640
29 859 840
31 703 040
33 546 240
35 389 440
37 232 640
39 075 840
40 919 040
42 762 240
44 605 440
46 448 640
48 291 840
50 135 040
Количество случайных чисел
GPU

Рис. 11.

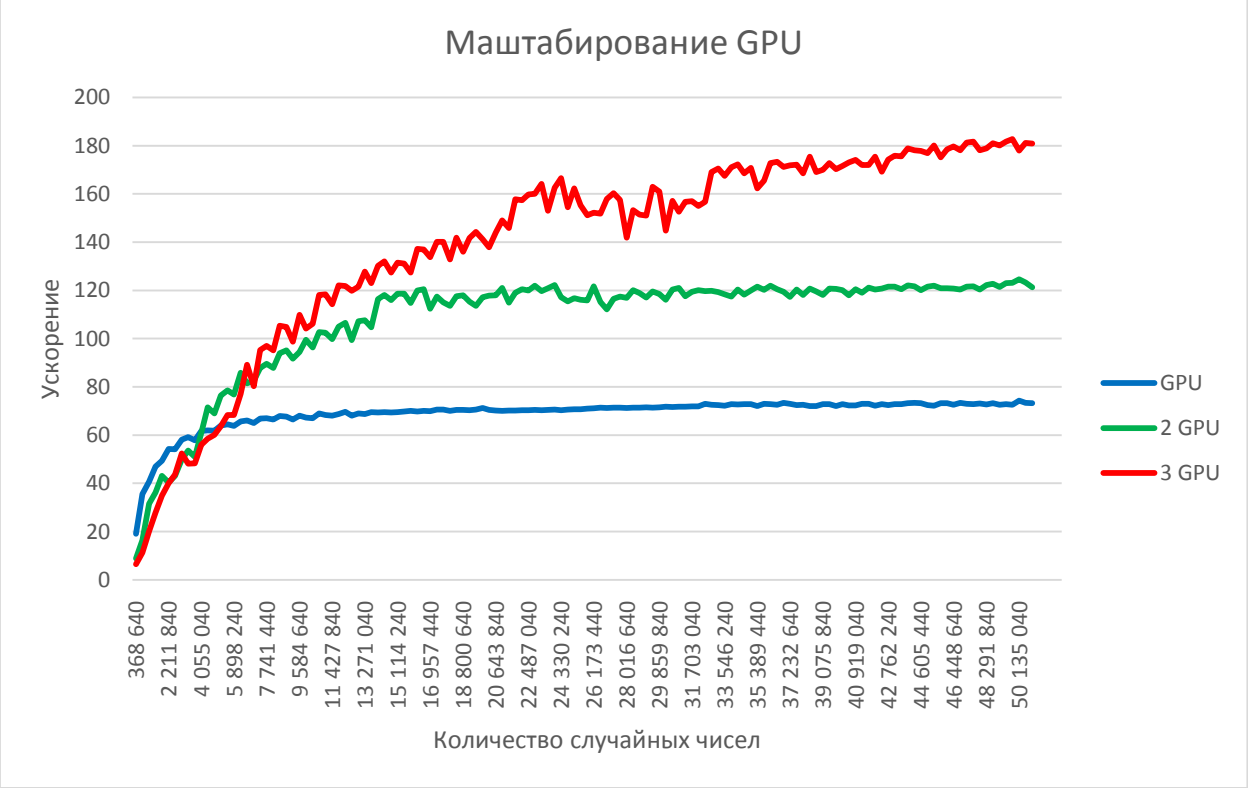
Маштабирование GPU
Ускорение
200
180
160
140
120
100
80
60
40
20
0
368 640
2 211 840
4 055 040
5 898 240
7 741 440
9 584 640
11 427 840
13 271 040
15 114 240
16 957 440
18 800 640
20 643 840
22 487 040
24 330 240
26 173 440
28 016 640
29 859 840
31 703 040
33 546 240
35 389 440
37 232 640
39 075 840
40 919 040
42 762 240
44 605 440
46 448 640
48 291 840
50 135 040
Количество случайных чисел
GPU
2 GPU
3 GPU

Рис. 12.

Рисунок 12 показывает хорошую масштабируемость данной задачи. Помимо этого, каждое ускорение стремится к конкретному значению. Например, при использовании одного GPU ускорение равно 72, при двух GPU – 120, при трех GPU – 180.

Такие результаты связаны с тем, что сами вычисления крайне мало зависят друг от друга.

