УДК 519.688

Д. Н. Шинкарук, Ю. А. Шполянский, М. С. Косяков

АНАЛИЗ ЭФФЕКТИВНОСТИ ПРИМЕНЕНИЯ ТЕХНОЛОГИИ CUDA ДЛЯ РЕШЕНИЯ СИСТЕМ ЛИНЕЙНЫХ УРАВНЕНИЙ С ТРЕХДИАГОНАЛЬНЫМИ МАТРИЦАМИ В ЗАДАЧАХ РАСЧЕТА ЦЕН ОПЦИОНОВ

Исследованы возможности графического процессора при решении линейных систем с трехдиагональными матрицами. Показано, что целесообразно формировать матрицы непосредственно в глобальной памяти видеокарты, при этом достигается более чем 20-кратное ускорение вычислений по сравнению с однопоточным расчетом. С учетом обмена данными между оперативной памятью и памятью видеокарты достигнуто ускорение 5—8 раз при использовании механизма отображаемой памяти.

Ключевые слова: *CUDA, GPU общего назначения, система линейных уравнений, параллельная циклическая редукция, метод прогонки.*

Введение. В последние годы все более широко для высокопроизводительной обработки данных применяются дополнительные ускорители, принимающие на себя часть вычислительной нагрузки. В частности, к таким ускорителям относятся видеокарты с графическими процессорами (ГП), изначально предназначенные для использования в качестве устройств обработки графики. Производительность этих устройств растет с каждым годом, а реализации алгоритмов обработки данных на ГП позволяют значительно повысить скорость вычислений по сравнению с использованием центрального процессора (ЦП) [1].

Результаты многочисленных исследований технологии CUDA (*Compute Unified Device Architecture*), позволяющей производить массово-параллельные вычисления с использованием ГП компании NVIDIA, показали значительное увеличение скорости обработки данных в не связанных с графическими приложениями областях [2—5].

В настоящей работе рассматривается задача расчета теоретических цен опционов как численного решения дифференциального уравнения в частных производных (ДУЧП), которое часто сводят к последовательному решению систем линейных алгебраических уравнений (СЛАУ) с трехдиагональными матрицами [6]. Целью работы является анализ эффективности применения технологии CUDA для ускорения решения СЛАУ с трехдиагональными матрицами на ГП и выявление условий, при которых достигается наибольшее преимущество ГП перед ЦП. При сравнении производительности таких вычислений использованы современные ЦП Intel и ГП компании NVIDIA. Детально изучены особенности применения различных методов обмена данными между оперативной памятью компьютера (ОЗУ) и памятью видеокарты, сформулированы соответствующие рекомендации.

Особенности архитектуры CUDA. Технология CUDA основана на следующей концепции: ГП выступает в роли массово-параллельного сопроцессора к ЦП и обладает своей иерархией памяти [7]. Наиболее востребованы разделяемая память видеокарты, располагающаяся на кристалле ГП и обладающая малым временем доступа, но малым объемом, и сравнительно „медленная“ глобальная память, имеющая наибольший объем.

Программа, написанная с использованием технологии CUDA, состоит из двух частей: последовательного кода, выполняющегося на ЦП, и параллельного кода, который выполняется на ГП. Параллельная часть программы, называемая ядром (*kernel*), исполняется одновременно большим количеством потоков (*threads*), которые организуются в блоки. С использованием последовательного кода подготавливаются и пересылаются данные из ОЗУ компью-

тера в память видеокарты, после чего осуществляется запуск ядра. По окончании работы ядра результаты передаются обратно в оперативную память.

Такие особенности архитектуры и программной организации работы ГП приводят к тому, что алгоритмы, предназначенные для работы на ЦП, не могут быть успешно применены для графического ускорителя. ГП требует применения специализированных параллельных алгоритмов обработки данных.

