Министерство образования Республики Беларусь
Учреждение образования
«Полоцкий государственный университет»

ИНФОРМАЦИОННО-КОММУНИКАЦИОННЫЕ ТЕХНОЛОГИИ: ДОСТИЖЕНИЯ, ПРОБЛЕМЫ, ИННОВАЦИИ (ИКТ-2018)

Электронный сборник статей
I Международной научно-практической конференции,
посвященной 50-летию Полоцкого государственного университета

(Новополоцк, 14–15 июня 2018 г.)

Новополоцк
Полоцкий государственный университет
2018

Информационно-коммуникационные технологии: достижения, проблемы, инновации (ИКТ-2018) [Электронный ресурс] : электронный сборник статей I международной научно-практической конференции, посвященной 50-летию Полоцкого государственного университета, Новополоцк, 14–15 июня 2018 г. / Полоцкий государственный университет. – Новополоцк, 2018. – 1 электрон. опт. диск (CD-ROM).

Представлены результаты новейших научных исследований, в области информационно-коммуникационных и интернет-технологий, а именно: методы и технологии математического и имитационного моделирования систем; автоматизация и управление производственными процессами; программная инженерия; тестирование и верификация программ; обработка сигналов, изображений и видео; защита информации и технологии информационной безопасности; электронный маркетинг; проблемы и инновационные технологии подготовки специалистов в данной области.

Сборник включен в Государственный регистр информационного ресурса. Регистрационное свидетельство № 3201815009 от 28.03.2018.

Компьютерный дизайн М. Э. Дистанова.

Технические редакторы: Т. А. Дарьянова, О. П. Михайлова.

Компьютерная верстка Д. М. Севастьяновой.

211440, ул. Блохина, 29, г. Новополоцк, Беларусь
тел. 8 (0214) 53-21-23, e-mail: irina.psu@gmail.com

ISBN 978-985-531-602-3

Секция 4
ПРОГРАММНАЯ ИНЖЕНЕРИЯ, ТЕСТИРОВАНИЕ И ВЕРИФИКАЦИЯ ПРОГРАММ

УДК 004.424

ВЫСОКОПРОИЗВОДИТЕЛЬНЫЕ ВЫЧИСЛЕНИЯ НА ГРАФИЧЕСКИХ ПРОЦЕССОРАХ В ЗАДАЧАХ ОПТИМИЗАЦИИ СТРАТЕГИЙ МЕХАНИЧЕСКОЙ ТОРГОВЛИ

О.В. ЛИШАЁВ
(Полоцкий государственный университет, Беларусь)

Бэктестинг представляет собой концепцию анализа выбранной стратегии механической торговли на исторических данных. Предполагается, что если стратегия работала ранее, возможно она будет работать достаточно хорошо и в будущем. И наоборот, если стратегия работала на исторических данных не очень хорошо, маловероятно что она будет работать хорошо в будущем.

Процедура бэктестинга является важным шагом в оценке выгодности стратегий при определённых обстоятельствах. Это помогает пользователям узнать, как стратегия будет вести себя на рынке. Также это помогает модернизировать и улучшать торговую стратегию, незначительное изменение параметров в которой может иметь непредсказуемые последствия без должной проверки.

Количество исторических данных всё время накапливается, а стратегии механической торговли перешли от использования междневных данных на обработку данных в течение дня. Это увеличивает количество обрабатываемых данных в тысячи раз. Также используются индикаторы, которые требуют выполнения большого количества операций для их вычисления. Кроме того, для лучшего анализа стратегии иногда возникает необходимость тестирования стратегии для разных типов акции или даже для разных рынков. Все эти факторы делают тестирование стратегии более ресурсоёмким. При этом количество оптимизируемых параметров стратегии может быть большим. Для выполнения тестирования и поиска параметров стратегии в приемлемое время необходимы эффективные алгоритмы и подходы в построении приложений [1].

Значительного прироста производительности можно добиться, выполняя вычисления на графической карте. Современные графические процессоры предоставляют возможность параллельной обработки данных тысячами ядер, связанных памятью с высокой пропускной способностью. В частности, дорогостоящие аналитические вычисления, используемые в контексте финансов ого рынка, можно запрограммировать для выполнения на графической карте, что приводит к сокращению требуемого вычислительного времени в десятки раз [2].

Программы, которые запускаются на графической карте, состоят из двух частей - кода хоста и кода ядер. Код хоста управляет графической картой, инициализирует передачу данных и осуществляет планирование исполнения программы на устройстве. Ядро - это упрощённая программа, которая формирует базовую единицу параллелизма. В ядре определяются операции, которые будут выполнены над данными на графической карте. Ядра планируются к одновременному выполнению на нескольких скалярных процессорах графической карты в ОКМД (одиночный поток команд, множе-

ственный поток данных) режиме: каждый вызов ядра (называемый потоком) выполняет тот же код для отведённых ему входных данных.

Важным фактором производительности в программировании для графических карт является передача данных между хостом и устройством: все данные должны проходить через шину PCI Express, что является узким местом. Передача данных на устройство может нивелировать весь выигрыш от использования графического процессора. Данный факт особенно заметен для задач, активно использующих ввод/вывод: поскольку доступ к памяти хоста обычно в 2–3 раза быстрее, чем отправка данных по шине PCI Express, CPU может завершить обработку раньше, чем данные попадут на устройство [3].

