РАСЧЕТ СТОИМОСТИ ЕВРОПЕЙСКОГО ОПЦИОНА С ИСПОЛЬЗОВАНИЕМ ПРОГРАММНО-АППАРАТНОЙ АРХИТЕКТУРЫ CUDA

Д. В. Мелещеня, П. Ю. Бранцевич
Кафедра программного обеспечения информационных технологий, Белорусский государственный университет информатики и радиоэлектороники
Минск, Республика Беларусь
E-mail: meleshchenya@gmail.com, branc@bsuir.edu.by

Численные методы нашли свое применение в различных сферах в том числе и в финансах. Одним из таких методов является метод Монте-Карло, который может быть использован для расчета цен производных финансовых инструментов. Реализация данного алгоритма с использованием технологии CUDA позволяет добиться высокой скорости вычислений и рассчитывать цены деривативов в реальном времени.

Введение

За последние десятилетия значительно выросла доля производных финансовых инструментов, таких как например опционы, фьючерсы, которые на ряду с обычными акциями торгуются на биржах по всему миру. В торговле более сложными и экзотическими инструментами также наметилась тенденция к росту, поскольку в условиях современных фондовых рынков они позволяют финансовым организациям более тщательно оговаривать нюансы заключаемых контрактов.

I. Производные финансовые документы

В сфере финансов, производный финансовый инструмент или деривати – это контракт, стоимость которого рассчитывается на основании стоимости базовых активов. Одним из активно торгуемых деривативов на рынке являются опционные контракты[1].

Говоря формальным языком, опционом – это договор, по которому одна сторона контракта (покупатель опциона) покупает право совершить покупку или продажу данного актива по определенной цене, заключенной в контракте, в определенный момент или на протяжение некоторого периода. При этом покупатель, в отличие от продавца, не обязан исполнять опцион и делает это лишь в том случае, когда ему это выгодно. Продавец в свою очередь получает за свое обязательство продать или купить актив вознаграждение, называемое премией опциона[2].

Сущетвует несколько основных видов опционов. В зависимости от права, которое предоставляет опцион, они делятся на:

- call опцион (подразумевает покупку);
- put опцион (подразумевает продажу).

В свою очередь по исполнению они делятся на:

- Американский опцион (может быть исполнен в течение всего срока);
- Европейский опцион (может быть исполнен только по окончании срока)

С течением времени цена базового актива может меняться, тем самым изменяя и цену производных инструментов зависящих от этого актива. Существует несколько основных моделей, позволяющих произвести оценку стоимости опционов: модель Блэка-Шоулза, биноминальная модель, модель Хестона, модель Монте-Карло. С развитием вычислительной техники последняя получила широкое распространение поскольку имеет довольно простую теоретическую базу, позволяет оценивать почти любые опционы, в том числе экзотические, а также проста в реализации на языке программирования. Суть метода заключается в получении большого числа реализаций стохастического процесса, в качестве которого выступает изменение цены базового актива. Генерирование случайного значения будущей цены актива осуществляется согласно следующей формуле [1]:

$$S_T = S_0 e^{(r-0{,}5\sigma^2)T+\sigma\sqrt{T}N(0,1)}$$

где S_T – цена актива в некоторый момент времени T;

S_0 – текущая цена базового актива;

r – математическое ожидание доходности акции, выраженной в % годовых (то, что чаще всего принято называть «ожидаемой доходностью» акции);

– стандартное отклонение доходности акции, выраженной в % годовых (данную величину чаще всего называют «волатильностью»);

N(0,1) – случайная величина, имеющая стандартное нормальное распределение (с нулевым мат ожиданием и стандартным отклонением, равным 0).

Общий алгоритм оценки опциона методом Монте-Карло состоит из следующих шагов:

1. Генерирование случайной будущей цены акции.
2. Расчет выплаты (денежного потока) по опциону.
3. Повторение шагов 1 и 2 M раз (где M – число порядка 10000)

4. Расчет среднего значения по опциону по результатам совершенных итераций.
5. Дисконтирование среднего значения выплат.