Методы решения СЛАУ. При организации вычислений на центральном процессоре СЛАУ обычно решают методом прогонки или LU-декомпозиции [8]. Алгоритм решения строго последователен и не может быть эффективно распараллелен. Поэтому при решении СЛАУ на ГП необходимы другие методы. В настоящей работе применен метод параллельной циклической редукции (ПЦР, *Parallel Cyclic Reduction*), который характеризуется большим объемом вычислений, но обладает высокой степенью параллелизма [9].

На рис. 1 представлена схема метода ПЦР. СЛАУ обрабатывается на ГП одним блоком, при этом расчетом каждого неизвестного x_i занимается отдельный поток P_i. Метод выполняется пошагово. На шаге j производится вычисление новых коэффициентов P_i^j для уравнений системы с использованием коэффициентов, рассчитанных на предыдущем шаге $j-1$. Взаимодействие потоков осуществляется через разделяемую память. На последнем этапе вычислений (на рис. 1 обозначен ромбами) производится итоговый расчет неизвестных x_i. Работа алгоритма требует $\log_2(N)$ шагов, где N — количество уравнений в СЛАУ.

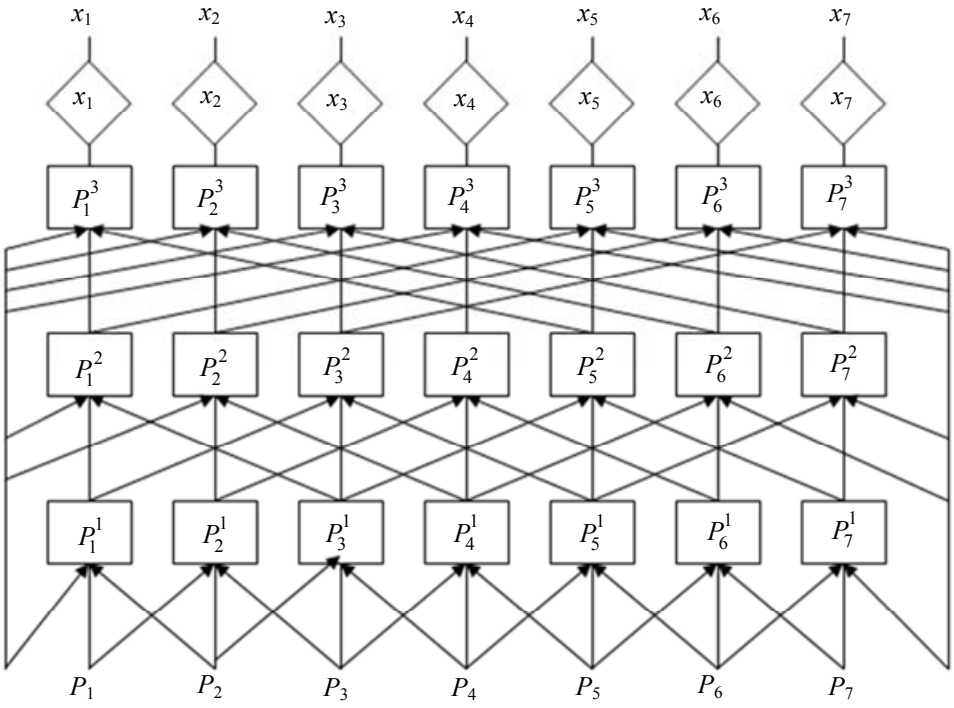

Рис. 1

Вычислительная сложность ПЦР пропорциональна $N\log_2(N)$, в то время как сложность метода прогонки линейно зависит от количества уравнений в системе. Но при массово-параллельном исполнении каждый отдельный поток выполняет лишь небольшую часть общего объема вычислений, пропорциональную $\log_2(N)$.

Решение одной системы. Расчеты цен опционов и измерения времени вычислений проводились на персональном компьютере (ПК) со следующими параметрами: ЦП Intel Core i5 3,8 ГГц; 4 ГБ ОЗУ; ГП NVIDIA GTX 580; ОС Ubuntu 11.04 (64 бит); версия драйвера CUDA 270.41.19 (64 бит).

