**УДК 004.5**

# ИСПОЛЬЗОВАНИЕ ГРАФИЧЕСКОГО ПРОЦЕССОРА ДЛЯ УСКОРЕНИЯ ВЫЧИСЛЕНИЙ ТЕОРЕТИЧЕСКОЙ СТОИМОСТИ КРЕДИТНЫХ ДЕРИВАТИВОВ ТИПА ОПЦИОН

**Овечкин Р. М.**

*ФГБОУ ВПО «Пензенский государственный университет», Пенза, Россия (440027, г. Пенза, ул. Красная, 40), email:nifect@gmail.com*

**В статье представлен новый метод ускорения расчетов на основе использования графического процессора, который может применяться для приблизительной оценки одной из важнейших аналитических величин – теоретической стоимости производного финансового продукта типа Опцион в системах поддержки принятия решений в области торговли кредитными деривативами. Дана сравнительная характеристика производительности центрального процессора серверной системы и графического процессора, а также показаны различия в подходах работы с памятью, графически проиллюстрированы характеристики эволюции производительности обоих типов устройств. Описан способ применения технологии CUDA от компании Nvidia, позволяющий создавать программы для графического процессора на языке программирования высокого уровня. Пояснен принцип адаптирования сложных алгоритмов для выполнения на графическом процессоре.**

Ключевые слова: графический процессор, GPU, CUDA, ускорение расчетов.

# USING GPU FOR ACCELERATING EVALUATION OF THEORETICAL PRICE OF THE OPTION CREDIT DERIVATIVE

**Ovechkin R. M.**

*Penza State University, Penza, Russia (440027, Penza, Krasnaya st., 40), email:nifect@gmail.com*

**In the article the new approach has represented for accelerating of calculations based on using graphical processing unit which may be applied for approximate evaluating the one of the most significant analitical parameters - theoretical price of derivative financial product – Option integrated into decission support system in context of credit derivatives trading. Given comparative characteristics of performance of server CPU versus the performance of GPU as well as memory interacting process, the evolution of both types of devices is illustrated. describes method of using technology CUDA by Nvidia corporation which allows to create hi-level programs and run them on GPU. Explained the method of adaptation for complex algorithms for execution.**

Key words: graphical processor unit, GPU, CUDA, performance acceleration.

Для эффективного увеличения производительности вычислений в системах поддержки принятия решений в области торговли кредитными деривативами выполнение некоторых алгоритмов расчета может быть перенесено на новую платформу, использующую графические процессоры нового поколения.

**GPU** (Graphics Processor Unit) – графический процессор (ускоритель трехмерной графики).

В последнее время для компьютерных расчетов все чаще используют графические ускорители. Их использование обладает следующим рядом преимуществ:

- GPU обладают высокой производительностью, т.к. имеют несколько ядер;

- GPU оснащены памятью с широкой полосой пропускания (т.е. высокой скоростью чтения и записи).

Например, в видеочипах NVIDIA основной блок – это мультипроцессор с восемью-десятью ядрами и сотнями ALU в целом, несколькими тысячами регистров и небольшим

количеством разделяемой общей памяти. Кроме того, видеокарта содержит быструю глобальную память с доступом к ней всех мультипроцессоров, локальную память в каждом мультипроцессоре, а также специальную память для констант.

Самое главное – эти несколько ядер мультипроцессора в GPU являются SIMD (одиночный поток команд, множество потоков данных) ядрами. И эти ядра исполняют одни и те же инструкции одновременно, такой стиль программирования является обычным для графических алгоритмов и многих научных задач, но требует специфического программирования. Зато такой подход позволяет увеличить количество исполнительных блоков за счёт их упрощения.

Конвейеры на обычных системах могут поддерживать лишь ограниченное число одновременных потоков. На серверах, которые имеют четыре quad-процессора, могут работать одновременно только 16 параллельных потоков (или 32 потока, если есть поддержка технологии HyperThreading). Например, минимальная исполняемая единица на устройства, которая называется блоком задач (warp) и состоит из 32 потоков или нитей (threads). Все графические процессоры NVIDIA могут поддерживать 768 активных потоков на каждый мультипроцессор, а некоторые более поздние версии до 1024 потоков на мультипроцессор. Устройства, у которых более 30 потоковых мультипроцессора (GeForce GTX280, например), могут создавать до 30000 активных потоков. Более того, благодаря многопоточности устройства могут выполнять миллиарды задач параллельно.