Шаги 1 и 2 алгоритма независимы друг от друга, т.е. каждая последующая итерация не зависит от предыдущей. Поэтому данный алгоритм может быть эффективно распараллелен. Иными словами, цикл, состоящий из этапов 1-3 может быть заменен на M процессов вычисления, выполняемых одновременно. Данный алгоритм не принимает во внимание возможность зависимости цен опциона для разных дат. Поэтому он пригоден лишь для расчета стоимости европейского опциона.

II. Применение технологии CUDA для расчета стоимости опциона

В настоящее время даже центральный процессор в большинстве компьютеров имеет несколько ядер и позволяет выполнять несколько потоков одновременно. Однако поскольку число M имеет порядок значительно больший, чем количество ядер современных процессоров, мы получим M/N итераций по N одновременно вычисляемых величин (N – количество вычислительных ядер). Это безусловно эффективнее, чем линейное выполнение алгоритма, но гораздо большие возможности по параллельному выполнению кода открывают графические процессоры, которые позволяют выполнять один и тот же код одновременно на десятках, а порой сотнях и тысячах ядер.

Алгоритм Монте-Карло для расчета цены опциона может быть эффективно реализован и запущен с использованием технологии CUDA, которая позволяет программистам реализовывать на специальном упрощённом диалекте языка программирования Си алгоритмы, выполнимые на графических процессорах Nvidia. В архитектуре CUDA используется модель памяти грид, кластерное моделирование потоков и SIMD-инструкции[3]. В сравнении с центральным процессором, ядра графического процессора значительно уступают в быстродействии, в доступном наборе команд и памяти ядра. Однако быстродействие целого граффического процессора выше за счет колличества вычислительных ядер. Все ядра организованы в виде блоков, которые в свою очередь объединены в таблицу. В каждый момент времени существует возможность получить номер текущего ядра в блоке и блока в таблице, что в свою очередь позволяет сопоставлять таблице ядер набор данных, представленных в виде массива. В случае с алгоритмом Монте-Карло, каждое генерирование цены акции выполняется отдельным ядром.

На физическом уровне не всегда получается достигать большого колличества параллельно выполняющихся потоков, поскольку возможности оборудования зачастую не позволяют включить в состав одного процессора большого колличества вычислительных ядер, достаточного для параллелльной обработки всего набора данных за один проход. Однако драйвер NVidida скрывает это от программиста и сам составляет расписание запуска процессов, что в свою очередь выглядит на более высоких уровнях абстакции как одновременный запуск.

Библиотека CUDA содержит стандартный генератор квазислучайных чисел, позволяющий получить различные распределения в том числе и нормальное, используемое в расчете цены базового актива. Таким образом и первый и второй шаг могут быть выполнены на графическом процессоре. Также следует заметить, что алгоритм расчета цены не содержит ветвлений, а поскольку процессор NVIDIA реализует SIMD архитектуру, это означает еще большую эффективность, так как нет необходимости разделять вычисления для каждой из ветвей и выполнять их последовательно, что пришлось бы сделать в случае алгоритма с уловными переходами.

Результат реализации и запуска алгоритма для GPU и CPU представлен в таблице 1.

Как видно, при расчете цены с небольшим количеством путей, затраты на копирование памяти из хоста в память графического процессора и организацию параллельного выполнения большого количества потоков оказали существенное влияние на результат и время выполнения оказалось больше. Однако уже на 0,2 млн. путей реализация на GPU оказалась эффективнее, а на 3,2 млн. превзошла CPU почти 2,5 раза, тем самым доказав, что можно получить ощутимый выигрыш в производительности, запуская расчеты цены опциона на графическом процессоре.

1. Hull, J. Options.Futures, and Other Derivatives (8thEdition) / J. C. Hull – 2013. – Hardcover. – 896 P.
2. Vine, S. Options: Trading strategy and Risk management / S. Vine – 2005. – Hardcover. – 382 P.
3. Sanders, J. CUDA by Example: An Introduction to General-Purpose GPU Programming / J. Sanders, E. Candrot – 2010. – Paperback. – P. 279.