При организации расчетов на ГП в ОЗУ компьютера формировались 4 массива данных, соответствующие трем диагоналям матрицы и правому столбцу СЛАУ. Для матрицы СЛАУ

обеспечивалось требование диагонального преобладания. Массивы передавались в память видеокарты, после чего запускалось ядро для обработки данных на ГП. Вектор результатов копировался из памяти устройства обратно в ОЗУ. Измеряемыми характеристиками были полное время T_f (*full*) от начала передачи данных на видеокарту до момента возвращения результатов в ОЗУ и чистое время вычислений на видеокарте, т.е. время выполнения ядра T_k (*kernel*). Согласно идеологии ПЦР, число запускаемых потоков выполнения совпадало с числом уравнений N. Для ускорения расчетов все потоки запускались в одном блоке, а требуемые данные располагались в разделяемой памяти.

Таблица 1

Время решения одной СЛАУ

N	2	4	8	16	32	64	128	256	512	1024
T_k, мс	4	4	5	6	7	7	8	9	12	23
T_f, мс	117	113	119	117	119	119	119	121	125	263
T_{CPU}, мс	1	1	2	2	2	3	5	11	19	39

Как видно из табл. 1, время работы ядра T_k на ГП сопоставимо со временем решения системы уравнений методом прогонки на ЦП T_{CPU}, однако для СЛАУ с $N \le 128$ превышает его. Связано это с тем, что при массово-параллельных вычислениях на ГП существуют затраты на создание и инициализацию потоков выполнения. Для CUDA время инициализации практически не зависит от количества создаваемых потоков, поэтому наблюдается более медленный рост T_k по сравнению с T_{CPU}. Важно отметить, что значение T_{CPU} линейно зависит от числа уравнений N в решаемой СЛАУ.

Значение полного времени T_f слабо изменяется с ростом размерности СЛАУ N, но в десятки раз превышает T_k и T_{CPU} (см. табл. 1). На рис. 2 представлена временная диаграмма работы видеокарты: передача данных из ОЗУ в память видеокарты (*1*) и обратно (*5*); работа ядра (*3*); простои (*2*, *4*). Время работы устройства составило 7,2 % от общего времени. Длительные простои (белые области) связаны с отработкой драйвера устройства, планированием процессов в системе, синхронизацией и пр.

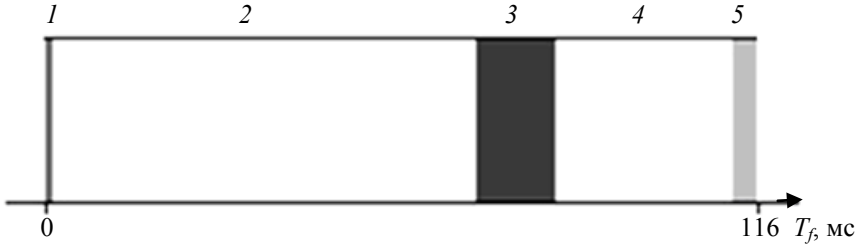

Рис. 2

Достичь преимущества ГП перед ЦП возможно, увеличив занятость устройства и эффективно используя его вычислительные ресурсы. Одним из способов достижения этой цели является параллельное решение большого числа СЛАУ.