Потоки в центральном процессоре обычно требуют много ресурсов. Операционная система должна обменивать потоки между собой, включать и выключать каналы обработки для обеспечения многопоточности. Переключение контекста (когда два потока меняются местами) – слишком медленное и дорогостоящее в смысле производительности. Для сравнения, на GPU потоки в десятки, а то и сотни раз "легче" потоков CPU. В типичной системе сотни потоков стоят в очереди на выполнение в блоках задач по 32 потока в каждом. Если процессору GPU приходится ждать, он просто переключается на выполнение другого потока. Так как регистры выделяются под активные потоки, не происходит обмена регистров и состояний между потоками GPU. Время жизни ресурсов, выделяемых на каждый поток, равно времени жизни самого потока.

Работа с памятью у GPU и CPU также несколько отличается. Так, не все центральные процессоры имеют встроенные контроллеры памяти, а у всех GPU обычно есть по несколько контроллеров, вплоть до восьми 64-битных каналов в чипе NVIDIA GT200. Кроме того, на видеокартах применяется более быстрая память, и в результате видеочипам доступна в разы большая пропускная способность памяти, что также весьма важно для параллельных расчётов, оперирующих с огромными потоками данных

На рисунке 1 приведена диаграмма, иллюстрирующая превосходство различных типов графических процессоров над обычными CPU. По оси y отложено количество операций с плавающей точкой в секунду – **FL**oat **OP**erations (GFLOPS = $10^9$ FLOPS).

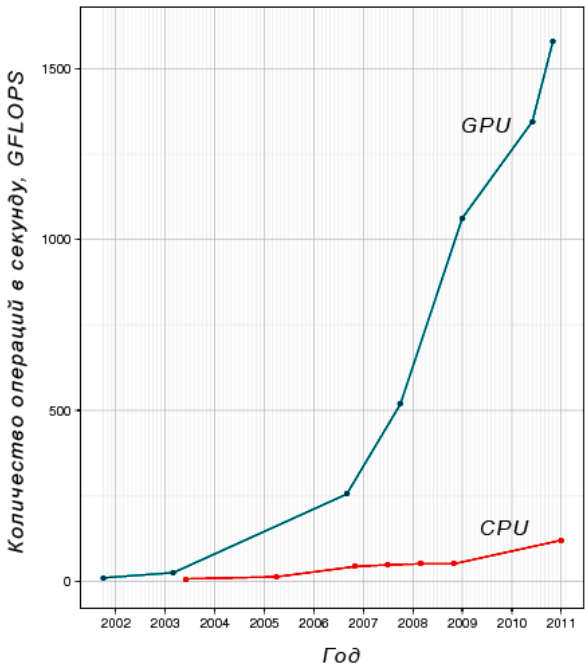


**Рисунок 1. Сравнение производительности CPU и GPU**

Понятно, что за счёт узкой специализации графический процессор может практически всю свою производительность отдавать на арифметические операции, тогда как архитектура CPU гораздо сложнее. Тактовая частота современных процессоров Intel остановилась в районе 3,4 ГГц из-за физических ограничений.

Например, производительность процессора Intel Core i7-975 XE 3,33 ГГц – 70 GFLOPS, в то же время NVIDIA Tesla s1070 выдает 4000 GFLOPS. С видеокартами могут сравниться разве что игровые консоли: PlayStation3 с 2000 GFLOPS и XBOX 360 с производительностью примерно 1000 GFLOPS. Разница очевидна.

Для всех современных видеокарт GeForce восьмого поколения и выше, Nvidia Tesla и некоторых Nvidia Quadro существует общий, стандартный интерфейс, обеспечивающий доступ к GPU- CUDA (Compute Unified Device Architecture).

Технология CUDA – это программно-аппаратная вычислительная архитектура NVIDIA, основанная на расширении языка Си, которая даёт возможность организации доступа к набору инструкций графического ускорителя и управления его памятью при организации параллельных вычислений. CUDA помогает реализовывать алгоритмы, выполнимые на графических процессорах видеоускорителей.

До 2006 года графические чипы были очень сложны для программирования, так как программисты вынуждены были использовать эквиваленты графических API для доступа к ядру процессора и памяти. Эта техника была названа GPGPU (General Purpose Programming using a Graphics Processing Unit). Графические API ограничивали виды приложений, которые могли быть написаны для этого чипа. Для этого требовались специальные знания и навыки программирования для графических чипов.