Уменьшить время, необходимое для копирования данных на устройство можно за счёт использования закреплённой памяти хоста (pinned memory) [4]. При этом данные не копируются на устройство предварительно, а запрос идёт от ядра напрямую через шину PCI Express к памяти хоста. Таким образом, можно использовать размер памяти хоста, а не размер памяти устройства, которая зачастую не конфигурируема и имеет меньшие значения. Другим преимуществом этого метода является то, что передача данных совпадает с выполнением ядра. Из-за выборки значений по требованию, данные передаются на устройство во время выполнения ядра, что, возможно, сократит общее время выполнения. Однако могут возникнуть и накладные расходы из-за избыточной передачи данных между хостом и устройством, и производительность может и ухудшится [5].

Графические карты достигают высокой производительности благодаря массивному параллелизму. Это означает, что проблема должна быть хорошо распараллелена, чтобы получить максимальную производительность. Ещё одно ухудшение в производительности можно получить при исполнении в ядре кода, с разными путями исполнения. Поскольку каждый мультипроцессор имеет только один декодер команд, все скалярные процессоры выполняют одну и ту же инструкцию. Если некоторые потоки в рабочей группе расходятся, например, из-за зависимых от данных условий, мультипроцессор должен сериализовать пути выполнения кода, что приводит к потере производительности. Хотя эта проблема была несколько смягчена в новых видеокартах, всё ещё по-прежнему рекомендуется избегать сложных ветвлений исполнения кода, где это возможно [6].

В настоящее время для программирования графических процессоров используется две основных библиотеки: CUDA (Compute Unified Device Architecture - архитектура устройств с поддержкой универсальных вычислений) и OpenCL (Open Computing Language – открытый язык вычислений). Обе библиотеки реализуют программную модель ядер и предоставляют программный интерфейс, который позволяет процессору хоста управлять вычислениями на графическом процессоре и передачей данных из памяти хоста на графическую карту. В отличие от технологии CUDA, которая поддерживает только графические процессоры NVIDIA, OpenCL может работать на широком спектре устройств [7]. Однако CUDA обеспечивает более высокую производительность [6].

В обеих библиотеках в качестве основного языка программирования используется C. Также для обеих библиотек есть высокоуровневые языковые обёртки на Python – PyCUDA и PyOpenCL. Однако для CUDA также существует поддержка языка Java и .Net платформы. К тому же для CUDA количество разработанных прикладных библиотек значительно больше [8, 9], что позволяет использовать уже готовые и оптимизированные алгоритмы. Однако библиотеки, использующие OpenCL являются открытыми, и

при необходимость их проще модифицировать, в отличии от CUDA библиотек, исходный код которых чаще всего закрыт.

При выборе библиотеки для вычисления на графическом процессоре необходимо учитывать вероятную целевую платформу пользователя, а также приемлемую скорость выполнения программ. Использование языков с высокими абстракциями замедляет исполнение кода, но в свою очередь позволяет проще вести разработку. Таким образом, при написании библиотеки для тестирования стратегий механической торговли необходимо приходить к компромиссам между удобством разработки и скоростью выполнения программ. Также для ускорения исполнения программы нужно учитывать некоторые особенности программирования для графических процессоров: избегать по возможности ветвление в путях исполнения ядра, проводить эксперименты с использованием закреплённой памяти и копированием участков памяти на устройство. Для профилирования подобных экспериментов в каждой из рассмотренных библиотек есть специальные программы.

Литература

1. Jiarui, N. An Efficient Implementation of the Backtesting of Trading Strategies / N. Jiarui, Z. Chengqi // Parallel and Distributed Processing and Applications. ISPA 2005. Lecture Notes in Computer Science. - 2005. - Vol 3758. - p. 126–131.
2. Preis, T. GPU-computing in econophysics and statistical physics / T. Preis // The European Physical Journal Special Topics. - 2011 - Vol 194. - p. 87–119.
3. Boyer, M. Improving GPU Performance Prediction with Data Transfer Modeling / M. Boyer, J. Meng, K. Kumaran // Parallel and Distributed Processing Symposium Workshops & PhD Forum (IPDPSW) 2013 IEEE 27th International. - 2013. - p. 1097-1106.
4. NVIDIA. CUDA C Programming Guide. [Электронный ресурс] / NVIDIA Developer Documentation 1993-2018. - Режим доступа: https://docs.nvidia.com/cuda/cuda-c-programming-guide/index.html. - Дата доступа: 16.05.2018
5. Choosing Between Pinned and Non-Pinned Memory. [Электронный ресурс] / Computer Science | University of Virginia School of Engineering and Applied Science 1836–2018. – Режим доступа: https://www.cs.virginia.edu/~mwb7w/cuda_support/pinned_tradeoff.html. – Дата доступа: 16.05.2018.
6. Breß, S. GPU-accelerated Database Systems: Survey and Open Challenges / S. Breß, M. Heimel, N. Siegmund, L. Bellatreche, G. Saake // Transactions on Large-Scale Data- and Knowledge-Centered Systems XV. - 2014. - Vol 8920. - p. 1-35.
7. Conformant Products - The Khronos Group Inc. [Электронный ресурс] / The Khronos Group Inc. 2000-2018. - Режим доступа: https://www.khronos.org/conformance/adopters/conformant-products/. - Дата доступа: 16.05.2018.
8. OpenCL Libraries and Toolkits - IWOCL. [Электронный ресурс] / The International Workshop on OpenCL. 2013-2018. – Режим доступа: http://www.iwocl.org/resources/opencl-libraries-and-toolkits/. – Дата доступа: 16.05.2018.
9. Tools & Ecosystem | NVIDIA Developer. [Электронный ресурс] / NVIDIA Developer. 1993-2018. - Режим доступа: https://developer.nvidia.com/tools-ecosystem. - Дата доступа: 16.05.2018