При анализе погрешностей вычислений рассматривалась невязка решения СЛАУ. Наибольшее наблюдаемое значение невязки для метода ПЦР составило $\Delta = 8{,}5 \cdot 10^{-5}$, в то время как для метода прогонки (МП) $\Delta = 7{,}8 \cdot 10^{-6}$ при $N = 300$. Причина такого различия заключается в разном количестве арифметических операций, производимых над элементом данных. В реализации метода ПЦР осуществляется несколько больше операций умножения по сравнению с МП. Важно отметить, что для ПЦР количество арифметических операций определяется размерностью системы N: при увеличении числа уравнений в СЛАУ возрастает количество шагов решения. В наших экспериментах для $N = 8$ невязка составила $\Delta = 8 \cdot 10^{-8}$, а для $N = 64$ уже $\Delta = 1{,}2 \cdot 10^{-6}$. Эти значения не зависят того, где запускаются методы расчета: на ЦП или ГП, следовательно, использование ГП не приводит к дополнительным погрешностям.

Параллельное решение многих систем. На рис. 3, *а* представлена зависимость времени T_k и T_f расчетов на ГП от количества M одновременно обрабатываемых СЛАУ при $N = 256$. Как видно, при небольшой нагрузке значения T_k (*1*) и T_f (*2*) мало меняются, что связано с недостаточной загрузкой устройства. При $M = 64$ достигается предел одновременно запущенных потоков выполнения (16 384 потоков для видеокарты NVIDIA GTX 580), после чего время вычислений растет линейно. Стоит отметить, что время расчета на ЦП T_{CPU} (*3*) изменяется линейно во всей области рассматриваемой нагрузки.

На рис. 3, *б* представлена зависимость скорости решения СЛАУ V (определяемой как отношение числа M решаемых СЛАУ ко времени их расчета T) от нагрузки на систему. Рост скорости выполнения расчетов V_f (*1*) на ГП с увеличением M происходит за счет сокрытия задержек доступа к памяти при одновременной работе большого количества потоков выполнения, при этом скорость ЦП (кривая *2*) практически не изменяется.

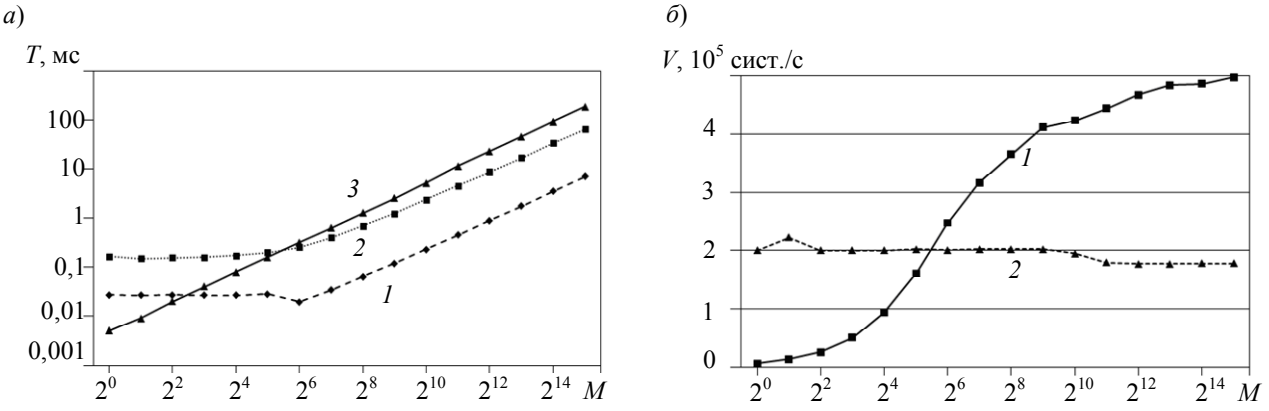

Рис. 3

При решении большого числа СЛАУ объем передаваемых данных между ОЗУ и памятью видеокарты значительно возрастает, что существенно сказывается на общем времени расчетов (табл. 2). Так, при увеличении числа решаемых систем M возможно сокращение чистого времени вычислений на ГП перед ЦП в 25 раз. Однако при пересылке данных между ОЗУ и памятью видеокарты преимущество в скорости расчетов с использованием ГП составляет 2,6 раза, 90 % общего времени работы устройства отводится на передачу данных. Один из возможных выходов — такая организация вычислительного процесса, при которой матрицы СЛАУ формируется непосредственно в глобальной памяти видеокарты.