Все изменилось в 2007 году, когда NVIDIA представила CUDA. NVIDIA представили не только программную модель, облегчающую работу программиста, но также и аппаратные решения. Поэтому теперь существует возможность использовать видеокарту для вычислений, причем программный интерфейс не будет пересекаться с графическим. Для этого разработан единый универсальный интерфейс для доступа к памяти и ядру графического процессора.

CUDA позволяет разработчику реализовывать на специальном упрощенном диалекте языка программирования Си алгоритмы, выполнимые на графических процессорах, и включать специальные функции в текст программы на Си. CUDA также позволяет по своему усмотрению организовывать доступ к набору инструкций графического ускорителя и управлять его памятью, организовывать на нем сложные параллельные вычисления.

Стоит заметить, что GPU обладают некоторым недостатком: единственный поддерживаемый тип данных с плавающей точкой – float, который использует 4 байта для представления числа, что несколько ограничивает точность вычислений, в то время как для CPU доступны любые форматы представления чисел. Как правило, чаще всего используется double, кодируемый 8 байтами. Однако планируется, что в следующих моделях GPU будет поддерживаться тип double.

Главным недостатком GPU является его ограничение по сложности выполняемых алгоритмов. Это связано со спецификой задач, для которых GPU и были разработаны – задач обработки графики. Типичной задачей считается расчет цвета для каждого пикселя экрана при визуализации изображений трехмерной графики, например, если разрешение экрана 1680x1050 получается почит 2 миллиона пикселей, для каждого из которых нужно выполнить несложный алгоритм расчета цвета. Именно для подобного рода операций графические ускорители и были разработаны, и именно этот тип операций более всего подходит для выполнения на GPU. С появлением и дальнейшим развитием **шейдеров** (микропрограмм для расширенной обработки пикселей) стало возможно выполнять небольшие блоки кода обработки графики, используя ограниченное число переменных, входных и выходных параметров. Также ограничено количество инструкций итогового байт-кода, полученного в результате компиляции программы.

В процессе диссертационной работы необходимо было найти решение, позволяющее использовать GPU для ускорения обработки финансовых данных по несколько более сложным алгоритмам, чем позволяет CUDA. Чтобы иметь возможность обрабатывать более сложные структуры данных, используя более сложные алгоритмы, был придуман принцип декомпозиции алгоритмов, иными словами – разбиение сложного алгоритма на более простые с сохранением возможности многопоточной обработки данных.

Рассмотрим пример: пусть имеется сложный основной алгоритм (рисунок 2) обработки данных, поступающих на вход в виде большого массива объектов (сотни тысяч, миллионы записей).