Выводы

В данной работе проанализирована эффективность применения технологии CUDA с использованием GPGPU NVIDIA для расчета цен европейского и азиатского опционов методами Монте-Карло с помощью стохастического дифференциального уравнения и континуального интеграла. Для решения этой задачи с помощью симуляции Монте-Карло на GPU использовались алгоритмы параллельной редукции и префиксных сумм, в зависимости от типа алгоритма. Также был проведен анализ генераторов случайных чисел и выбор оптимального из них для поставленной задачи.

Проведенные расчеты показали, что использование GPU достаточно эффективно в применении к данной задаче. Для того чтобы достичь высокой точности вычислений, требовалось сгенерировать большое количество случайных величин, поэтому использование последовательного алгоритма на CPU является менее эффективным. Реализация методов Монте-Карло на GPGPU NVidia Tesla M2050 позволила достичь следующих результатов:

- 106-кратное ускорение для вычислений с помощью стохастического дифференциального уравнения для европейских опционов
- 2-кратное ускорение для вычислений с помощью стохастического дифференциального уравнения для азиатских опционов
- 70-кратное ускорение для вычислений с помощью континуального интеграла для европейских опционов
- 75-кратное ускорение для вычислений с помощью континуального интеграла для азиатских опционов

Вычисление цены азиатских опционов с помощью СДУ показало неэффективность применения GPU, так как использовался алгоритм префиксных сумм. Также алгоритмы обладают хорошей масштабируемостью.

Заключение

Так как, в настоящее время появляется больше суперкомпьютеров с гетерогенной архитектурой, которые используют CPU и GPU одновременно, технология CUDA применяется к огромному количеству задач, в том числе и в финансовой математике.

Для вычисления цены опционов широко используются методы Монте-Карло. Эти алгоритмы хорошо подходят для реализации на GPU, так как они основываются на большом количестве независимых операций, которые затем помогают получить конечный результат. То есть методы Монте-Карло обладают высокой степенью параллелизма.

По этим причинам была проведена данная работа. Сформулированные задачи в рамках этой работы были выполнены, а ее результаты были проанализированы.

Список литературы

[1]. F. Black, M. Scholes. The pricing of options and corporate liabilities // Journal Political Economy, 1973. Vol. 81. No. 3. P. 637-659

[2]. Paul Glasserman, Monte Carlo Methods in Financial Engineering, Springer, 2003, 599 p.

[3]. Соболь И.М., Численные методы Монте-Карло. М.: Наука, 1973. 307 с.

[4]. Опцион, Википедия. https://ru.wikipedia.org/wiki/Опцион

[5]. Джон К. Халл. Опционы, фьючерсы и другие производные финансовые инструменты. Изд. дом. Вильямс, 2008. 1044 с.

[6]. Hongbin Zhang. Pricing asian options using Monte Carlo methods // Examensarbete i Matematik, U.U.D.M. Project Report. 2009 Vol. 2009:7

[7]. Vadim Linetsky. 14 approach to financial modeling and options pricing // Computational Economics, 1998. Vol. 11: P. 129-163

[8]. Guido Montagna, Oreste Nicrosini. A path integral way to option pricing // Physica A: Statistical Mechanics and its Applications, 2002. Volume 310, Issues 3-4, P. 450-466

[9]. Э. Таненбаум, М. ван Стеен. Распределенные системы. Принципы и парадигмы. СПб.: Питер, 2003. 877 с.

[10]. Hyesoon Kim, Richard Vuduc, Sara Baghsorkhi. Performance Analysis and Tuning for General Purpose Graphics Processing Units (GPGPU) // Morgan & Claypool Publishers, 2012

[11]. А. В. Боресков и др. Предисл.: В. А. Садовничий. Параллельные вычисления на GPU. Архитектура и программная модель CUDA: Учебное пособие. Изд-во Московского университета, 2012, 336 стр.

[12]. А. В. Боресков, А. А. Харламов. Основы работы с технологией CUDA. Изд. дом. ДМК Пресс, 2010, 232 стр.

[13]. Официальный сайт Nvidia http://www.nvidia.ru/object/gpucomputingapplications-ru.html

[14]. Simoes B. General-purpose computing on the GPU (GPGPU). http://www.think-techie.com/2009/09/general-purpose-computing-on-gpu-gpgpu.html
[15]. NVIDIA CUDA C Programming Guide. Version 4.2 http://developer.nvidia.com/nvidia-gpu-computing-documentation
[16].Библиотека cuRand http://www.nvidia.ru/object/tesla-gpu-accelerated-libraries-curand-ru.html