Таблица 2

Время решения различного количества систем уравнений

M	T_k, мс	T_f, мс	T_{CPU}, мс	T_{CPU} / T_k	T_{CPU} / T_f
1	0,027	0,167	0,005	0,18	0,03
16	0,026	0,171	0,08	3,01	0,47
256	0,065	0,701	1,263	19,41	1,80
1024	0,231	2,418	5,255	22,70	2,17
4096	0,893	8,763	23,077	25,84	2,63

Видеокарты последних поколений позволяют осуществлять асинхронную передачу данных из ОЗУ в память видеокарты и обратно одновременно с исполнением кода ядра [10]. Таким образом достигается эффективное функционирование приложения при наличии большого объема входных данных. Одно из ограничений при этом — необходимость обозначения области ОЗУ, в которой хранятся передаваемые данные, как невыгружаемой области памяти. При реализации этого метода входные данные „вручную“ разбиваются на фрагменты таким образом, чтобы время передачи одного фрагмента приближалось ко времени его обработки.

Таблица 3

Полное время решения СЛАУ при синхронном и асинхронном обмене данными

M	T_f^s, мс	T_f^a, мс	T_{CPU}, мс	T_{CPU} / T_f^a
1	0,167	0,041	0,005	0,12
16	0,171	0,060	0,08	1,32
256	0,701	0,389	1,263	3,25
1024	2,418	1,374	5,255	3,82
4096	8,763	5,141	23,077	4,49

Применение асинхронной передачи данных позволило уменьшить общее время расчетов T_f^a более чем в 1,5 раза по сравнению со случаем синхронного обмена T_f^s (см. табл. 3). Для небольшого числа решаемых систем M наблюдается уменьшение времени расчетов в 3—4 раза, поскольку при асинхронной передаче данных исчезает значительная часть простоев (см. рис. 2). Однако необходимость „ручного“ разбиения входных данных на части весьма усложняет применение данного метода.

Нами исследована эффективность применения механизма отображаемой (*mapped*) памяти для прямого доступа к данным в ОЗУ из исполняемого на ГП ядра. В этом случае соответствующая область ОЗУ также должна быть обозначена как невыгружаемая. Результаты сравнения полного времени выполнения для синхронного T_f^s, асинхронного T_f^a обмена данными и T_f^m при использовании отображаемой памяти приведены в табл. 4.

Таблица 4

Сравнение времени расчета

M	T_f^s, мс	T_f^a, мс	T_f^m, мс	T_{CPU}, мс	T_{CPU} / T_f^m
1	0,167	0,041	0,026	0,005	0,19
16	0,171	0,060	0,038	0,08	2,10
256	0,701	0,389	0,269	1,263	4,69
1024	2,418	1,374	1,001	5,255	5,25
4096	8,763	5,141	3,101	23,077	7,44

Из таблицы видно, что использование отображаемой памяти дает значительное преимущество перед другими методами обмена данными. В этом случае достигается наилучшее перекрытие процессов передачи данных и решения СЛАУ во времени. Полное время выполнения на ГП по сравнению с ЦП может сокращаться в 5—8 раз для большого числа СЛАУ ($N > 1000$).

Выводы. В работе проанализирована эффективность применения технологии CUDA, позволяющей производить массово-параллельные вычисления с использованием ГП компании NVIDIA, для ускорения решения СЛАУ с трехдиагональными матрицами в задачах расчета цен опционов как численного решения ДУЧП. При организации вычислений на ЦП СЛАУ решались методом прогонки, который является строго последовательным, поэтому при расчетах на видеокарте использовался метод ПЦР.

Результаты показали допустимую для практических задач погрешность математических вычислений вещественных чисел с плавающей точкой для метода ПЦР. Использование ГП не вносит дополнительных погрешностей.