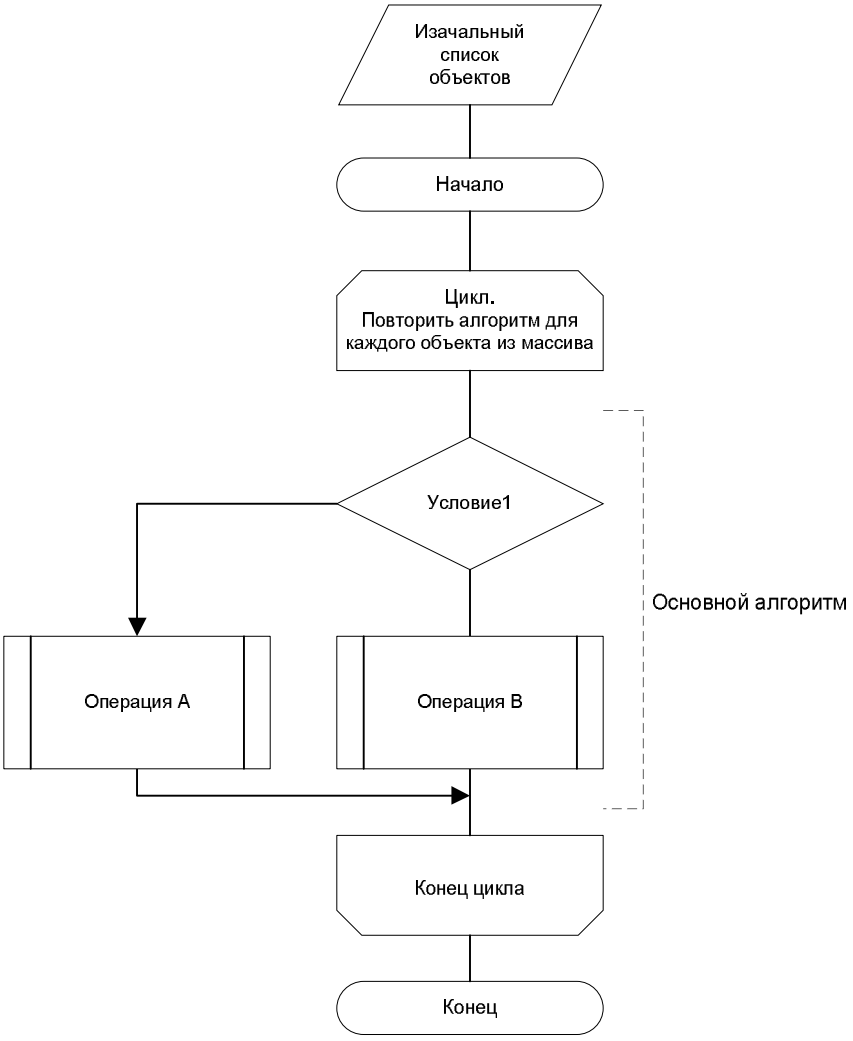


**Рисунок 2. Примерная схема основного алгоритма, подлежащего декомпозиции**

Алгоритм выполняется для каждого объекта из массива и может быть разделен на 3 составные части: «Условие 1», «Операция А», «Операция В». В зависимости от условия для каждого объекта выполнится только одна из операций – А или В. Иными словами, проверка

«Условие 1» выполняется для всех объектов, для одной части объектов выполнится «Операция А», а для другой части объектов – «Операция В».

При декомпозиции алгоритм разделяется на 3 отдельных алгоритма. Первый из них (рисунок 3) содержит в себе только «Условие 1», в зависимости от выполнения условия объекты поступают в 2 разных массива – «Массив А» или «Массив В», вместо того чтобы сразу идти на вход операций А,В. Массивы размещаются в видеопамяти, обладающей сравнительно небольшим временем доступа, что позволяет без потерь производительности разделить изначальный массив объектов на массив, соответственно каждому из алгоритмов.

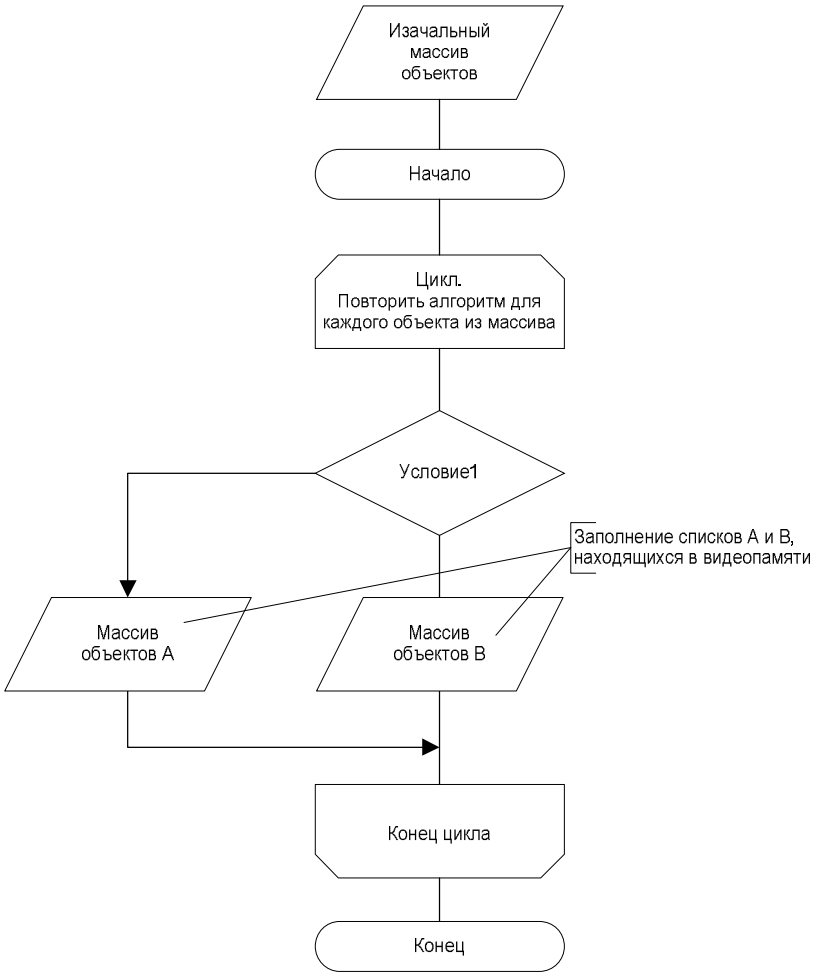


**Рисунок 3. Схема алгоритма, включающего только условие**

После выполнения первой части декомпозитного алгоритма в видеопамять загружаются по очереди и выполняются программа, выполняющая только операцию А, затем программа для операции В (рисунок 4). На вход операции А поступает массив объектов А, а на вход операции В – массив объектов В. После завершения работы всех под-алгоритмов результирующий список объектов копируется из видеопамяти в оперативную память сервера и обрабатывается CPU.