Сравнение времени выполнения расчетов показало неэффективность применения ГП для решения одной СЛАУ ввиду малой загрузки устройства. Для увеличения доли занятости ГП авторы рекомендуют решать одновременно большое число СЛАУ, что актуально для современных систем алгоритмической торговли. Матрицы СЛАУ целесообразно формировать непосредственно в глобальной памяти видеокарты.

Наша реализация алгоритма ПЦР на видеокарте NVIDIA GTX 580 позволила достичь более чем 20-кратного ускорения вычислений по сравнению с однопоточным расчетом на ЦП

Intel Core i5 3,8 ГГц при хранении данных в памяти видеокарты и одновременном решении более 1000 СЛАУ. Для задач, в которых формирование матриц СЛАУ непосредственно в памяти видеокарты невозможно, был проведен детальный анализ различных способов обмена данными между ОЗУ и памятью видеокарты. В этом случае для ГП удалось добиться ускорения в 5—8 раз при использовании механизма отображаемой памяти.

СПИСОК ЛИТЕРАТУРЫ

1. *Simoes B.* General-purpose computing on the GPU (GPGPU) [Электронный ресурс]: <http://www.think-techie.com/2009/09/general-purpose-computing-on-gpu-gpgpu.html>.
2. *Zhang Y., Cohen J., Owens J. D.* Fast Tridiagonal Solvers on the GPU // Proc. of the 15th ACM SIGPLAN Symp. on Principles and Practice of Parallel Programming (PPoPP 2010). 2010. January. P. 127—136.
3. *Egloff D.* High Performance Finite Difference PDE Solvers on GPUs. Technical report. QuantAlea GmbH. February. 2010.
4. *Lyuu Y. D., Wen K. W., Wu Y. C.* Performance of GPU for Pricing Financial Derivatives Convertible Bonds // To appear in J. of Information Science and Engineering [Электронный ресурс]: <http://www.iis.sinica.edu.tw/page/jise/FILE/AcceptedList/110/110444-POG.pdf>.
5. *Gaikwad A., Muni Toke I.* Parallel Iterative Linear Solvers on GPU: A Financial Engineering Case // Proc. of the 18th Intern. Conf. on Parallel, Distributed and Network-Based Computing. 2010. February. P. 607—614.
6. *Wilmott P.* Paul Wilmott on Quantitative Finance. Wiley, 2006. 1380 p.
7. *Боресков А. В., Харламов А. А.* Основы работы с технологией CUDA. М.: ДМК Пресс, 2010. 232 с.
8. *Бахвалов Н. С., Жидков Н. П., Кобельков Г. М.* Численные методы. М.: Бином. Лаборатория знаний, 2003. 632 с.
9. *Sweet R.* A parallel and vector variant of the cyclic reduction algorithm // SIAM J. Sci. Statits. Computing. 1988. Vol. 9, N 4. P. 761—765.
10. NVIDIA CUDA C Programming Guide. Version 4.2 2011 [Электронный ресурс]: <http://developer.nvidia.com/nvidia-gpu-computing-documentation>.

Сведения об авторах

Дмитрий Николаевич Шинкарук — аспирант; Санкт-Петербургский национальный исследовательский университет информационных технологий, механики и оптики, кафедра вычислительной техники; E-mail: dimashink@gmail.com

Юрий Александрович Шполянский — д-р физ.-мат. наук, доцент; Санкт-Петербургский национальный исследовательский университет информационных технологий, механики и оптики, кафедра фотоники и оптоинформатики; Тбрикс АБ; ведущий математик; E-mail: shpolyan@mail.ru

Михаил Сергеевич Косяков — канд. техн. наук; Санкт-Петербургский национальный исследовательский университет информационных технологий, механики и оптики, кафедра вычислительной техники; E-mail: mkosyakov@gmail.com

Рекомендована кафедрой вычислительной техники

Поступила в редакцию 08.02.12 г.