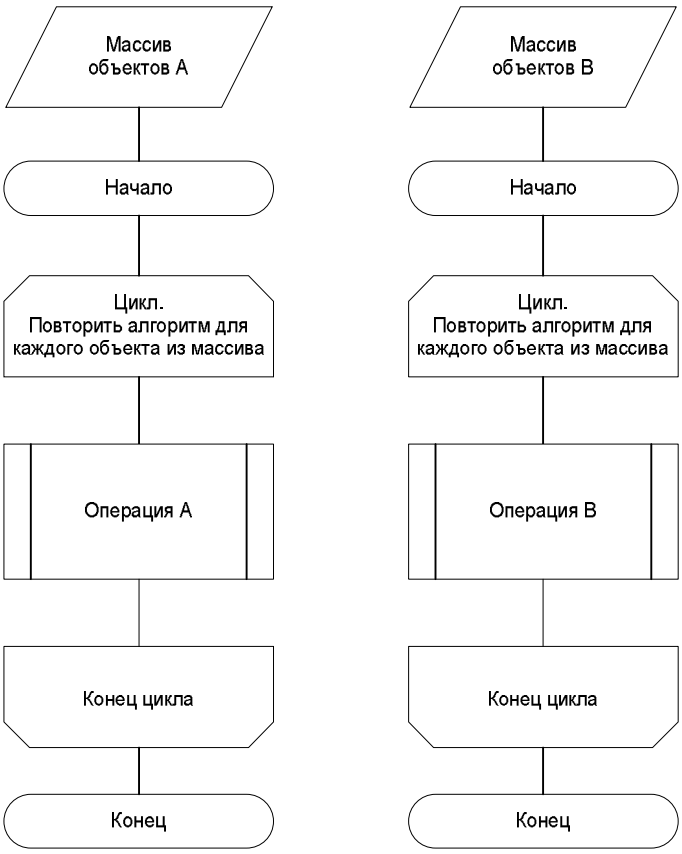


**Рисунок 4. Схема простых алгоритмов для операций A, B**

Таким образом, над всеми объектами могут быть произведены действия, аналогичные действиям по сложному алгоритму, однако выполненные во множестве параллельных потоков на GPU, в результате чего скорость выполнения выше в сотни раз.

## Список литературы

1. Кречетов Н., Иванов П. Продукты для интеллектуального анализа данных // ComputerWeek-Москва. – 1997. – № 14–15. – С. 32–39.

2. Щавелев Л. В. Способы аналитической обработки данных для поддержки принятия решений // Открытые системы. – 2008. – № 4–5.

3. Misra M. Design of Systolic arrays for QR Decomposition / Manoj Misra, Rajat Moona. - 1994. [Электронный ресурс]. – Режим доступа: http://www.cse.iitk.ac.in/users/moona/papers /iccse94.pdf/

4. Kerr A. GPU Performance Assessment with the HPEC Challenge / Andrew Kerr, Dan Campbell, Mark Richards // HPEC 2008, Lexington, 23-25 September 2008 / Lincoln Laboratory,

Massachusetts Institute of Technology. – Lexington, 2008. [Электронный ресурс]. – Режим доступа: http://www.ll.mit.edu/HPEC/agendas/proc08/Day3/58-Day3-Session6-Kerr-abstract.pdf.
5. Karimi K. A Performance Comparison of CUDA and OpenCL / Kamran Karimi, Neil G. Dickson, Firas Hamze // D-Wave Systems Inc. [Электронный ресурс]. – Режим доступа: http://arxiv.org/ftp/arxiv/papers/1005/1005.2581.pdf.

**Рецензенты:**

Бершадский А. М., д.т.н., профессор, зав. кафедрой САПР, ФГОУ ВПО «Пензенский государственный университет», г. Пенза.

Бождай А. С., д.т.н., профессор кафедры САПР, ФГОУ ВПО «Пензенский государственный университет», г. Пенза.